Cooperative Internet Access in Resource Constrained Environments

Dissertation
zur Erlangung des Doktorgrades der
Mathematisch-Naturwissenschaftlichen Fakultäten der
Georg-August-Universität zu Göttingen

vorgelegt von

Martin Stiemerling

geb. in Köln

Göttingen 2011

D7

Referent:	Professor Dr. Xiaoming Fu
Koreferenten:	Professor Dr. Dieter Hogrefe und
	Professor Dr. Henning Schulzrinne

Tag der mündlichen Prüfung: 28. Februar 2011

Abstract

Access to the Internet has become available at virtually any place in developed and developing countries, either by fixed-line or wireless access. The Internet users are used to check emails, read web pages, or see a video at any time via the Internet and are annoyed if the access does not work or does work but too slowly for the used application. The access to the Internet is stable and fast enough in metro areas, but even in these area the access network is just too slow under certain circumstances. In many other situations, such as in rural areas or while being on the move, the network capacity is usually not sufficient to keep up with the capacity demand of the applications used by the users. The easiest conclusion in cases where the network access is too slow for a specific application is simply: *give up*. Another approach is to call for an upgrade of the network to offer more capacity, but this is not as easy, as this involves the network operator to do the upgrade.

We call such a situation where the Internet access is too slow a *resource constrained environment*. This thesis presents throughput measurements of such an environment for 2 UMTS-based mobile wireless networks to show the limits of a real deployments.

However, typically there is not a single user in an area with such a limited Internet access, but multiple users with their Internet terminal. We use the possibility of having multiple users in the same area to let them cooperatively retrieve data from the Internet. The scheme we introduce allows a group of users in physical proximity to jointly access a resource or resources, by efficiently using the Internet access ressources each user contributes to the group, with the goal to overcome a resource constrained environment of each single host. The basic idea is: use several (fixed-line or wireless) Internet access links at the same time and distribute the data retrieval load among them (“application layer channel bonding”). This assumes the presence of a 2nd high-bandwidth local network (e. g., WLAN in ad-hoc mode) which allows content redistribution between these users free of charge.

The framework we are proposing is called *Cooperative Internet Access* and is defined on the conceptual level in this thesis and elaborated with 2 applications using the frame work.

We show how the *cooperative Internet access* enables near live peer-to-peer video streaming in a *resource constrained environment* where it would be otherwise impossible to watch video. We present the system design for peer-to-peer video streaming in such an environment and evaluate it with a simulative study which uses the results of the above mentioned 3G measurements for the network environment.

As 2nd application we use web browsing which is also an application type which suffers from a *resource constrained environment*, as the web content is delivered too slowly to the users, making most of today's web pages hard to use. The system design for our *cooperative web access* approach re-uses the foundation framework, as also used by the peer-to-peer use case, to improve the access to web pages.

Zusammenfassung

Der Zugang zum Internet ist heute an vielen Orten in Industrieländern und auch Entwicklungsländern entweder per Festnetz oder Mobilnetz möglich. Dies wiederum erlaubt es den Internetbenutzern quasi überall und zu jeder Zeit Emails abzurufen, Webseiten zu lesen, oder Videos abzurufen, wobei aber eine Störung oder ein zu langsamer Internetzugang die Benutzer eher verärgert. Der Zugang zum Internet ist in der Regel nur in Städten schnell und auch stabil genug für die heutigen Internetseiten. In vielen ländlichen Gebieten und auch auf Reisen außerhalb von Ansiedlungen, ist die Netzkapazität nicht ausreichend um die von den Benutzern gewohnten Netzinhalte in der erwarteten Geschwindigkeit und Zuverlässigkeit zu übertragen. Der offensichtliche Schritt im Falle einer zu langsamen Internetanbindung für eine Anwendung ist aufgeben. Ein anderer Schritt wäre auf eine Verbesserung der Anbindung seitens des Netzanbieters zu warten, was aber eine sehr lange Wartezeit zur Folge haben könnte.

Solch eine Situation in der die Internetanbindung zu langsam ist, nennen wir im Rahmen dieser Arbeit *resource constrained environment*. Wir präsentieren Meßergebnisse zum Datendurchsatz in zwei UMTS-Mobilfunknetzen in Deutschland um die Grenzen dieser aufzuzeigen.

Typischer Weise ist aber nicht nur ein Internetbenutzer in einem Bereich mit einem eingeschränkten Internetzugang, sondern mehrere Benutzer. Wir gehen in dieser Arbeit davon aus, das es mehrere Benutzer in einem Bereich gibt, die dann kooperativ Daten aus dem Internet laden. Das hier eingeführte Schema erlaubt einer Gruppe von Benutzern die sich in räumlicher Nähe befinden, gemeinschaftlich auf eine Ressource im Internet zuzugreifen, mit dem Ziel den eingeschränkten Internetzugang jedes einzelnen Hosts zu beseitigen. Die grundsätzliche Idee ist: Benutze zeitgleich mehrere Internetzugangslinks (Festnetz oder Funknetz) und verteile den Datenaustausch zwischen ihnen ("Application-Layer Channel Bonding"). Das setzt ein 2. lokales Netz mit einem hohen Datendurchsatz, z.B. WLAN im Ad-Hoc Modus, vorraus, welches eine kostenlose Umverteilung der Daten zwischen den Benutzer erlaubt.

Das hier vorgestellte Rahmenwerk nennen wir *Kooperativer Internet-Zugang* (engl. *Cooperative Internet Access*), welches in dieser Arbeit konzeptionell erarbeitet wird und an Hand von 2 Beispielanwendungen weiter ausgearbeitet wird.

Wir zeigen wie der *Kooperativer Internet-Zugang* benutzt wird, um Peer-to-Peer Video-Streaming in einem *resource constrained environment* zu ermöglichen, wo es ohne diesen Ansatz unmöglich ist Streaming-Videos anzusehen. Wir stellen das Systemdesign für Peer-to-Peer Video-Streaming in solch einer Umgebung vor und evaluieren das System mit einer Simulationstudie, welche auf den Messungen der UMTS-Mobilfunknetz beruht.

Als zweite Anwendung nehmen wir Webbrowsing, da diese Anwendungsart ebenfalls durch eine *resource constrained environment* beeinträchtigt wird. Eine zu langsame Übertragung von Webseiten erschwert den Zugang für Benutzer zu diesen Webinhalten. Das Systemdesign für den *Kooperativen Web-Zugang* (engl. cooperative web access) verwendet ebenfalls das Rahmenwerk des *Kooperativer Internet-Zugang*, um den Zugang zu Webseiten zu verbessern.

Contents

List of Figures

List of Tables

Abbreviations and Symbols

Abbreviation

802.1	Ethernet High Level Interface (Internetworking)	12
AAA	Anytime, Anywhere, Anything	1
ADSL	Asymmetric Digital Subscriber Line	30, 51
AJAX	Asynchronous JavaScript and XML	105, 108
ALMI	Application Level Multicast Infrastructure	21, 22
ATM	Asynchronous Transfer Mode	12
BET	Bit Error Rate	37
BGP	Border Gateway Protocol	10
BRAS	broadband remote access server	39
CARP	Cache Array Routing Protocol	112
CATV	Cable Television	30
CDMA2000	Code Division Multiple Access 2000	30
CDN	Content Delivery Network	106
CUBS	coordinated upload bandwidth sharing	16, 62
DHT	Distributed Hash Table	19, 21
DNS	Domain Name System	11
DSLAM	Digital Subscriber Line Access Multiplexer	39
DTN	Delay Tolerant Networking	13
EDGE	Enhanced Data rates for GSM Evolution	31
EMS	Encoded Multipath Streaming	15

ESM	End System Multicast	21, 22
ETA	Estimated Time of Arrival	117
FEC	Forward Error Correction	24, 25, 85, 88
FFD	First-Fit-Decreasing	85
FTP	File Transfer Protocol	19, 64
FTTH	Fiber To The Home	30, 51, 107
GPRS	General Packet Radio Service	26, 31
GPS	Global Positioning System	72, 90
GSM	Global System for Mobile Communications	31
HDTV	High Definition TV	52
HIP	Host Identity Protocol	17, 72
HMTP	Host Multicast Tree Protocol	21, 22
HSDPA	High Speed Downlink Packet Access	38, 88
HTCP	Hyper Text Caching Protocol	112
HTML	Hyper Text Markup Language	63, 103, 105
HTTP	Hyper Text Transfer Protocol	30, 55, 63, 64, 103, 105, 111, 114
ICMP	Internet Control Message Protocol	40
ICP	Internet Cache Protocol	112
IEEE	Institute of Electrical and Electronics Engineers	12
IETF	Internet Engineering Task Force	11
IGMP	Internet Group Management Protocol	74
IP	Internet Protocol	10, 13
IPv4	Internet Protocol Version 4	10, 142, 143
IPv6	Internet Protocol Version 6	10
ISDN	Integrated Services Digital Network	2, 111
ISP	Internet Service Provider	5, 10, 11, 53
LAN	Local Area Network	39
LRU	Least Recently Used	85

LTE	3GPP's Long Term Evolution	49
MANETs	mobile ad-hoc networks	26
MAR	Mobile Access Router	14, 15
mDNS	Multicast Domain Name System	71, 74
MIME	Multipurpose Internet Mail Extensions	114
MIXTP	MOPED inverse multiplexing transport protocol	17
MMOG	Massively Multiplayer Online Games	20
MNO	Mobile Network Operators	37
MOPED	MObile grouPEd Device	17, 62
MOVi	Mobile Opportunistic Video-on-demand	18
MPTCP	Multi Path Transmission Control Protocol	12, 57, 63
MTU	Maximum Transmission Unit	77, 143
MVNO	Mobile Virtual Network Operators	37
NACK	Negative-Acknowledgment	77
NAT	Network Address Translator	1, 17, 36, 40
NEMO	Network Mobility	17
NORM	NACK-Oriented Reliable Multicast	77
OSI	Open Systems Interconnection	12
P2P	Peer-to-Peer	40
PAN	Personal Area Network	17
PCMP	Persistent Connection Management Protocol	13
PERM	practical end-host multihoming	16, 56, 62
PPP	Point-to-Point Protocol	12
pTCP	parallel TCP	12
RMT	Reliable Multicast Transport	77
RPC	Remote Procedure Call	64
RTP	Real-time Transport Protocol	27, 64
RTSP	Real-Time Streaming Protocol	27, 64

RTT	Round Trip Time	16, 115
SACK	TCP Selective Acknowledgment	41
SCAMP	scalable membership protocol	22
SCTP	Stream Control Transmission Protocol	12
Shim6	Level 3 Multihoming Shim Protocol for IPv6	17
SIP	Session Initiation Protocol	64
SLP	Service Location Protocol	71
SOAP	Simple Object Access Protocol	64
SPI	Security Parameter Index	146
SSH	Secure Shell	40
TCP	Transmission Control Protocol	11, 12, 116, 117, 123, 142
TDMA	Time Division Multiple Access	32, 35
TLS	Transport Layer Security	146
UDP	User Datagram Protocol	41, 115, 143
UE	User Equipment	31
UMTS	Universal Mobile Telecommunications System	1, 14, 30, 31, 49, 52
URL	Uniform Resource Locator	103, 106, 116
USB	Universal Serial Bus	39
VDSL	Very High Speed Digital Subscriber Line	52
VoIP	Voice over IP	16, 51, 64
WCDMA	Wideband Code Division Multiple Access	26, 32, 34
WiMAX	Worldwide Interoperability for Microwave Access	2, 30, 37, 49
WLAN	Wireless LAN	30, 49, 52
WLAN	Wireless Local Area Network	2, 10, 11, 13, 14, 18, 27, 71, 128, 143
WWW	World Wide Web	2, 51, 107

1 Introduction

1.1 Problem Statement

The Internet has been designed as a general purpose network with no assumptions about applications built-in. Applications in this context includes any type of protocol and attached computer process operating over the Internet. While on one hand the Internet is application agnostic, applications make assumptions about the Internet. There is a full set of assumptions made, such as, but not limited to, full end-to-end connectivity – even though there are Network Address Translators (NAT) deployed –, an always on Internet where connectivity is always available, uncongested Internet paths, and sufficient *achievable throughput* for every application. Sufficient achievable throughput is meant complementary to uncongested Internet paths. Congestion refers to an overload situation at a bottle neck on an Internet path, but sufficient achievable bandwidth refers to the application's bandwidth demand without congestion in regular operations.

The base line of these assumptions made by applications goes back to that access to the Internet seems to be available at a ubiquitous scale in developed nations and also in rising nations in areas with dense population. The term *Internet access*, as used by today, is often used in way that may suggest the Internet to be available virtually anywhere, at anytime, for any type of data (or content, using a more general term), and with any type of device (e. g., laptop, desktop computer or mobile phone). This vision is promoted under the term *Anything, Anywhere, Anytime* (AAA) [2] .

However, there are certain circumstances where the application's demand for achievable throughput cannot met due to natural limitations of the access to the Internet. More and more users are accessing the Internet with their mobile devices, for instance, with a mobile phone via a mobile wireless networks based on UMTS technology. These networks cannot deliver the same throughput in all and every location, as the combination of radio technology and environment impacts the achievable throughput just by the physical limitations of the radio channel; and not all locations supply the same amount of network capacity, as either the location is not provisioned to supply the capacity or as the used radio technology at this location has a lower capacity per se. These facts will let people using mobile wireless Internet access experience high-speed access at some times, e. g., in metro areas in the range of 1 Mbit/s, but also low-speed access in the country side in the range of 50 kbit/s, as cellular networks, such as UMTS, cover large areas of many countries with a varying degree of deployed network capacity. This results also in uncovered spots and connections that are be aborted, e. g., in railway tunnels, in some valleys, etc.

There are also many rural areas which are less densely populated and which are cut-off from high-speed Internet access. There are a number of reasons for this, but the main reason seems to be the fact that deploying such a high-speed Internet access technology incurs capital expenses for the network operator which cannot be recovered by the limited amount of potential users in an area. The users in such areas are constantly experience a low achievable throughput, as the access technology used provides low throughput. A typical example for such deployments is the use of Integrated Services Digital Network (ISDN) telephone dial-up Internet access with 64 kbit/s.

On the other hand does each application have typically a certain demand in throughput to operate properly, though this demand may depend on what the user of this application deems sufficient. For instance, web browsing via WWW may demand a sufficient achievable throughput in the range of 600 kbit/s to 900 kbit/s for youtube [3] videos [4]; and peer-to-peer video streaming systems operate with video transmission rates of 500 kbit/s to 600 kbit/s [5]. The requirement of a minimal throughput is already discussed in [6] (1997) for realtime and non-realtime applications with the apparent conclusion of non-realtime applications being more relaxed on variantions of the achievable throughput and realtime applications in strong need for a minimum achievable throughput. An interesting fact mentioned in [6] is the minimal achievable throughput for retrieving single elements of an web page between 240 kbit/s and to 1.6 Mbit/s to achieve the so-called block delay of 100 ms. This is the delay for loading a a single specific element of a single web page, whereas this web page contains of several such elements.

A shown above, there is under certain circumstances a gap between the in general achievable throughput via an Internet access and the applications demand. There are attempts to improve the situation by either deploying more wireless access capacity, by technological improvements, or by adapting the accessed content to mobile devices. There are other technologies with a higher achievable throughput, such as Worldwide Interoperability for Microwave Access (WiMAX) or Wireless Local Area Network (WLAN), but which are not either not deployed at a large scale or offer only a limited range, as compared to mobile wireless radio technologies. But there is no general solution for users being in a situation where the capacity of the Internet access cannot met the application's demand or the user's expectation, other than to give up using the Internet or to wait for the deployment of more capacity.

This gap between the requirements of the applications today (2010) and the de-facto achievable throughput of mobile wireless leads to a first conclusion:

> *A single mobile wireless interface on a single node will not be able to deliver the required throughput for the applications running on this particular node under every circumstances.*

The historical development of the maximum achievable bitrate for xDSL [7] and mobile wireless in Germany [8, 9, 10, 11, 12] in Figure 1.1 illustrates the gap between fixed line and mobile wireless. The x-axis denotes the time when the technology was available to customers in Germany and the y-axis denotes the maximum bitrate. It should be noted that the maximum bitrate for 3GPP mobile wireless is the theoretical maximum which may not be reached under real conditions (cf. Section 3). The figure illustrates the growing gap between the widely used xDSL technology and the mobile wireless access.

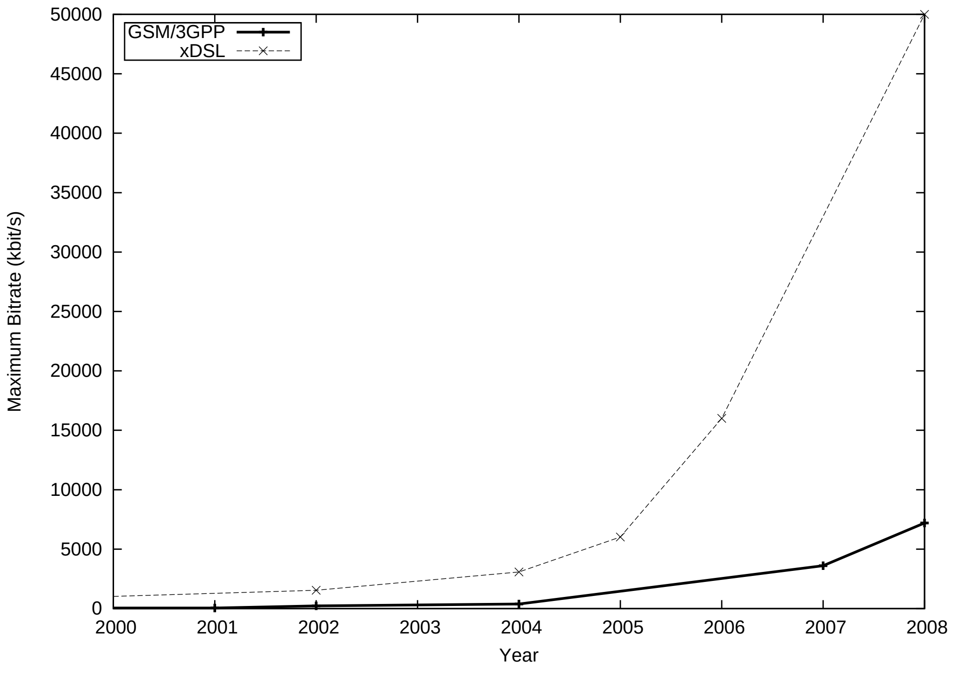


Figure 1.1: Development of the maximum downlink bitrates for xDSL and mobile wireless

1.2 Cooperative Internet Access

This thesis proposes a scheme that allows a group of users in physical proximity to jointly access a resource or resources, by efficiently using the Internet access ressources each user contributes to the group, with the goal to overcome a resource constrained environment of each single host. The basic idea is very simple: use several wireless Internet access links at the same time and distribute the data retrieval load among them ("application level channel bonding"). These links do not have to use the same technology or connect to the same network operator. Actually, diversity reduces the risk of "fate sharing", i. e., the likelihood that several links become unavailable (or very slow) at the same time. However, having several subscriptions to network operators at the same time would introduce significant costs for a user. We propose a scheme for several users to contribute their wireless Internet access resources for retrieving the content in a joint effort. This assumes that there is a high-bandwidth local network (e. g., WLAN in ad-hoc mode) which allows content redistribution between these users free of charge.

We call this approach **cooperative Internet access**, as multiple users (nodes) are cooperatively retrieving data from the network. This resource can be, for instance, a web site, a peer-to-peer video stream or a file. We define the resource as below.

Definition 1.1 (Resource): A resource is a piece of data stored in one or multiple replicas on nodes in the Internet. The resource is accessible via a network and it can be accessed in sub-units, the so-called the **the information elements**.

Definition 1.2 (information element): An information element is an atomic unit of data which can be identified by a unique handle (an identifier) on a global scale.

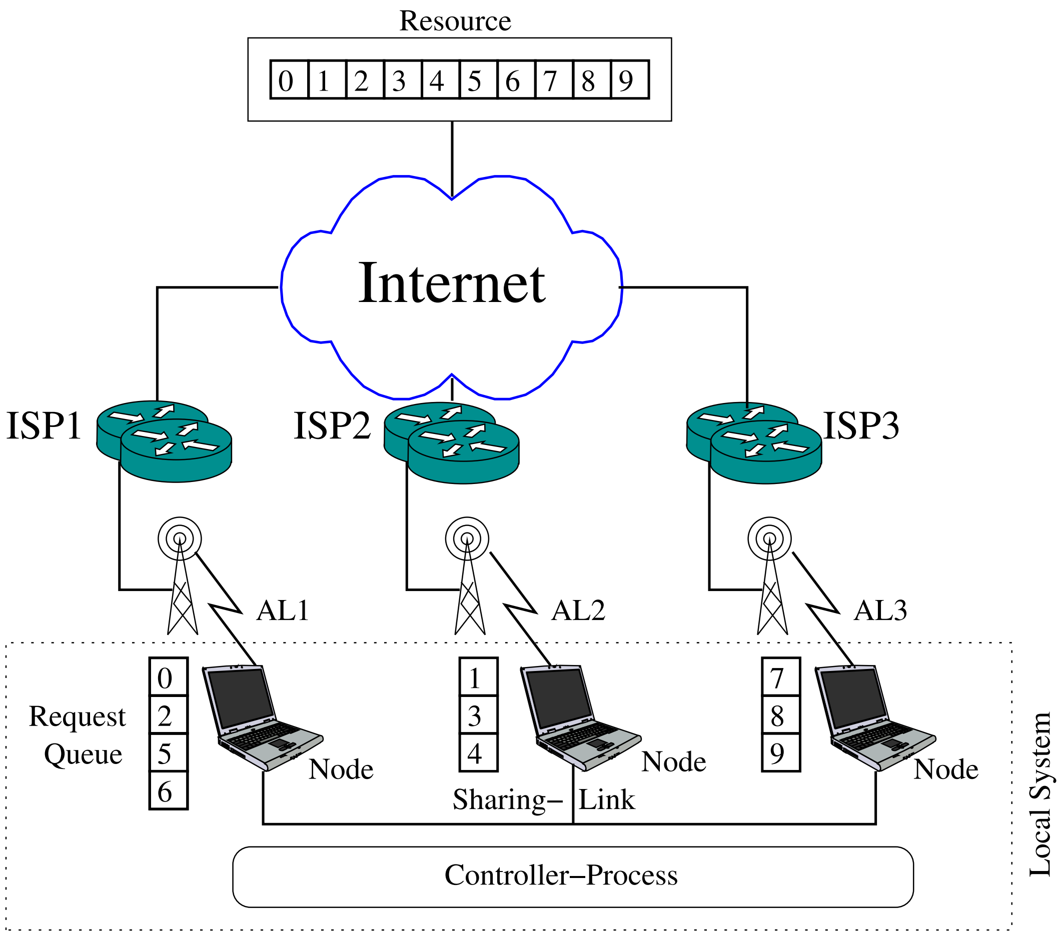


Figure 1.2: Schematic view of the cooperative Internet access

The approach exhibits these basic properties:

information centric The system requires a resource to be available in smaller units, the so-called **information elements**, or short element, so that the resource can be retrieved in elements rather as a single entity. The retrieval and redistribution uses the information elements as ”trading good” between the participating nodes.

application level striping The system aggregates the capacity of all access-links in the system to provide a better resource to the application, as compared to what each single access-link could provide. For instance, more achievable throughput or better resilience against link failures of a single access-link. The striping is implemented by retrieving the information elements, as smallest unit, over multiple access-links.

delegation A node can **delegate** the retrieval of information elements to other nodes which retrieve the elements on-behalf of the requesting node.

flow agnostic The handling of the resource and its parts is not bound to specific flows, network address, etc. But is completely handled at level of the information elements.

local support only The system requires only ”local” support of participating nodes, but not of any other node in the Internet. This eases the deployment, as no end-to-end support is needed.

system multihoming There is typically the differentiation between site-multihoming and host-multihoming. However, the proposed approach implements a **system multihoming**, as there is no need for site or host multihoming, but the whole local system is (cf. also Section 2.1.1).

coordinated access The system coordinates access to the resource where possible.

We assume each node is equipped with at least two network interfaces, the access-link and the sharing-link.

Definition 1.3 (access-link): We commonly denote the link providing the access to the Internet as **access-link**. This link can be of any type of technology and it can be connected to any network operator which provides access to the Internet.

Definition 1.4 (sharing-link): We commonly denote the link providing the high-bandwidth local network connectivity amongst the nodes of the users as the **sharing-link**. The sharing-link interconnects the participating nodes in physical proximity to detect each other, to exchange coordination messages, and to redistribution of data retrieved via the access-links. The sharing-link is assumed to provide a larger bandwidth as required to retrieve the element and to coordinate the redistribution.

Figure 1.2 shows an example setting for the cooperative Internet access with 3 nodes. Each **local node** is connected via its access-link (AL1 to AL3) to an network operator (Internet Service Provider (ISP) in the figure) and it is also connected to the sharing-link. All **local nodes** form the **local system**. The accessed resource can be located either on a single **remote node** or on multiple **remote nodes**. It is divided in information elements, for instance, with an index starting from 0 to 9, in Figure 1.2. Each local node has a request queue which states the order of elements to be retrieved by that particular node. A controller process, which either be hosted by one of the local nodes or be distributed across the local nodes, determines which local node shall retrieve what information element.

Here are the definitions of **local system**, **local node**, and **remote node**:

Definition 1.5 (local system): The nodes participating in the cooperative Internet access are part of the **local system**.

Definition 1.6 (local node or local peer): A node which is part of the **local system** is denoted as **local node** or **local peer**.

Definition 1.7 (remote node or remote peer): A node which is not part of the **local system** is denoted as **remote node** or **remote peer**.

The transmission of information elements over untrusted local nodes raises the question about security. Such an untrusted local node, as well as any other node in the Internet, can read, change, or suppress the information elements transferred via it. We assume that the data transmitted via these nodes is either protected by the application itself, e. g., by cryptographic hashes applied to each information element in peer-to-peer systems [13], or by deciding to forward only less important data via these nodes, e. g., banner pictures of web pages. Furthermore, a trusted local node could be coerced to handle information elements which violate law, e. g., handling of illegally copied movies in peer-to-peer systems. This last threat might be difficult to mitigate.

The further consideration of security is out of scope of this thesis.

1.2.1 Use Case 1: Peer-to-Peer Video Streaming

Near-live peer-to-peer video streaming (e. g., with PPLive [14]) is increasingly popular among Internet users, but typically it is limited to fixed Internet access. Video streaming, no matter whether based on client/server or peer-to-peer paradigm, requires – depending on content type and image quality – a sustained bit rate in a range from 0.3 to 16 Mb/s, which can be achieved easily with wireline Internet access such as xDSL. In contrast, this is challenging for users accessing the Internet via wireless links. They basically have the choice between high speed or good coverage area, but usually they cannot have both at the same time, i. e., leaving them in a resource constrained environment: video streaming for peer-to-peer systems requires approximately 600 kbit/s achievable throughput which cannot be delivered via a single mobile broadband wireless link, as we show in Chapter 3.

Let's assume multiple users are riding on the same train and their are interested in watching the same TV channel supplied by a peer-to-peer video streaming system. Each user on its own cannot watch the video due to limitation of the wireless link's achievable throughput. The set of users is now using the cooperative Internet access to distribute the video retrieval amongst their nodes (and thus each node's access-link) in a coordinated way, leavering the resource of each access-link. The nodes use the sharing-link to coordinate the video retrieval and to redistribute the video locally. This will enable the users to watch the TV channel as the set set of nodes is cooperatively retrieving the video, instead of letting each node acting selfishly.

1.2.2 Use Case 2: Web Browsing

Web browsing is the most popular way of retrieving content from the Internet. However, web browsing of today's web page in resource constrained environments requires either adapted content, e. g., specialized mobile web pages, or other support. However, specialized web pages are not always available and even these pages require a decent capacity of the access-link. One traditional way of deadling with slow, resource constrained access-lines is the usage of HTTP cache proxies, and in fact we reuse and extend this approach. The original approaches are limited to the use of a single access-link only, i. e., all nodes are located behind the same access-link.

The cooperative web browsing allows the coordinated access to content by multiple users, for fully shared, partially shared, and content that cannot be shared. A web page consists of multiple elements, such as pictures or videos, and these information elements can be either cached in the local system or they can be retrieved via one of the access-links.

1.3 General Definitions

This section gives some definitions required for the general understanding of the remaining document.

Definition 1.8 (maximum bitrate): We define the **maximum bitrate** $\hat{B}_\ell$ as the gross bitrate a particular access-link offers.

Definition 1.9 (achievable throughput): We define the **achievable throughput** Θ as the actual throughput on a single Internet path which is effectively usable by an application. This excludes any further header below the application, i. e., it is the net throughput.

We differentiate between the achievable throughput Θ and the maximum bitrate of an access-link $\hat{B}_\ell$. The maximum bitrate is determined by the used technology and only gives a rough estimate what this particular links is able handle in terms of throughput. The overall achievable throughput is always determined by the bottleneck of the complete Internet path between source and destination, but we assume the bottleneck to be located at the access-link for our further considerations. This assumption is solely made for the ease of the further discussions. The approach discussed in this thesis applies also to situations where the bottleneck is not at the access-link.

This differentiation is necessary as there may be huge differences between the maximum bitrate of an access-link technology and the actual achievable throughput. The maximum bitrate is the theoretical upper bound of the gross throughput (i. e., which is the total of achievable throughput and the throughput required for the headers below the application level).

We also define a **instantaneous combined download capacity** to use it later on in the thesis.

Definition 1.10 (instantaneous combined download capacity): The **instantaneous combined download capacity** of the local system is the aggregated throughput of the access-links of a local nodes.

Definition 1.11 (network availability): The **network availability** denotes the probability of the network being available to deliver any achievable throughput.

The network availability is another factor to be considered, as not every network will always be able to deliver throughput. For instance, fixed-lined access links will be almost always available (unless there is a network fault), but wireless access links will by their nature have periods with no achievable throughput.

The next definition 1.3 deals with what applications require in terms of achievable throughput and also in terms of network availability; and when applications or nodes are in *resource constrained environment*.

Definition 1.12 (resource constrained environment): An Internet application has a minimum required achievable throughput Θ_{min} to work as expected by the user of it. We denote a situation a *resource constrained environment* where the achievable throughput is below this minimum for all times or at least for a non-neglectable amount of time: $\Theta(t\prime) < \Theta_{min}\ t\prime \subseteq t$

1.4 Contributions of this Thesis

The main contributions of this thesis are:

1. We identified the resource constrained environment and challenged the status quo of today. Nodes in such an environment either cannot access the Internet as the users simply give up or there is the need to adapt the accessed content to the particular environment. But content adaptation requires action from entities not involved in the environment.
2. The quantification of the achievable throughput and the network availability in a resource constrained train scenario through measurements (Chapter 3).
3. We propose a Cooperative Internet Access where multiple nodes access one or multiple resources in a joint and coordinated effort to minimize the number of information elements to be retrieved by the participating nodes.
4. The specification of cooperative Internet access system and the solution design for two use cases: peer-to-peer video streaming and web access.
5. The simulation of the peer-to-peer video streaming use cases with a new bin packing algorithm for the peer-to-peer chunk scheduler and a light-weight throughput estimator.

1.5 Outline

First, we will examine the related work in the field of cooperative Internet access and related fields in Chapter 2 and discuss the differences between prior works and our approach.

Chapter 3 lays the foundation to understand the network environment of mobile Internet users, by introducing the environment of mobile wireless broadband networks. Subsequently we set the measurement methodology for measurement of such mobile wireless broadband networks, present the measurement results, and an analysis of the measurements. This chapter is the basis to understand the need for our system, as it shows the need for applications to cope with resource constrained environments.

The conceptual design of a Cooperative Internet Access approach is outlined in Chapter 4 and also the implementation options for the system.

The usage of the conceptual system design for peer-to-peer video streaming in resource constrained environments is described in Chapter 5. This chapter presents the peer-to-peer system design, an estimator to forecast the achievable throughput on an access-link and a new scheduler for peer-to-peer video streaming in our setting.

The usage of the conceptual system design for web browsing in resource constrained environments is described in Chapter 6. This chapter discusses the basic properties of web browsing and current findings on web pages and their caching, and the system design.

Chapter 7 presents the simulator model for the peer-to-peer video streaming usage of the cooperative Internet access approach, the evaluation metrics, and the findings of the simulation. It would be possible to terminate all these connections in one mobile device with several wireless interfaces, and in fact we have analyzed this configuration in our simulation.

The thesis concludes with Chapter 8 with conclusions, open issues, and an outlook.

2 Related Work

This chapter examines and discusses related work to the our approach, as Cooperative Internet Access shares properties well-known from other areas. We start with discussing individual parts related to Cooperative Internet Access, such as multihoming or striping, in Section 2.1 and Section 2.2, while Section 2.3 and Section 2.4 discuss related systems. The basics of peer-to-peer networking and peer-to-peer video streaming are introduced in Section 2.5 and Section 2.6. The related work of mobile peer-to-peer video streaming is explored in Section 2.7. This chapter closes with a comparison of the various related work and a discussion in Section 2.8.

2.1 General Networking

2.1.1 Multihoming

IP multihoming, independent of whether IPv4 or IPv6, allows to use multiple links to the Internet to improve the reliability of the Internet access for a complete network site, e. g., an enterprise network, or a single host. Multihoming for a network site is commonly called site multihoming or enterprise multihoming while multihoming for hosts is commonly called host multihoming.

Site multihoming is introduced in [RFC 2260], where a single site connects to multiple Internet Service Providers (ISP) to protect the site against a failure of one of the ISPs, as a protection against a single point of failure. Site-multihoming is implemented by announcing the IP prefix of the site to more than ISP via a routing protocol (shown in Figure 2.1(a)). By today (2010), the Border Gateway Protocol (BGP) [RFC 4271, RFC 1998] is commonly used to implement site multihoming. However, the end hosts in the site cannot actively influence or participate in the site multihoming, other than by the benefit of the improved reliability. The hosts typically have a single IP address and rely on the routing to select one of the uplinks to the Internet.

Host multihoming is an emerging topic, as most hosts have multiple interfaces, e. g., Ethernet, Wireless LAN, or UMTS, but rarely use these multiple interfaces (i. e., the access-link) at the same time. The main reasons are:

- there is typically only one access-link offering Internet access (e. g., wired access in the office or WLAN in a coffee shop);
- the applications are designed and built to use a single IP interface only;

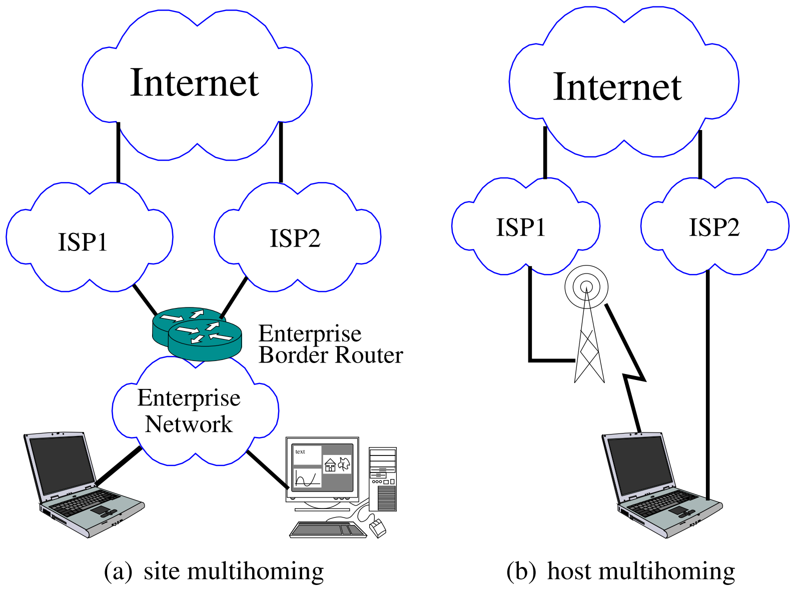


Figure 2.1: Multihoming

- there is no real multihoming support in today's operating systems, e. .g, there are no helper functions to support applications (see also next point) and there are no transport protocols supporting multihoming available in the operating systems right now (see in Section 2.1.2 on multipath TCP for more about this);

- there are configuration conflicts if multiple access-links are connecting to the Internet and each receives a different set of configuration information (e. g., different DNS server, different default routes, etc). This issue is current examined in the IETF Multiple Interfaces (MIF) working group [15].

Figure 2.1(b) schematically shows a host with 2 interfaces connecting to 2 different Internet Service Providers (ISP) to access the Internet. These single host multihoming-homing approaches, such as in [16], are typically well explored for static WLAN deployments, but not beyond towards more challenging environments, such as, on UMTS networks, and only in testbeds but not in actual deployments on hosts.

Site multihoming is not applicable for our approach, as the access-links are of the nodes are considered as the limiting factor. However, the network operators may use site multihoming within their network. Host multihoming is applicable, as a node may have multiple access-links, but it is unlikely to have a host with two Internet connections at the same time, simply to cost reasons. Two Internet access-links will usually require two subscriptions or is limited to place with, for instance, UMTS and WLAN deployments. At least by today's measures, most users have a single mobile wireless subscription only.

2.1.2 Striping

A key element of Cooperative Internet Access is the idea of *logically bundling Internet access-links* of several nodes to overcome a resource constrained environment. The network resource of each link is aggregate for the application on top of it, but the application's data is actually splitted up and send across multiple paths, or at least sub-paths, through the Internet. This approach of resource aggregation is well-known in the network field under several names, such as, for instance, as striping where multiple physical resources are aggregated to achieve a higher performance. Striping can be implemented at different levels of the network stack [17, 18]. We discuss the various striping options on each OSI layer:

Link Layer Striping Link layer striping aggregates multiple links to a single logical link for the network layer. The striping at this layer depends on the used technology and can be implemented at the byte layer or at the cell layer (ATM) [17] or frame layer IEEE 802.1 (Ethernet High Level Interface) links [19]. This striping approach can only be used if the links to be aggregated are of the same technology and if the particular technology supports striping. Implementation of this approach requires at least changes to the network hardware (e. g., switches and network host interfaces), but probably also to network host interface software drivers. Another link-layer related technology is the Point-to-Point Protocol (PPP) [RFC 1661] with its PPP Multilink protocol [RFC 1990] extension.

Network Layer Striping Network layer striping makes the striping independent of the used link layer technology and thus also goes beyond a single link-layer domain, i. e., it can run end-to-end spanning multiple link layer technologies. At this level, IP packets can be scheduled to the various links to achieve the aggregation. IP-in-IP tunnels to realize striping is proposed in [20], relying on modifications in the IP stack at both communication ends.

Transport Layer Striping Striping at the transport layer requires support from the transport layer protocols, such as, TCP [RFC 793] or SCTP [RFC 4960], and allows to use multiple Interfaces concurrently, but requires adapted applications that know how to handled such, so-called, multipath protocols. However, TCP and SCTP are not multipath capable by their nature: TCP is a point-to-point connection between 2 hosts and SCTP follows the same principle with the capability of *path failover*. SCTP's path failover refers to the ability to create and keep multiple connections (SCTP association) between 2 hosts, while one is active and the others are used as a hot stand-by and one of those is used if the active association fails, otherwise not. There are multiple approaches to modify or augment TCP to support striping. Multipath TCP [21] lets hosts use multiple TCP connections via differnet network interfaces while ensuring fairness of these multiple flows with other TCP flows in the network. A similar approach to MPTCP is parallel TCP (pTCP) [22] that considers wireless links as to be the bottleneck and not the complete network path.

Application Layer Striping Parallel sockets (Psockets) [23] is an early work on striping with TCP, but with the original goal of improving the overall throughput between hosts by opening multiple TCP connections (sockets) simultaneously. The striping itself is done as an application-level library. This approach raises serious concerns about TCP fairness in the network, as one application is not competing with a single TCP flow with other flows in the network but with multiple flows at the same time. The approach can basically be reused to support striping over multiple Interfaces.

Session Layer Striping The position paper [18] argues for a session layer striping where this layer gets hints from the application about the data characteristics about to be sent (e. g., reliable or unreliable, bulk or real-time) and the session layer can decide on its own if multiple connections are used. However, in fact the proposal is an enhanced version of the parallel Sockets described in the application layer striping part.

The above striping techniques are applicable in scenarios where a single node is the origin of striping and where typically the other communication end on the particular level also supports the specific technique (with the exception of PSockets). These techniques are not applicable to our approach, as the cooperative Internet access uses the access-links of multiple nodes to achieve application level striping across these nodes and not just a single. However, nodes which are participating in our approach may use transport layer striping on their access-link to retrieve data.

2.1.3 Intermittent Connectivity

Today's Internet is assumed to be always on, i. e., the resources of the network and in the network are always reachable. But there are a number of cases where this is not applicable, sometimes referred to Intermittent Connectivity in the literature, discussed in this Section.

A node (or a set of nodes) which is disconnected from the Internet for longer times or which use communication links with a extreme delay, but these nodes still need to participate in data exchanges with the Internet or an IP-based network. For instance, but not limited to, in these cases:

Terrestrial Mobile Networks Nodes connected to UMTS networks will also experience disconnection periods, but still have the need to exchange data with the Internet [24].

Space Networks Space nodes, e. g., such as satelites or deep-space exploration vehicles (probes), that have low-bandwidth and high-delay communication links [24] and probably long disconnection periods.

Sparsely Deployed Access Points Situations where access points to the Internet can only be deployed in limited areas. Such a model was initially proposed in the Infostations model [25] and one example implementing such an approach is the "Drive-Thru Internet" approach [26] where WLAN access points are deployed at some locations on the motorway, e. g., bridges or gas stations, but there is no full coverage of the motorway. The Persistent Connection Management Protocol (PCMP) is used to maintain the connectivity for the applications, even though the Internet connectivity is only given in a few spots.

The above is in general classified as Delay Tolerant Networking (DTN) [24] and is intended to be a complementary architecture to the Internet [RFC 4838].

There is another class of intermittent connectivity where nodes from ad-hoc networks to exchange data and where these nodes are without connectivity afterwards. An application of the these ad-hoc networks, as one example out of many, is car-to-car and car-to-roadside communication [27]. This concept was initially envisioned for safety or traffic condition messages for

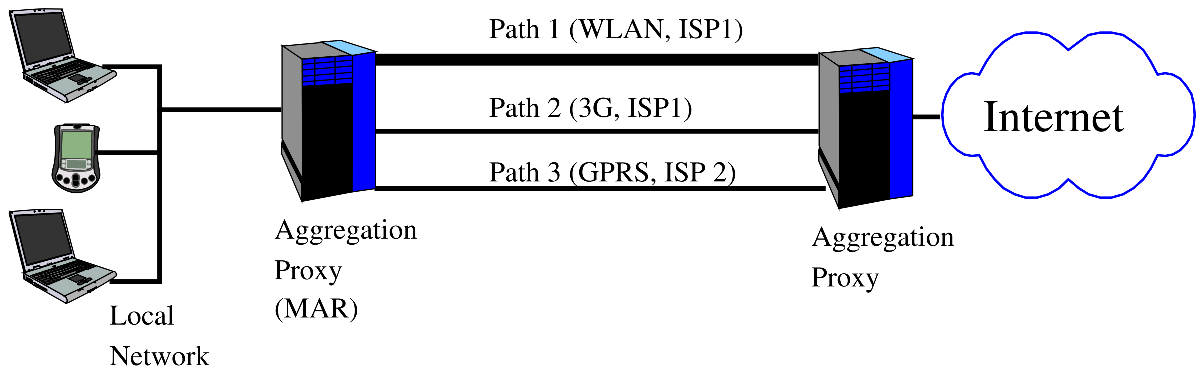


Figure 2.2: Mobile Access Router (MAR) approach multipath aggregation

cars and drivers, but recently multimedia communication is emerging in this field [28, 29], with a focus on how to enable video delivery in such a setting.

Cooperative Internet Access falls also in this area, as we try to mitigate intermittent connectivity of individual nodes by combining several access-links of several nodes. The DTN approach looks for a generalized network approach, as the Internet does, to be application agnostic. However, Cooperative Internet Access is not application agnostic, as only the knowledge about the application's needs and also the chance to influence the behavior ensures the proper adaptation to a resource constrained environment.

The ”Drive-Thru Internet” and Cooperative Internet Access are conceptually similar, as both approaches try to enable applications to run in similar environment, but differ in the system approach, as the ”Drive-Thru Internet” assumes a single access-link per node and no cooperation between the nodes.

2.2 Multipath Aggregation Systems

An Internet-style approach to link aggregation is to use transport layer striping, as applications in need of aggregating multiple access-links can directly interact with the transport protocols, but less with lower layers, due to limits in implementation, i. e., the transport protocols are the "face" of the network stack to the applications. The discussed multipath aggregation systems discuss make use of this to overcome a resource constrained environment to increase the application performance for the user. The differences between approaches are in the place where the aggregation happens, e. g., in the access network or access the Internet.

The Mobile Access Router (MAR) approach in [30] aims to improve data performance for wireless users by aggregating multiple wireless access-links. The MAR uses multiple access-links (e. g., UMTS and WLAN) to connect to the Internet, as shown in Figure 2.2.. The MAR and a network-side proxy are in charge of handling the various access-links and the scheduling of the data traffic across these. The nodes connecting to the MAR are using the Internet as used to. All their traffic is forwarded to the MAR proxy that is the anchor for all traffic to and from these nodes. However, this concept is based on deployed infrastructure, i. e., the MAR must be

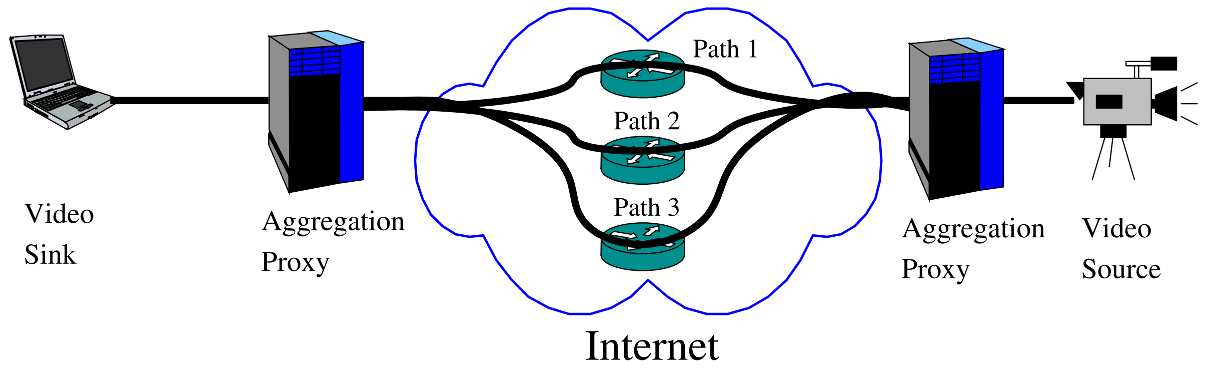


Figure 2.3: EMS-approach multipath aggregation

installed in a commuter bus and the MAR proxy must be located at a well-connected Internet site.

The Encoded Multipath Streaming (EMS) [31] approach implements multipath aggregation to enable high-quality video streaming over the Internet. EMS assumes the availability of multiple Internet paths between the video source and the video receiver that can be used simultaneously, as shown in Figure 2.3. The video stream is split into equal fractions according to the number of available paths and depending on the loss rate on the paths adapted to minimize the loss. The approach considers only a scenario with a single video sender and a single video receiver without discussing details about how the multiple paths are constructed (if any at all) and how multiple such settings interwork in resource constrained scenarios where there is just not sufficient throughput to a single node.

The Mobile Access Router (MAR) approach has similarities with Cooperative Internet Access as it also mitigates the risk of a single access-link failure by aggregating multiple links. The difference is the aggregation point: MAR requires the deployment of host and network side aggregation proxies while Cooperative Internet Access does not require any further infrastructure, as the set of participating nodes is the aggregation point and only on the side of the local Cooperative Internet Access system.

The Encoded Multipath Streaming (EMS) approach is not directly comparable with Cooperative Internet Access, as it targets multi-paths across the Internet but has a single access-link. EMS will suffer in resource constrained environments, as any other traditional application. EMS could possible be extended with the MAR approach or the Cooperative Internet Access approach, but we do not examine this in more detail.

2.3 Collaborative Internet Access

The core idea behind Cooperative Internet Access is that several nodes share their access-link with other nodes to overcome a resource constrained environment and their joint effort to download a resource. This is an extension of the Collaborative Internet Access where multiple access-links are shared amongst a set of nodes but without a joint effort to download a resource, but to satisfy the nodes demand only. This section discusses the Collaborative Internet Access and the following Section 2.4 discusses the cooperative approach.

Cooperative Internet Access makes use of the the access-links of several nodes where each node shares its access-link and also has a second link that is used to find other interested nodes, to coordinate the access, and to exchange data. The concept of collaborative Internet access is proposed in *flow scheduling for end-host multihoming* [32]. The framework is called "practical end-host multihoming (PERM) in the residential area" and allows users to re-use other user's Internet access by assigning outbound flows to the access-link of other users via a shared-link, e. g., WLAN. This shared-link is shared between the active nodes but not across all participating local nodes, in contrast to the sharing-link of this thesis, where all active nodes share the sharing-link.

New flows are scheduled at flow initialization time to one of the access-link and remain on that link for the complete flow time. PERM implements a UDP and TCP flow predictor that aims at predicting if the flow is a light-volume flow or a heavy-volume flow. However, the flow predictor is much focused on HTTP traffic and is not considering other traffic types appropriately, such as, e. g., Voice over IP (VoIP) or peer-to-peer applications. The flow scheduling follows this principle if a flow is anticipated to be below a threshold of 8 KB it is placed on the link with the lowest Round Trip Time (RTT). Flows that are anticipated to be larger are placed based on the anticipated volume and the current load situations on the access links. This requires a well working anticipation scheme which is outlined for HTTP-based traffic in the paper but not beyond.

A similar approach to PERM is the coordinated upload bandwidth sharing in residential networks (CUBS) [33]. CUBS uses several fixed-line access-links that are reachable via Wireless LAN. The aim is to re-use the upload bandwidth of these other links to improve the upload throughput of peers locally attached in peer-to-peer file sharing. This paper presents an extensive study about the upload capacity of residential access networks in the USA to discuss the availability of idle upload capacities. CUBS conceptually follows PERM that for each new TCP flow a measured-based scheduler is selecting the local or a remote access-link. PERM schedules new TCP flows to a remote access-link if this link is idle (or has spare capacity) and the TCP flow is upload intensive. Otherwise new TCP flows are pushed to the node's own access-link.

PERM, CUBS and Cooperative Internet Access allow a node to share its access-link with other nodes in physical proximity. PERM and CUBS allow a node to assign a flow to another access-link under the assumption of idle capacity on the access-link, but the data retrieved with this flow is only intended to the requesting node. There is no redistribution of the data envisioned, as only the throughput improvement of individual nodes is considered. This is applicable for the mentioned residential access networks, but not applicable in our scenario. In Cooperative Internet Access, the idle capacity of the access-link is assumed to be so low that there is no real capacity left to satisfy the selfish demand of each node, but there is sufficient capacity to satisfy the joint demand of all nodes.

An advantage of PERM and also CUBS are unmodified applications (where Cooperative Internet Access does require changed applications), but changes below the applications. Both approach can assign a flow to an access-link at flow initialization time and cannot change it afterwards anymore. However, the ability to use one or the other access-link at any point of time (and changing between them) for the same information element is required for more challenging environments and shows the limit of flow-based approaches. For the scheduling it is not so important how particular flows are behaving, but if the information elements will arrive in-time in the local system.

The changing between different access-links for both approaches can be facilitated by using other mobility or multihoming techniques that allow to reallocate flows to different IP addresses and thus also to different host interfaces, such as Mobile IP [RFC 3344, RFC 3755], or by the Host Identity Protocol (HIP) [RFC 5206], or with Level 3 Multihoming Shim Protocol for IPv6 (Shim6). However, using such approaches always requires further signaling, in the case of moving a flow to a different IP address, and also support from both communication ends and probably also public IP addresses or some form of NAT support (e. g., for HIP [RFC 5770]). Moving a flows will also increase complexity in scheduling these flows, as an anticipated move would be a re-scheduling on the flow level.

The MOPED (MObile grouPEd Device) [34] aggregates also the access-links of multiple devices to achieve striping for a single node. The devices are envisioned to belong to a single user and they are also using a sharing-link (a Personal Area Network (PAN) to act in a coordinated way. MOPED is similar to the IETF Network Mobility (NEMO) [RFC 4885], where complete networks are moving (compared to a host only moving with Mobile IP [RFC 3344, RFC 3755]). MOPED hides the mobility of the moving PAN by using a network proxy that is the anchor point for all communication to and from the moving PAN. The nodes in the PAN coordinate their access-links in terms of available throughput and they use a specialized protocol called *MOPED inverse multiplexing transport protocol* (MIXTP) to spread the data transmission over multiple links.

MOPED approach is similar to CUBS, PERM, and Cooperative Internet Access, with the same caveat as CUBS and PERM. The participating nodes are selfishly retrieving the data. MOPED requires also the deployment of a network-side proxy to support the aggregation.

2.4 Cooperative Internet Access

This section elaborates first the peer-to-peer related work relevant for the approach presented in Section 5 and afterwards the related work for the web browsing case presented in Section 6.

2.4.1 Peer-to-Peer Related

Peer-to-peer video streaming in mobile environments attracts a lot of attention as this environment is much more challenging as compared to client/server video streaming applications [35]. Client/server video streaming requires a decent download rate from the server to the client while peer-to-peer video streaming requires download and upload rate at the same time plus the challenges resulting from the (more or less) complete decentralization.

A related technique to cope with peer-to-peer and resource limited access-links is detection of peers that are located on the same access-link. Bittorrent [36], a popular file sharing application (as of today (2010)) uses a link-local node discovery protocol to detect if other peers are in the same link segment (e. g., in the same Ethernet broadcast domain). The technique is for home users that have a home gateway connecting the home network to the Internet and all hosts are on the same link-layer link [37]. The main benefit of locating peers with the same content on the local link is to avoid that the same data, a chunk in peer-to-peer, is retrieved multiple times via the access-link. The nodes can exchange chunks they already have directly via the local network, instead of clogging the access-link with redundant chunk downloads. The approach does not coordinate the chunk retrieval amongst the local nodes, but just lets them exchange the current status of the already retrieved chunks.

The Mobile Opportunistic Video-on-demand (MOVi) [38] approach assumes that there are only sparsely deployed access points to the Internet and that only a few nodes are actually connected to these. MOVi assumes a centralized MOVi server that is in charge of controlling the participating nodes and scheduling of the data transfers between the participating nodes. Only a few nodes are connected to the MOVI server and the main video data exchange happens between the nodes that are in reach of a WLAN. The participating nodes use an 802.11 extension [38] to enable a direct communication between the nodes which enables the video exchange between them. This increase the overall system capacity without increasing the load on the server or the need to use more access-links to connect to the Internet in order to carry the additional load.

PatchPeer [39] assumes that the wireless access network supports multicast to deliver video on demand to a set of mobile nodes. A node that is missing parts of the multicast delivery could retrieve these parts with a unicast transmission from the server, consuming resources on the wireless part which may be also used by other nodes retrieving data. However, it is assumed that there are also other nodes in the vicinity of the node in need for a video patch (i. e., the missing parts). These nodes can deliver the missing video directly to the node, instead of letting the node fetching the patch via the access-link from the server. This approach works well in rather static situations, where the achievable throughput on the wireless part is stable, and where the video is distributed via multicast.

The 2Fast approach [40] aims at improving the download throughput for Bittorrent file-sharing peer-to-peer systems in a collaborative way. A peer, called collector, can search for helper-peers which will download a file on-behalf of the collector. This will allow the collector to retrieve the file faster, as it can utilize the network resources of others. The assumption of the paper is that the downlink of collector has still spare capacity, as the download rate is assumed to be limited by Bittorrent's tit-for-tat mechanism to the collectors upload capacity. The 2Fast approach is the closest related work to our approach, as it implements a system where a node can delegate the retrieval of elements to other nodes that retrieve the elements on-behalf. However, this approach reuses the regular Bittorrent chunk system which is not suitable for peer-to-peer video streaming [41] and it also does not consider resource constrained situations where there is no or low idle throughput.

The above mentioned cooperative approaches claim to be peer-to-peer but in fact they are mainly not relying on peer-to-peer systems but exchange data directly between nodes (peers) instead of retrieving it from a server. The 2Fast approach is an exception to this.

However, the major assumption of these approaches is that wireless access networks are dimensioned in a way that the video can be retrieved at least by the peers via their access-link. Further, they assume a single Internet access-link to be capable of delivering the whole stream to one single user. These approaches will fail if the access-link is not capable of delivering the stream.

2.4.2 Web Browsing Related

This thesis envisions the application of the Cooperative Internet Access principle also to web browsing as described in Section 6. A natural way for web browsing to deal with a resource constrained environment is the deployment of so-called HTTP caches. These caches store elements of an web page to reuse them in the future by loading them from the local cache instead of retrieving them via the access-link. There is a substantial number of related work in the space of HTTP caching and cache [42, 43]. Those discuss how proxies can cooperate to provide a better cache result whereas the proxies are located within the provider network and not at the hosts (nodes), i. e., they do not consider the full approach taken in Section 6. We propose to deploy caches on the nodes and let nodes retrieve elements on behalf of other nodes, if the element is not cached in any local nodes. The caching can reuse existing techniques to organize cooperative web caches.

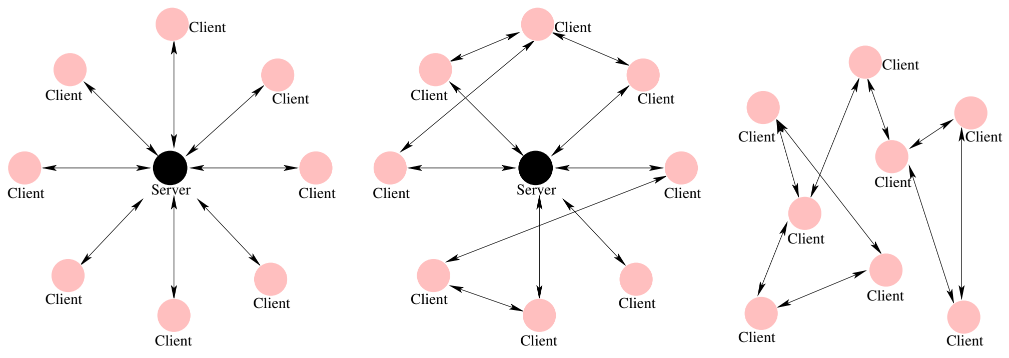
BuddyWeb[44, 45] proposes to place HTTP cache proxies on nodes being located in the same site, e. g., an university campus or in an enterprise, and to allow the nodes to query first the local caches before downloading an element from the Internet. The caches are communicating via a peer-to-peer protocol with each other. However, in BuddyWeb a cache miss cannot be handled on behalf of the requesting node, but the requesting node has to retrieve the element on its own, i. e., BuddyWeb nodes will also be limited to the capacity of their access-link.

Squirrel [46] is peer-to-peer web cache using a Pastry [47] DHT to locate the information elements in the participating nodes and implements a distributed web cache, but solely examines the DHT related aspects. Squirrel could be reused to locally organize the caches.

2.5 Peer-to-Peer Networking

The term *peer-to-peer* is probably known to most Internet users, either from news coverage about illegal peer-to-peer file sharing and because they use a peer-to-peer application. However, the term peer-to-peer covers a broad range of meanings and technologies. This section gives a first insight in this broad area. In peer-to-peer systems each participating node is client and server of the application at the same time, as compared to, for instance, file transfer with FTP (File Transfer Protocol) [RFC 959], where there is a server and a client. In peer-to-peer, each peer retrieves data from other peers and at the same time uploads data to other peers. Each peer brings a *resource* in the system where the *resource* can vary from pure forwarding or relaying capacity (e. g., call relaying in Skype [48]), data storage (e. g., file distribution with Bittorrent [36]), or even processing capabilities (usually known as grid computing).

A commonly used basic definition for the term peer-to-peer, according to [49] is:



(a) Client/Server (b) Hybrid Peer-to-Peer (c) Peer-to-Peer

Figure 2.4: Classification of Peer-to-Peer Systems

> [A peer-to-peer system is] *"a self-organizing system of equal, autonomous entities (peers)* [which] *aims for the shared usage of distributed resources in a networked environment avoiding central services."*

However, this definition is inaccurate in some points. In some peer-to-peer systems not all nodes are necessarily equal, e. g., as there powerful well-connected peers in peer-to-peer systems, such as the super-nodes in SopCast [50] (peer-to-peer video streaming) and while other nodes are even less powerful, e. g., mobile nodes. Some peer-to-peer systems are also not clearly distinguishable as pure peer-to-peer, as they involve a centralized server for some specific tasks, e. g., a login server [48], a server is improving the system's capacity, e. g., a super-node [50], or the peer-to-peer system acts as a support for a client/server infrastructure [51]. This leads to the differentiation between pure peer-to-peer systems and hybrid peer-to-peer systems, and client/server applications, as shown in Figure 2.4.

2.5.1 Peer-to-Peer Applications

This section gives an incomplete overview about peer-to-peer applications and thus not meant to be comprehensive nor complete, as there just too many peer-to-peer applications deployed.

The first peer-to-peer applications emerged in the filesharing space to exchange data on a large scale without the need to deploy the required server and network capacity to handled the exchange of this data. An early peer-to-peer filesharing application is Napster [52] while recently Guntella [53], Bittorrent [36], and eDonkey [54] are used by larger communities.

Another emerging area are peer-to-peer Massively Multiplayer Online Games (MMOG) [55, 56, 57]. Online games may require centralized server to exchange data between the participants and those server need to be able to handle the load. [58] lists a load of approximately 1 Mbit/s for 20 gaming users while popular games are said to have users in the range of several millions [59]. This results in high load situations at the gaming servers, where the research community is looking for solutions to lower the load by using peer-to-peer techniques.

Peer-to-peer video streaming has seen a massive proliferation in terms of commercialization in recent years, e. g., by PPLive [14], Zattoo [60], or SopCoast [61]. Peer-to-peer video streaming ranges from application-level multicast to mesh-based systems, from pure peer-to-peer to hybrid systems, and with the difficulty that most deployed systems are proprietary systems without publicly available documented system architecture and algorithms.

Nonetheless, peer-to-peer video streaming can be broadly categorized in tree-based approaches, e. g., ESM [62], the Application Level Multicast Infrastructure (ALMI) [63], the Host Multicast Tree Protocol (HMTP) [64], or SplitStream [65]; data-driven approaches that consider more the data flow instead of the overlay construction, e. g., CoolStreaming [66]; and mesh-based approaches, e. g., PULSE [67], GoalBit [68], or GridMedia [69].

Peer-to-Peer systems can be further classified according to how they organize their data information repository, i. e., how a request for data is mapped to the location (the peers) of the data, and how the actual data is distributed between the peers. The next sub-sections elaborate on these differences.

2.5.2 Unstructured and Structured Peer-to-Peer Systems

One of the differentiators in peer-to-peer systems is how the data to location mapping is organized or stored, i. e., how does a peer find what content (e. g., files, users) is available and where this content is actually located (i. e., the set of peers serving a particular content).

Unstructured peer-to-peer systems are either using a centralized server to perform this task, a hybrid system as in Figure 2.4(b), such as Napster [52], or they use a flooding (or gossipping) technique for this, as e. g., Gnutella [53]. Flooding refers to sending the look-up queries to all nodes in the system or a subset of them until the content is located. There are alternatives approaches to flooding such as random walk [70] to improve the scalability of search.

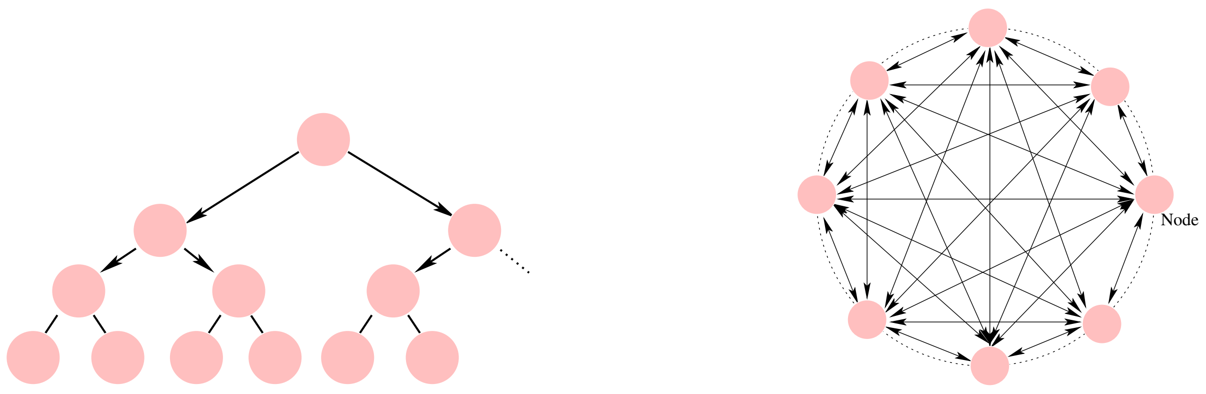
The challenge to develop scalable unstructured peer-to-peer systems [71] lead to the development of so-called structured peer-to-peer systems. Structured peer-to-peer systems are called Distributed Hash Tables (DHTs) and are basically distributed databases hosted on peers in the Internet, compared to regular databases being hosted on server machines. There are several approaches to DHTs, such as, e. .g, Chord [72], Kademila [73] or Pastry [47].

A mixed system used today (2010) for filesharing is Bittorrent [36], as some implementations, e. g., Azureus [74] uses a centralized server (tracker) and also a DHT perform the lookups.

2.5.3 Tree and Mesh Peer-to-Peer Systems

Peer-to-peer systems can follow different organization schemes for the data exchange between the peers belonging to the same overlay. There are tree-based or mesh-based systems, as shown in Figure 2.5.

Tree-based systems are inspired of multicast, where data is distributed in a tree from the data source to the data sinks. Each leave in the tree replicates the data and pushes the data further down the the trees. The tree structure is computed by a tree construction mechanism which pins down which node is put in which part of the tree as leave. Mesh-based systems are inspired by



(a) Tree-based (b) Mesh-based (full mesh)

Figure 2.5: Tree and Mesh-based Peer-to-Peer Systems

file swarming mechanisms, as implemented in Bittorrent [36] and do not follow a predetermined structure, but chose their neighbors on their own. This points also to an issue with tree-based systems, that such a system will fail to deliver the data down the tree, if the upload throughput of a leave node is below the required data transmission throughput [75].

It should be noted that some systems use both forms for peer-to-peer communication, i. e., they use a mesh to connect the participating nodes and to exchange control messages, but use a tree on top of this to disseminate the actual data to the participating peers: End System Multicast (ESM) [62] or Application Level Multicast Infrastructure (ALMI) [63]. On the opposite, there are also systems building a tree first, e. g., the Host Multicast Tree Protocol (HMTP) [64].

2.6 Introduction to Mesh Peer-to-Peer Video Streaming

Mesh-based peer-to-peer video streaming is a rather new class compared to the application-level multicast approaches. This section introduces these systems and their main properties. We focus our further discussions on 3 mesh-based peer-to-peer systems which are reasonable well documented: PULSE [67, 76], GoalBit [68], and GridMedia [69]. There are other more popular peer-to-peer video streaming systems, such as PPLive, but they are not well documented by the implementers but only through reverse engineering [77].

2.6.1 Peer Lookup

Mesh-based systems are usually unstructured peer-to-peer systems which do not rely on DHTs, but on gossiping or flooding or on centralized nodes to lookup peers. PULSE relies on a randomized gossip protocol (SCAMP [78]) for the peer lookup, while GoalBit and GridMedia rely on a centralized server. However, the type of peer lookup is not part of our later consideration for Cooperative Internet Access, as the peers simply have to able to participate in the peer lookup process. The peers will otherwise not be able to participate anyhow.

2.6.2 Chunks

Mesh based peer-to-peer video streaming systems, as well as peer-to-peer file sharing systems (e. g., Guntella [53], Bittorrent [36], and eDonkey [54]) work with a smallest atomic data unit. This unit is called *chunk* and can be uniquely referenced by a chunk identification number, called *chunk ID*, for a single resource of the peer-to-peer system, i. e., for a file or a TV channel, on a system global scale. However, there may be sub-chunk units, i.e., a chunk that is splitted in smaller parts, that are exchanged between particular peers, but which cannot be identified on a system global scale but only in a local scale between two peers. A chunk may contain one or several video frames and audio segments, depending on the used codec standard, codec rate, and codec settings.

2.6.3 Chunk Retrieval

The transmission of chunks from one peer to another can be either triggered by the receiver of the chunk or by the sender of the chunk. A system where the receiver triggers the transmission of a chunk is called a *pull* system, as the receiver pulls the chunk from the sender peer. A system where the sender triggers the transmission of a chunk is called a *push* system, as the sender pushes the chunk to the receiver peer.

In pull-based peer-to-peer systems, e. g., PULSE [76] or GoalBit [68], the peers exchange chunk maps that list the available (downloaded) chunks at that particular peer. These maps are often called *buffer maps*. The requesting peer uses the remote peer's buffer map and compares it with its local chunk buffer. The scheduling algorithm (Section 2.6.5) at the local peer decides what missing chunks shall be retrieved from the remote peer based on the chunk map comparison and additional information. The additional information can, for instance, include the measured throughput from the remote peer to the local peer, latency information, and behavioral information (see Section 2.6.6).

In push-based peer-to-peer systems, peers push a chunk to the remote peers. This push can be either a blind-push, i. e., without any knowledge about the remote peers chunk buffer, or an informed push where the peer pushes chunks in a smart way, as implemented in GridMedia [69]. GridMedia uses a push/pull combination: it uses first the pull mechanism if a peer has a new peer to exchange data with and changes to push after a transient phase). GridMedia uses chunk trains in the push mechanism: The receiving peer assigns a series of subsequent chunks, called train, to the sender peer and the sender peer pushes this train to the receiving peer without any further signaling. The receiving peers switches back to push if chunks get lost.

GridMedia combines the advantages of push and pull: it saves most of the signaling overhead caused by the exchange of buffer maps of the pull mechanism and also avoids the drawback of blind push systems. A blind-push may result in a receiving peer to get the same chunk pushed to it several times from several peers.

A blind-push system will not work with the Cooperative Internet Access approach, as the receiving peer cannot schedule which chunk is transmitted at what time. Cooperative Internet Access requires the receiving peers to actively influence the retrieval order of chunks, as only

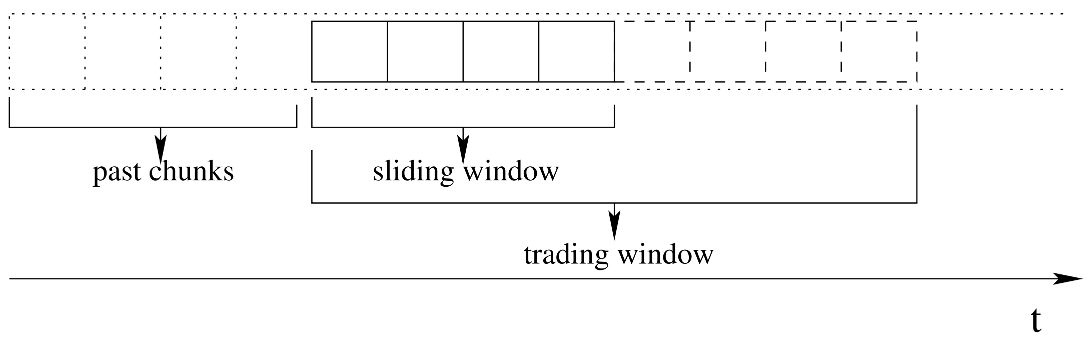


Figure 2.6: Chunk Buffer – General

the local participating nodes have the knowledge about the capacity of the access-links. However, a peer-to-peer system as GridMedia would still fit, as it allows the receiving peer (i. e., a local Cooperative Internet Access peer) to influence the set of pushed chunks. Only a blind-push system won't fit, as the local peers won't be able to influence which chunks are locally needed.

2.6.4 Chunk Buffer and Sliding Window

Peer-to-peer streaming systems use a sliding window mechanisms to manage their chunk buffer and thus also the video play out buffer towards the video player, independent of what approach is used. Figure 2.6 conceptually shows the chunk buffer. The buffer itself has a fixed size of n chunks (with $n = 12$ in the figure) and new chunks are added to the right while old chunks that are not needed anymore are dropped off at the left side. The *past chunks* are chunks that probably still in playout but are not needed for the chunk exchange with other peers anymore. The *sliding window* is used by the peer to determine the chunks needed from other chunks and to determine which chunks can be omitted in the retrieval process, if Forward Error Correction (FEC) is used (see below for FEC). The sliding window is the primary buffer where the peer works on, i. e., the peer has to retrieve the chunks in this window first to be able to playout the video later on. The sliding window, as well as the trading window is moved on, if the sliding window is completely filled with chunks. The *trading window* is range of chunks to be exchanged with the other peers.

Forward Error Correction (FEC) is used in many video streaming systems to compensate sporadic losses or corruption of packets or chunks without retrieving the missing elements again from the source. Especially in peer-to-peer video streaming systems, FEC is used to compensate missing chunks without disturbing the playout process. FEC allows to restore chunks from successful received chunks, if those chunks carry FEC information, i. e., the so-called erasure codes [79]. We assume a chunk-level FEC scheme, as defined in PULSE. This scheme allows to retrieve only K out of N chunks (N,K) but still be able to recover the remaining missing chunks. PULSE uses a loss rate of $1 - \frac{K}{N} = 25\%$, i. e., 25 % of the chunks in a trading window can be missed without degrading the video quality.

Figure 2.7 shows an example how the buffer is filled over time and the behavior of the sliding window. The example assumes the presence of Forward Error Correction (FEC) which allows to

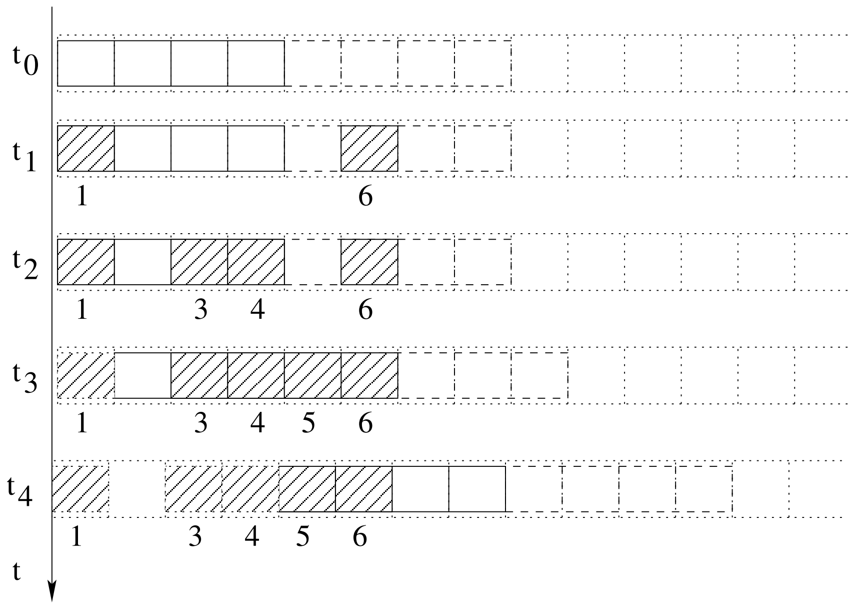


Figure 2.7: Sliding Window Chunk Buffer (Example)

deliberately skip certain chunks, if these can be restored by the FEC mechanism. The example assumes 3 chunks out of 4 as required minimum for the FEC to restore a missing element. At time t_0 in Figure 2.7, no chunk was loaded and the trading window starts at left most chunk position. At t_1, 2 chunks (1 and 6) were retrieved and at t_2, chunks 1, 3, 4, 6 are available in the buffer. The sliding window can now move on, given that 3 out of 4 chunks are required, i. e., the 4th missing can be recovered by the FEC. At t_3 the window moved on and chunk 5 arrived, so that the window can move on again. The window gets stuck at t_4, as the chunks from 7 are missing.

2.6.5 Chunk Scheduling

A main property of a peer-to-peer video streaming system is the used chunk scheduling approach. The scheduler determines in which order missing chunks are assigned for the retrieval from the peer's neighboring peers.

The first step is to determine the retrieval order of the missing chunks. PULSE ranks the chunks according to their availability amongst its neighboring peers, i. e., a rare chunk is preferred over not so rare chunks. It picks randomly a chunk in the case there are more chunks of the same "rareness". This ensures that rare chunks are distributed with a higher priority, as they are not well distributed in the system.

The documentation for GridMedia is unclear about the used scheduling approach, but from [69] it looks like that a sequential approach, i. e., starting with a low chunk ID towards higher chunk IDs, is used.

Goalbit divides the trading window in three ranges which influence the chunk scheduling behavior: urgent (close to playout point), usual (middle of trading window) and future (at the end).

Chunks are requested in a subsequential order in the urgent range and for the usual and future range, Goalbit uses a negative exponential function. This function prefers to fetch chunks from the usual range, but from time to time it also requests a chunk from the future range.

2.6.6 Peer Selection

Before a peer can exchange chunks with other peers, it has to find and select other peers which serve the same content and that are "able" to exchange chunks with the peer. "Able" refers to peers that have chunks that are needed by the requesting peer, the peers are willing to send chunks, and they have sufficient upload capacity to send chunks. Finding other peers is discussed in Section 2.6.1.

In PULSE, the peer selection algorithm is basically influenced by the behavior of the remote peers (if they responded to requests or if they upload a chunk or part of it) and the arrival of new peers (new peers may have more chunks already and thus become more interesting for the peer). This peer selection and evaluation algorithm is executed at a fixed interval, the so-called epoch. PULSE uses an epoch of 2 seconds.

Goalbit uses the tit-for-tat approach if Bittorrent [80]: It prefers generous peers sending a relative large amount of chunks to the receiving peers. These peers are *unchoked*, i. .e, they are served with chunks. Goalbit implements also the *optimistic unchoking*, i. e., from time to time unknown peers are selected and tested. A new peer replaces a peer if it serves better. Goalbit uses also an epoch to evaluate the usefulness of peers with 10 s as the epoch value.

The peer evaluation process is not clearly described for GridMedia, other than that half of the peers are chosen with "closer IP address and others chosen from the remaining nodes randomly" by the server [69].

2.7 Mobile Peer-to-Peer Video Streaming

Video streaming over mobile networks has been explored for a while, for instance, streaming over GPRS [81] and over WCDMA [82], but with the focus on client/server based streaming and the necessary parameters in these environment. An early work on peer-to-peer video streaming over UMTS cellular networks is reported in [83], with the focus on determining if it is possible in general. However, this paper solely considered a regular peer-to-peer system with mobile (i. e., UMTS nodes) and non-mobile nodes (i. e., fixed line) and if the mobile nodes are able to receive a sufficient amount of data to display the video. The key finding is that peer-to-peer video streaming is only possible in static environments, which in fact is also our finding in Section 3.

A range of publications discusses the application of peer-to-peer to mobile ad-hoc networks (MANETs), such as conceptual work on bringing peer-to-peer music streaming to MANETs [84], extending peer-to-peer video streaming to mobile ad-hoc by exchanging data between ad-hoc nodes [85], or design and simulation of a general peer-to-peer protocols [86] without specific considerations of file-sharing or video streaming.

There are other related works on peer-to-peer video streaming, such as, proposing a new peer-to-peer streaming system based on Real-time Transport Protocol (RTP) [RFC 3550] and RTP flow splitting [87] reusing existing protocols, e. g., Real-Time Streaming Protocol (RTSP) [RFC 2326]; or using a mobile peer-to-peer system to transcode videos in a collaborative way so that less powerful nodes or nodes with a small screen can receive the video [88].

The general observation is that mobile peers are assumed to have sufficient achievable throughput to receive or sent the video stream, or that there are only short periods of interruption [83]. For instance, this demo [89] of a peer-to-peer streaming between mobiles using WLAN which is for sure delivering sufficient throughput. Or the artificial low video bitrate in [87] of 112 kbit/s compared to average required download rate of 600 kbit/s to 900 kbit/s for youtube videos [4] with a fair quality and also the fact that most peer-to-peer video streaming systems run with video bit rates of 500 kbit/s to 600 kbit/s [5].

2.8 Conclusions of Related Work

The cooperative Internet access approach presented in this thesis is an information-centric approach which leverages application level striping of multiple access-links connected to multiple nodes to jointly overcome a resource constrained situation.It requires only changes in the local participating nodes but of no other node.

None of the related work solves the resulting challenges of resource constrained environment, as they either not considering this (e. g., 2Fast), or they require global modifications outside the local systems (e. g., MPTCP, MAR, PERM). We give an overview of the related work in relationship with cooperative Internet access in resource constrained environments in Table 2.1 The criteria are:

Applicable to Constrained Can the approach be used to mitigate a resource constrained environment for a single node or a set of nodes?

Multi-Host capable Can multiple hosts use the approach to mitigate a resource constrained environment?

Modification needed where Where are modifications needed to support this approach: locally in participating nodes only or globally with all possible communication partners.

Striping-level at which level of the network stack is the striping implemented?

It is interesting to see that several approaches suggest to use multiple access-links to mitigate the shortcomings of a single link, without discussing the monetary aspects related to this. Typically, it is required to have a subscription with a network operator to access the Internet. The subscription is valid for one device and typically for a single technology and single link; with exceptions in the area of UMTS and WLAN dual use for some operators in very limited spots. However, having several subscriptions to network operators at the same time would introduce significant costs for a user. Most users will have a single subscription, which is the base assumption of Cooperative Internet Access as we do not intend to incur additional costs for the

Category	Approach	Applicable to Constrained	Multi-Host capable	Modification needed where?	Link-Level	Network-Level	Transport-Level	Application-Level	Information Element Level
Protocols	IEEE 802.1 [19]			global	✓				
	PPP [RFC 1990]			global	✓				
	"IP-in-IP" [20]			global		✓			
	MPTCP [21]	✓		global			✓		
	pTCP [22]	✓		global			✓		
	Psockets [23]	✓		global				✓	
General Systems	MAR [30]	✓		global			✓		
	EMS [31]			global			✓	✓	
	PERM [32]	✓	✓	local			✓		
	CUBS [33]	✓	✓	local			✓		
	MOPED [34]	✓	✓	global		✓			
P2P Systems	MOVi [38]			global			✓		
	PatchPeer [39]			local			✓		
	2Fast [40]		✓	local					✓
Web Systems	BuddyWeb [45]			local					✓
	Squirrel [46]			local					✓
	Cooperative Internet Access	✓	✓	local					✓

Table 2.1: Comparison of Approaches

users. This calls for the proposed system multihoming, where the local Cooperative Internet Access system is actually multihomed but not necessarily the nodes.

The approaches to enable nodes to reuse other users' access-links (e. g., PERM or CUBS) consider how to schedule flows to the access-links. Cooperative Internet Access does not consider flow scheduling, as the achievable throughput on an access-link may vary a lot, compared to what PERM and CUBS expect: fixed-line networks will keep their maximum bitrate, unless there is a severe fault. Mobile wireless will even change the maximum bitrate and thus heavily impact the achievable throughput. CUBS and PERM would assign the flow to an access-link and will not be able to react to changes afterwards anymore, other than stopping the transmission and restarting new on another link. Cooperative Internet Access avoids this issue by scheduling on the level of information elements, as flows are too inaccurate, as the throughput of a link changes to fast. The level of information elements also allows to split the schedule the retrieval of elements to multiple access-links at various times without dealing with transport level issues in the scheduler, or in the first place at all. The transport level is offering multiple ways of retrieving elements but it should not mandate how elements are retrieved, i. e., the transport is a vehicle – nothing more. For instance, Cooperative Internet Access does not depend on whether the transport is TCP, MPTCP, or MIXTP (of MOPED), but can reuse whatever is possible.

3 Understanding the Environment

The access to the Internet is possible through a plethora of different access technologies and also different bitrate ratings within a particular technology. But most of the people using the Internet today (2010) are connected to the Internet via fixed-line access, such as, for instance, Asymmetric Digital Subscriber Line (ADSL), Cable-TV (CATV), and Fiber To The Home (FTTH) for the simple reasons that access is at low cost, it is stable (i. e., in terms of provided bitrate over time and availability of the fixed line), and the achievable bitrates are perceived as fast enough to access most of the Internet services. For mobile users there are wireless access technologies that provided Internet access, such as UMTS-based cellular networks, Worldwide Interoperability for Microwave Access (WiMAX), or Wireless LAN (WLAN). However, the only deployed technology for mobile Internet access at a large scale (almost anywhere in a country) is UMTS based pr CDMA2000 based.

On the other hand, networking applications and systems are typically developed with some implicit assumptions about the network and the operational conditions of the network. The application and system designers assume that the network will deliver the throughput needed for the application at all times, or only with few times where these conditions are not met. For instance, a typical HTTP-based application, such as web mail access, assumes that the web page can be retrieved from the HTTP-server within a reasonable time (e. g., within less of a minute); peer-to-peer live streaming applications assume that every peer has sufficient download and upload capacity to participate in the data exchange for the video.

The applications' requirements to network is growing: Years ago, most web pages were mainly made up by text with a few pictures, compared to today where most web pages have several pictures and also videos, causing a much higher throughput demand of the applications. Considering today's wireless access technologies and the trend of using the Internet more and more will being on the move, instead of connecting at a single point to it, it becomes obvious that not all applications will work easily via wireless networks.

The intention of this chapter is to explore the space of wireless Internet access and its properties as seen by the applications. We focus on a widely deployed cellular data network technology and give measurement results on what network throughput can be achieved and the likelihood to get this through by applications using this technology.

3.1 Overview of Mobile Selected Wireless Technologies

This section gives a brief overview about mobile wireless technologies used by today to provide users with Internet access without having the need to be connected to a fixed line. The major technology used in Europe to provide mobile Internet are based on GPRS, EDGE and UMTS technologies.

Cellular data communication systems are used across the globe to to provide mobile Internet to users in large geographical areas, i. e., for a whole country. In Europe, GSM-based networks offer GSM (for voice and limited data services), General Packet Radio Service (GPRS) (for data services), and the evolution of GPRS with an increased bitrate, called Enhanced Data rates for GSM Evolution (EDGE). GPRS and EDGE are coined as 2.5 Generation, i. e. they are a transition step from 2G (GSM) to UMTS. UMTS offers better data networking support and much higher bitrates.

3.1.1 Cellular Data Communication Systems

Cellular data networks are typically built as shown in Figure 3.1, independent of their very specific details, such as the employed wireless transmission or transport protocols. They consist of building blocks, such as the mobile node, called User Equipment (UE), the Base Stations (here BS1 to BS3), Radio Controllers (here RC1 and RC2), a switching center and an IP-Gateway-Node, if they interconnect to an IP-based network. The User Equipment, better known as mobile terminal, connects via the air-interface to the base stations. The base stations can pass all data received via the air interface to a radio controller. The base stations and the radion controller are connected via the mobile backhaul. The actual services of the cellular network are executed in the mobile core network, where the switching center is one center piece. The connection to the Internet is performed via an IP-Gateway-Node, where the cellular network is terminated and all traffic is handed over to the Internet or is received from the Internet. The figure as shown here is a simplification, as more entities are probably needed to operate such a cellular network, and as there more base stations, switching centers, etc, deployed all over the country.

For our particular context, it is important to understand how the UE is attached to the cellular network and what resource can be used. The wireless part of the cellular network is made up of network cells, where each Base Station is creating one or more of these cells. Each cell has multiple adjacent cells. An UE attaches via the air interface to Base Station serving the cell. The cellular network takes care if the UE moves, i.e., if the UE leaves the coverage area of one cell, the network takes care that the UE is performing a handover to the next cell.

3.1.2 GPRS, EDGE, and UMTS

This section introduces the basics of GPRS, EDGE, and UMTS for the further discussion on what Internet application can expect from a cellular network in terms of achievable throughput, network availability, and coverage under every situation (parameters necessary for us). The technological differences between GPRS, EDGE and UMTS are large, in terms of used air

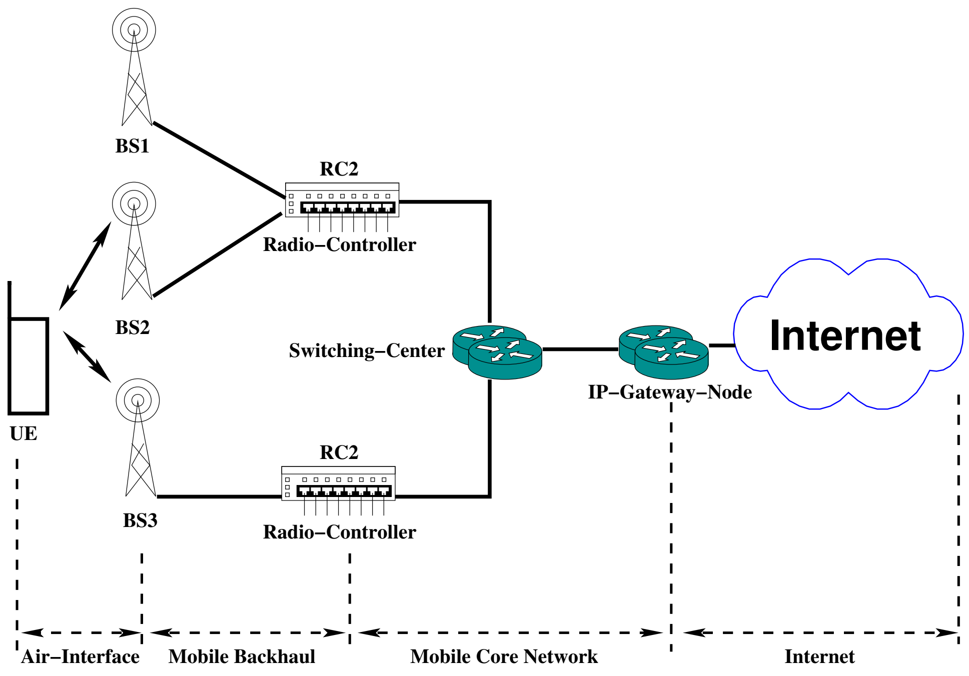


Figure 3.1: Schematic design of cellular networks

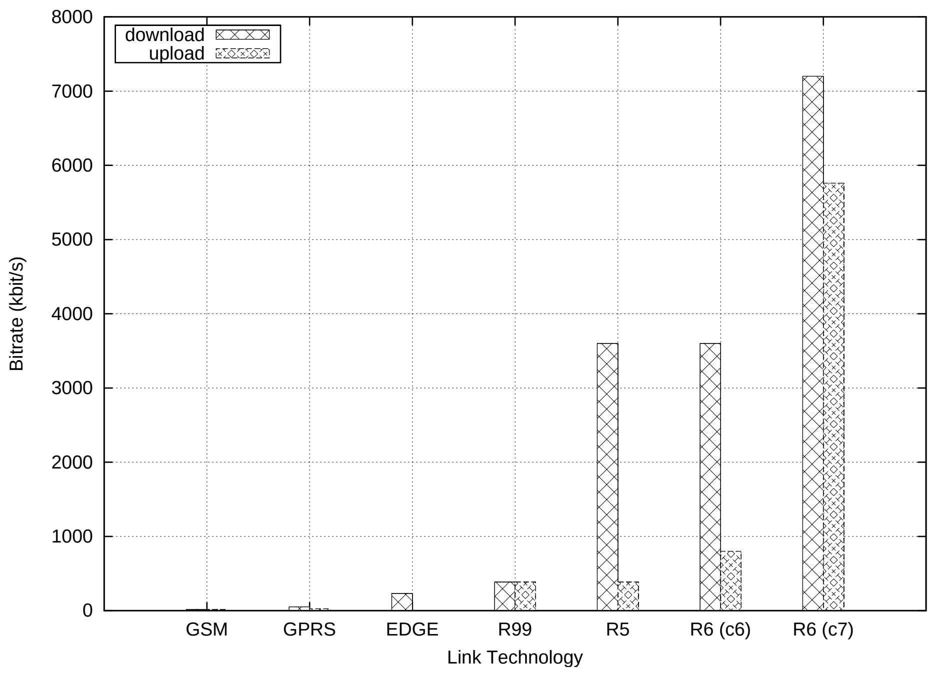


Figure 3.2: Comparison of mobile wireless maximum bitrates

interface, signalling, network architecture, etc. For instance, Wideband Code Division Multiple Access (WCDMA) is used for the UMTS air interface [1], while GPRS/EDGE uses Time Domain Division Multiple Access (TDMA) [90].

Figure 3.2 gives a comparison of the maximum bitrates per user device of the various technologies used in cellular networks in Europe. The x-axis denotes the different technologies

Technology	Downlink Bitrate	Uplink Bitrate	Reference
GSM	9.6 kbit/s	9.6 kbit/s	[1], p. 90
	14.4 kbit/s	14.4 kbit/s	[1], p. 90
GPRS	60 kbit/s	40.0 kbit/s	[1], p. 95 (TDMA 3+2)
EDGE	230 kbit/s	60 kbit/s	[1], p. 90 (TDMA 4+1)
UMTS Release 99	384 kbit/s	128 kbit/s	[1], p. 151
UMTS Release 5	3600 kbit/s	384 kbit/s	[1], p. 160
UMTS Release 6	3600 kbit/s	800 kbit/s	UE cat.6 [1], p. 240, [TS 25.306b]
	7200 kbit/s	(T) 5760 kbit/s	UE cat.7, [1], p. 240,[TS 25.306b]

Table 3.1: Maximum bitrate of GSM, GPRS, EDGE, and UMTS (T: marks a theoretical maximum bitrate, UE cat.: User Equipment category, see [TS 25.306b])

and the y-axis denotes the maximum bitrate. For UMTS, only the release numbers are listed in Figure 3.2, i. e., Release 99 (R99), Release 5 (R5), Release 6 with UE category 6 (R6 (c6)), and Release 6 with UE category 7 (R6 (c7)). The UE category [TS 25.306a] specifies, amongst other properties, which maximum bitrate the device can achieve on the radio interface. The download rate (i. e., from the network to the User Equipment) and the upload rate (i. e. from the User Equipment to the network) are shown for each technology. The maximum bitrate on the air interface is increasing with every technology, but there is still a gap between download rate and the upload rate, like in other access technologies (e. g. ADSL). Only for UMTS Release 6 with device category 7 (R6 (c7)) the theoretical upload bitrate reaches the same dimension as the download bitrate. However, this is the theoretical upper bound, which will turn to a much lower upload bitrate when used in real deployments. The author could not find the real upload value, neither in literature nor by measurements, as he does not have access to a category 7 UE.

The maximum bitrates for down- and upload are listed in Table 3.1.2. These values represent the bitrate for an ideal transmission case between User Equipment and Base Station. The ideal case is, for instance, when the User Equipment is not moving and there is no obstacle between both (i. e., there are no building, trees, walls, etc). There may be other factors that impact the maximum bitrate, depending on the used technology, for example, distance between User Equipment and Base Station in UMTS. User Equipment that is far away from the Base Station, e. g., situated at the cell edge, will only be able to achieve a lower bitrate, as compared to the ones stated in Table 3.1.2.

However, the actual achievable bitrate of an User Equipment in the cellular network in a particular environment depends on a number of factors, such as:

- wireless condition (e. g., whether in-house, static, moving, etc);
- number of concurrent users per cell;
- used UE (e. g., whether the terminal is UMTS enabled, which category, etc);
- capacity of the mobile backhaul link between base station and radio controller;
- capacity of the mobile core network;

- capacity of the IP-Gateway-Node;

Additionally for the applications there are more factors than can limit the the achievable throughput, such as, the capacity of the whole Internet path from the IP-Gateway-Node to the other end (e. g., a web server); and the capacity of the other end.

The following sections briefly discuss the impact of each factor to the achievable throughput.

3.1.2.1 Wireless Conditions

Wireless data transmission is always subject to interference on the air interface due to a number of issues. An interference causes a degradation of the achievable bitrate or may even cause a complete black-out of the transmission. While some interferences are cause by radio specific issues, for instance, by a jamming transmitter of other radio sources, some are simply caused by a movement of the User Equipment or obstacles between the communicating parties.

Another limiting factor in WCDMA is the distance between the UE and the Base Stations. The UE has a maximum transmission power (0.250 Watt in Europe for UMTS power class 3, see p. 167 in [1]) and the further the UE is away from the Base Station, the more of the wireless channel's transmission capacity is used for error correction and is not available for the transmission of user data.

We use two illustrative examples to show the impact of obstacles and moving through a radio cell:

> An User Equipment is located on a line of sight to the base station, with no obstacle in between, and reaching the maximum bitrate. If the user that is carrying the UE is now moving for few centimeters behind the corner of a building, the whole situation may have changed dramatically. The location behind the building places one or more obstacles between the UE and the Base Station, possibly causing a complete drop or decrease in the achievable bitrate.

> The user with its UE is moving in a car or train at speeds around 100 km/h (compared to pedestrians which move at 5 km/h or less). In this case, the UE will pass very fast through a single radio cell and hand-over to the next one. Therefore, the UE moves through a cell, where the achievable bitrate depends also on the distance to the Base Station (amongst other parameters), already causing a change for the application's throughput. Second, it is very likely that the car will pass road cuts, buildings, etc.

3.1.2.2 Number of Concurrent Users per Cell

UMTS cellular networks use Wideband Code Division Multiple Access (WCDMA) for the air interface. In WCDMA, all Base Station use the same radio frequency and thus all User Equipment are sending and receiving data via the same frequency. In GSM (also GPRS and EDGE),

Spreading Factor (Downlink)	Gross Bitrate	Usable Bitrate
8	960	384
16	480	128
32	240	64
64	120	32
128	60	8
256	30	5.15
512	15	1.7

Table 3.2: Relationship between UMTS spreading factors and achievable bitrate ([1], p 172)

each cell uses a different radio frequency and Time Division Multiple Access (TDMA) for the actual data transmission. A specific transmission channel is created by using code spreading [1], i. e., where each bit to transmit is actually spread over a large radio spectrum. The cellular network assigns a spreading factor to each UE, thus creating the transmission channel between both of them. For each spreading factor, a maximum bitrate is pre-set. This pre-set bitrate depends on a number of factors, with are left for the interested reader for further reading [1]. The operator of the cellular network can actually choose how much bitrate a UE is assigned by using a different spreading factor, as shown in Table 3.1.2.2. A spreading factor of 8 refers to 8 UE's being able to run each at a maximum bitrate of 384 kbit/s.

However, the operator is free to assign other spreading factors, resulting in lower or higher throughput per UE, e. g., when many people are in one particular cell, the operator could assign a spreading factor of 32, resulting in a maximum bitrate of 64 kbit/s.

3.1.2.3 User Equipment Issues

Today's mobile cellular terminals, called here User Equipment, (i. e., mobile handsets, data cards for laptops, etc) are capable of GSM, GPRS, EDGE, and UMTS. They can seamlessly switch from one technology while keeping the data transmission channel up and running. This involves also support from the cellular network. This is of key benefit, as not all technologies are deployed in all locations. Typically, UMTS is deployed only in densly populated areas, where the country side is usually equipped with less costly GPRS or EDGE for data transmission.

Nonetheless, there are also different device categories within a particular technology. For instance, within the UMTS standard there are different releases of the cellular network technology, e. g., Release 99 being the first UMTS standard, and each release may add another new category of devices. A category of a device describes mainly the wireless capabilities of this device, i. e., which radio technology is used, the coding scheme, and also the maximum bitrate for down- and uplink. For UMTS Release 7 the categories are described in [TS 25.306a]. For instance, an UMTS UE Release 6 category 6 is able achieve a maximum bitrate of 3.6 Mbit/s, while an UMTS UE Release 6 category 7 is able achieve a maximum bitrate of 7.2 Mbit/s.

There are also other issues related to the UE, such as the connection between the UE and the computer using the UE to connect to the Internet. The link between can be either via USB or Bluetooth. This link can be also an issue, for instance, when using an UMTS UE Release 6 category 7 that is connected via Bluetooth EDR+ running with maximum 2 Mbit/s vs. 7.2 Mbit/s of UMTS).

3.1.2.4 Mobile Backhaul Link Capacity

Each base station is connected via a mobile backhaul link to its serving radio controller, as depicted in Figure 3.1. This backhaul link can be wired or wireless, dedicated for a single base station or shared with multiple Base Stations (i. e., aggregating traffic from several Base Stations).

The capacity of this backhaul link can be dimensioned to fit all voice or data communication coming or going to a Base Station (see p. 123 in [1]). However, there may be also deployments, where the backhaul link is dimensioned with a smaller capacity than the aggregated capacity of all connected Base Stations, i. e., overbooking this link. This is done for cost reasons, as low capacity backhaul can be cheaper, and where not the aggregated Base Station capacities are needed all times.

3.1.2.5 Mobile Core Network Capacity

The mobile core network should be actually dimensioned such that it is able to carry all traffic, even at peak times. However, it is up to the network operator to dimension the network, i. e., link speed or processing power of the network nodes, and some operators may deploy a core network that is not able to carry the full load of all user data traffic.

From the UE's perspective it is safe to assume that this network is dimensioned with sufficient capacity.

3.1.2.6 Capacity of IP-Gateway-Node

The first hop with IP capabilities visible to the user traffic is the IP-Gateway-Node. This node terminates the cellular network specific protocols and connects the network to the public Internet, or to an enterprise intranet.

This IP-Gateway-Node typically implements a Network Address Translator (NAT) and also a number of other, cellular network related states. All of these require processing power and memory.

From the UE's perspective it is safe to assume that this IP-Gateway-Node is dimensioned with sufficient capacity.

3.2 Mobile Wireless Market in Germany

The previous sections discussed wireless technologies that connect users to the Internet. This section here gives an overview about the operator market and their cellular network deployments in Germany. Germany as place was taken, as the author lives there and has therefore access to the mobile wireless networks. Most of the said within this section may be valid also for other countries in Europe and also world-wide.

In each country, there are typically a number of operators that own and operate their network infrastructure. These are called Mobile Network Operators (MNO). Furthermore, there are so-called Mobile Virtual Network Operators (MVNO) that do not operate their own network infrastructure. A MVNO operates its virtual network on the network of a MNO, i. e., a MVNO sells solely the service to its customer under its brand name.

In Germany there are four MNOs and an uncounted number of MNVOs. We limit our further considerations to the MNOs, as their networks are the limiting factor in terms of what can be achieved with today's cellular network deployments. The MNOs are the real operators with their own network infrastructure.

The MNOs are T-Mobile, Vodafone, EPlus, and O2. The coverage of each of the operator seems to be more less the same, but the offered cellular technology varies between them. The bigger operators, T-Mobile and Vodafone, have way more HSPA deployments than the smaller operators.

There is at least one WiMAX operator in Germany (as of today (2010)), Deutsche Breitband Dienste. They do not offer WiMax as a mobile wireless access service, but more for stationary service, as a xDSL replacement.

3.3 Measurement Methodology

There are various ways to assess the performance characteristics of a wireless technology, such as, for instance, observable Bit Error Rate (BER) and achievable bitrate in a cell for static or slowly moving mobile devices. The observations can either be achieved by simulations of the radio technology or system wide evaluation (including also the backhaul), and also by field measurements once the mobile wireless technology is deployed. However, these observations and the following assessment of the performance do provide a limited view of how the particular technology will perform in an operational deployment where a user or an application is using the Internet access. The typical assessment of wireless technologies is typically not very helpful in terms of what throughput can be achieved by an application while the user is traveling by car or train, as this not of interest when evaluating a specific radio technology.

For our field of interest, we consider deployments of wireless technologies that usually will experience many more challenges, for example, impact of the surrounding (e. g., hilly areas, trees) and whether the terminal is moving (e. g., static, high-speed train travelling at speeds greater than 150 km/h) and in what context it is moving. The context is important to be considered, as for instance, a building can cause a different attenuation to the radio signal as compared

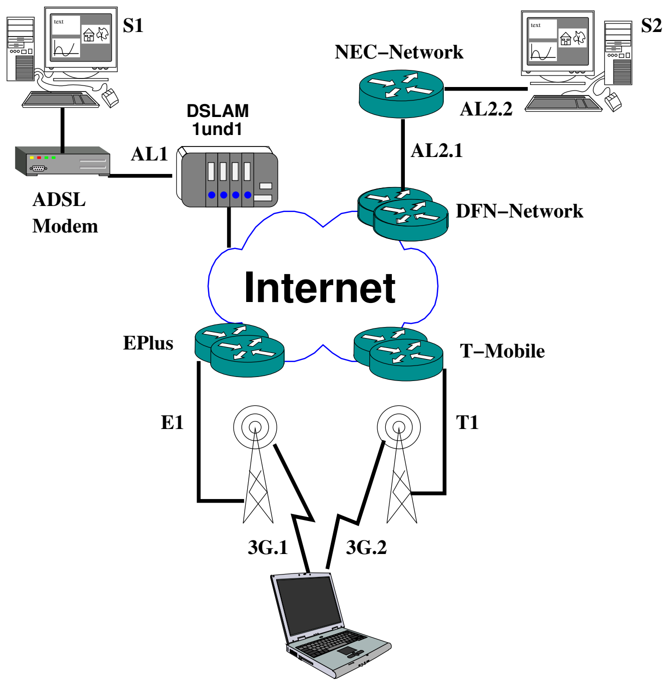


Figure 3.3: The measurement setup

to driving in a car or driving in a train. For instance, an ICE 3 high-speed train [91] running in Germany is made of a steel tube and metallic coated windows which cause an attenuation to the radio signal. On the other hand, there are deployment issues that are not enforced by a particular technology, but are enforced by the network operator, e. g., the decision whether GPRS, UMTS or HSDPA is deployed, is solely made by the operator and is the primary factor to level of performance to be expected.

Nonetheless, there are too many factors that can impact what performance can be expected at the mobile terminal, even though a the usage of a particular technology can hint a the expected performance level. We see the need that it is much more important for the user of a mobile terminal (or for an application on that device and the designer) to determine what the achievable throughput on the TCP/IP level is and how often this can be or cannot be achieved. There are multiple ways to deduct what can be expected by a mobile terminal in terms of the above define performance, such as, simulations based on theoretical studies of mobile wireless technologies and measurements of real deployments. We have decided to use measurements of real network deployments, as we want to find out what a mobile terminal is getting in terms of performance and also related to our working topic if the usage of multiple operaters concurrently is beneficial at all, e. g. do multiple operators enhance the resilience of the system.

> The goal of the measurements is to obtain the achievable throughput, uptimes of the wireless link and the downtime of the wireless link, on a per cellular network operator base, as seen by the application.

The only way to gain the above insight is to do active measurements of multiple mobile wireless operators when moving.

Item	Value	Description
AL1	16Mbit/s	ADSL line downlink maximum bitrate
AL1	1Mbit/s	ADSL line uplink maximum bitrate
AL2.1	100Mbit/s	LAN maximum bitrate
AL2.2	34.368 Mbit/s	E3 line between NEC Heidelberg and DFN POP Heidelberg
E1	unknown	cell tower backhaul EPlus
T1	unknown	cell tower backhaul T-Mobile
3G.1	UMTS/EGDE-capable	Sony Ericsson K610i mobile phone
3G.2	HSDPA/UMTS/EGDE-capable	Sony Ericsson c702 mobile phone (cat. 6 [TS 25.306a])

Table 3.3: Measurement related network parameters

The measurement setup is depicted in Figure 3.3. We used two operator networks to perform our measurements, as the author had access to two SIM cards only. The operators were T-Mobile and EPlus, both are offering UMTS-based Internet access.

Figure 3.3 shows on the top the location and network connectivity of the measurement servers *S1* and *S2*. The mobile device with the measurement clients is depicted on the bottom of the figure and is connected to 2 mobile networks via 2 wireless interfaces *3G.1* and *3G.2*. The mobile networks are depicted in a simplified way by the cell towers, the backhaul links (*E1* and *T1*), and a multi-router icon connected to the public Internet. Each provider (in this particular case *EPlus* and *T-Mobile*) do operate their infrastructure.

The measurement server *S1* is connected via an ADSL modem router to an ADSL access line *AL1* that in turn is connected to a DSLAM and (not shown) to a BRAS to the public Internet. The measurement server *S2* is connected via a LAN *AL2.2* to the a router that is connected to the German Research Network (DFN-Network) via *AL2.1*. The DFN-Network connects to the public Internet. This configuration ensures that none of the access lines from the Internet to measurement server (*AL1* or *AL2.1* and *AL2.2*) are the bottleneck. The maximum measured throughput for EPlus was below the maximum line bitrate *AL1* (384 kbit/s vs. 1 Mbit/s) and the maximum measured throughput for T-Mobile was below the maximum of the minimum of the access lines bitrate of *AL2.1* and *AL2.2* (480 kbit/s vs 34.368 MBit/s), see also Table 3.3.

The measurement servers run the server side measurement software, i. e., *S-MP1* at *S1* and *S-MP1* at *S2*. The client side measurement software runs with two instances *C-MP1* and *C-MP2* on the mobile device, as show in Figure 3.4. Each client instance connects via TCP to one server instance: *C-MP1* to *S-MP1* and *C-MP2* to *S-MP2* and starts downloading bulk data to the client. The client instances log the measured TCP level throughput at a fixed interval. The interval is stated together with the measurement results. The mobile phones are connected via USB to the mobile client and each phone is assigned an interface on that client. For each mobile phone (interface) an instance of the client side measurement software is started. The routing is pre- configured so that each TCP measurement flow is directed over another interface, as shown in Figure 3.4.

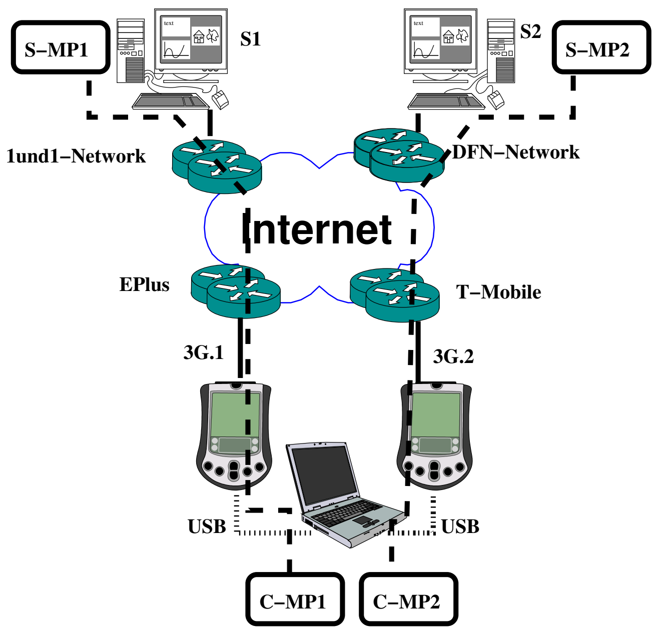


Figure 3.4: Measurement Flows (schematic)

```
traceroute to 109.250.139.167 (109.250.139.167), 64 hops max, 40 byte packets
 1  fritz.box (192.168.178.1)  1.385 ms  0.406 ms  0.506 ms
 2  bras2.fra.qsc.de (213.148.133.3)  27.879 ms  38.008 ms  29.270 ms
 3  core1.fra.qsc.de (87.234.13.21)  21.419 ms  21.626 ms  23.605 ms
 4  ffm-b7-link.telia.net (213.248.72.33)  21.892 ms  43.777 ms  21.015 ms
 5  ge-6-1-3.BR1.FFT1.alter.net (146.188.112.41)  21.687 ms  22.518 ms  21.599 ms
 6  ge-1-2-0.XT2.FFT1.ALTER.NET (146.188.6.77)  21.682 ms  22.616 ms  21.368 ms
 7  so-0-0-0.XT1.HDN2.ALTER.NET (146.188.14.194)  29.579 ms  29.482 ms 30.296 ms
 8  POS6-0.GW4.HDN1.ALTER.NET (149.227.17.158)  29.803 ms 30.929 ms  30.059 ms
 9  E-Plus.Hilden.de.ALTER.NET (139.4.54.122)  32.858 ms  31.445 ms  32.573 ms
10  * * *
11  * * *
```

Figure 3.5: S-MP1 to C-MP1 (EPlus)

The configuration in Figure 3.4 ensures that the TCP flows are using disjoint Internet paths, so that at least these TCP flows are not impacting each other. The results below show the "reverse" traceroute output from *S-MP1* to *C-MP1* (EPlus) in Figure 3.5 and from *S-MP2* to *C-MP2* (T-Mobile) in Figure 3.6 to show that the paths are disjoint (under the assumption that the forward and reverse path are the same). We use traceroute in the reverse direction, because both used operators, T-Mobile and EPlus, blocked outbound ICMP packets at the time of the measurements (June 2009). Therefore, we connect via SSH to the measurement server and run traceroute towards the originating IP address of the SSH-session. This originating IP address is the external IP address at the operator NAT.

The measurement configuration ensures that the 2 TCP flows are not interfering with each other, but there is still the chance that both Internet paths are experiencing congestion or that the measurement flows are at least competing with other IP flows originated from somewhere else in the Internet or in the provider networks. This cannot ruled out, since we used the public Internet, but the measurement results suggest, that most of the throughput is impacted by the wireless part. TCP was used for two deciding factors: (i) TCP is the most used transport protocol, also for P2P video streaming, and (ii) the UMTS network deployments are typically deploying Network Address Translator (NATs) at the boundaries between the UMTS IP network

```
traceroute to 80.187.104.124 (80.187.104.124), 64 hops max, 40 byte packets
 1  195.37.70.126 (195.37.70.126)  0.877 ms  0.784 ms  0.729 ms
 2  xr-hei1-ge9-4.x-win.dfn.de (188.1.233.197)  1.605 ms  2.818 ms  1.980 ms
 3  xr-gsi1-te2-1.x-win.dfn.de (188.1.145.89)  3.232 ms  2.952 ms  3.982 ms
 4  zr-fra1-te0-0-0-6.x-win.dfn.de (188.1.145.38)  5.230 ms  4.955 ms  4.054 ms
 5  zr-erl1-te0-7-0-2.x-win.dfn.de (188.1.145.142)  14.632 ms  9.622 ms  8.055 ms
 6  80.156.160.141 (80.156.160.141)  14.139 ms  13.748 ms  13.933 ms
 7  n-ea5-i.N.DE.NET.DTAG.DE (62.154.52.70)  18.377 ms
    n-ea5-i.N.DE.NET.DTAG.DE (62.154.52.74)  17.366 ms
    n-ea5-i.N.DE.NET.DTAG.DE (62.154.52.70)  18.246 ms
 8  194.25.208.210 (194.25.208.210)  20.184 ms  22.447 ms  21.219 ms
 9  * * *
10  * * *
```

Figure 3.6: S-MP2 to C-MP2 (T-Mobile)

and the public Internet (see 3.1.2.6). The usage of NATs limits the use of UDP, which is the main reason for most applications to simply rely on TCP.

The TCP implementation used was the standard TCP stack delivered with MacOS 10.4, which is TCP Reno [RFC 2581] with TCP Selective Acknowledgment (SACK) [RFC 2018] enabled.

3.4 Measurements

This section presents the results obtained from a set of measurement runs performed while traveling on an ICE 3 high-speed trains [91] between the stations Mannheim Hauptbahnhof and Frankfurt Flughafen Fernbahnhof, in Germany. The author travelled multiple times (7 times in June 2009) with the measurement setup described in Section 3.3 along the track. The distance between both stations is 76 km (according to the web site of Deutsche Bahn [91], under mobility-check) and the journey takes regularly 30 minutes.

The measurement was performed for almost all of the travel time, expect when boarding the train and setting up the equipment. The measurement client logged the measured throughput every 5 seconds. In the case that the client did not receive data within 5 seconds, it logged 0 kbit/s as result. At times when the client received data, it noted the measured throughput averaged over 5 seconds.

The measurement results are discussed first in Section 3.4.1 and the processed measurement results are shown in Section 3.4.2.

3.4.1 Measurement Results

This section discusses representative examples of a few measurement runs in a train and also in more stationary scenarios with two operators in Germany. The measurement runs are discussed to show the volatile character of the wireless link.

Figure 3.7 shows the achievable throughput while being in a UMTS coverage area within a city (Leimen, Germany). The mobile terminals are not moving and are located within a house. The throughput is varying in narrow confines for T-Mobile, but for Eplus, the throughput is already less steady.

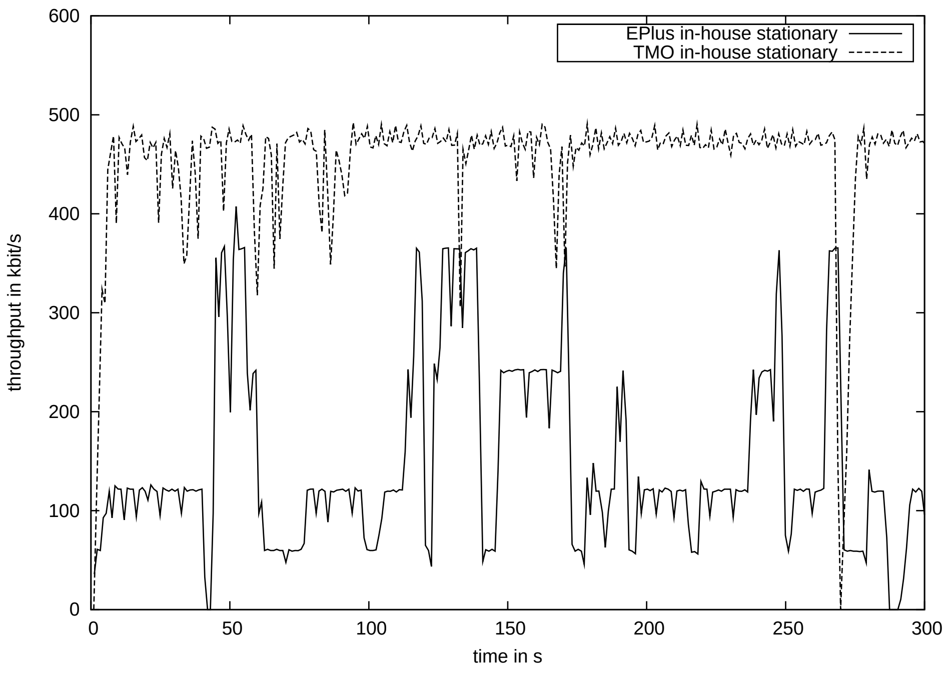


Figure 3.7: EDGE/UMTS measured throughput - stationary in a building

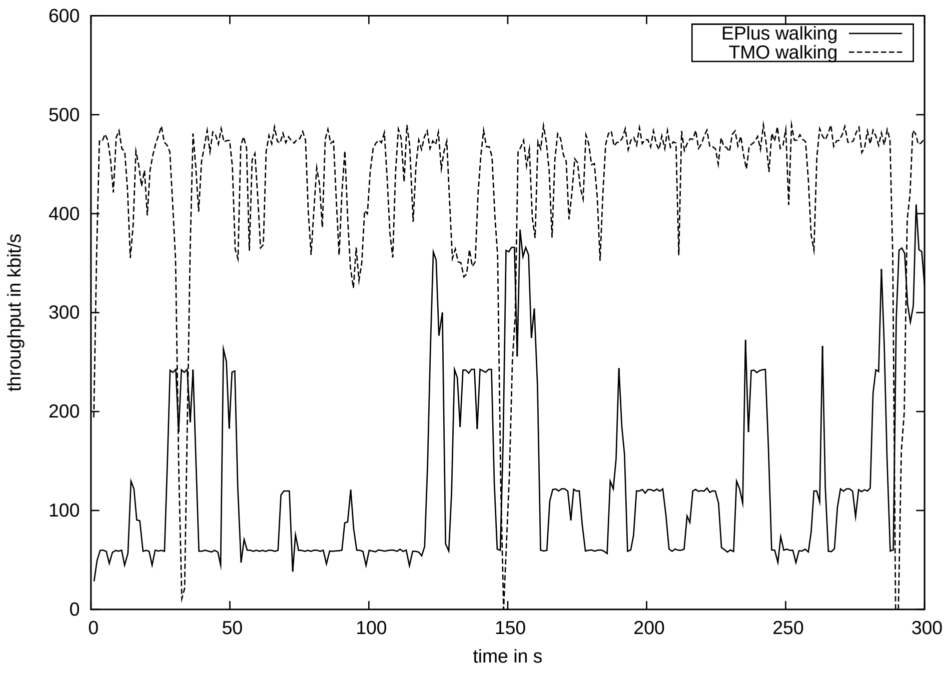


Figure 3.8: EDGE/UMTS measured throughput - while walking through a city

Figure 3.8 shows the the achievable throughput while talking a walk in the same city of Leimen, Germany. The city of Leimen has in average two story buildings and only very few taller buildings. The throughput for T-Mobile is mostly still in the range of 400 kbit/s to 500 kbit/s, even while moving. The throughput measured for Eplus shows a lower level, mostly in the range of 50 kbit/s to 100 kbit/s, with a few peaks where the throughput reaches the maximum UMTS R99 throughput of 384 kbit/s (see also Table 3.1.2).

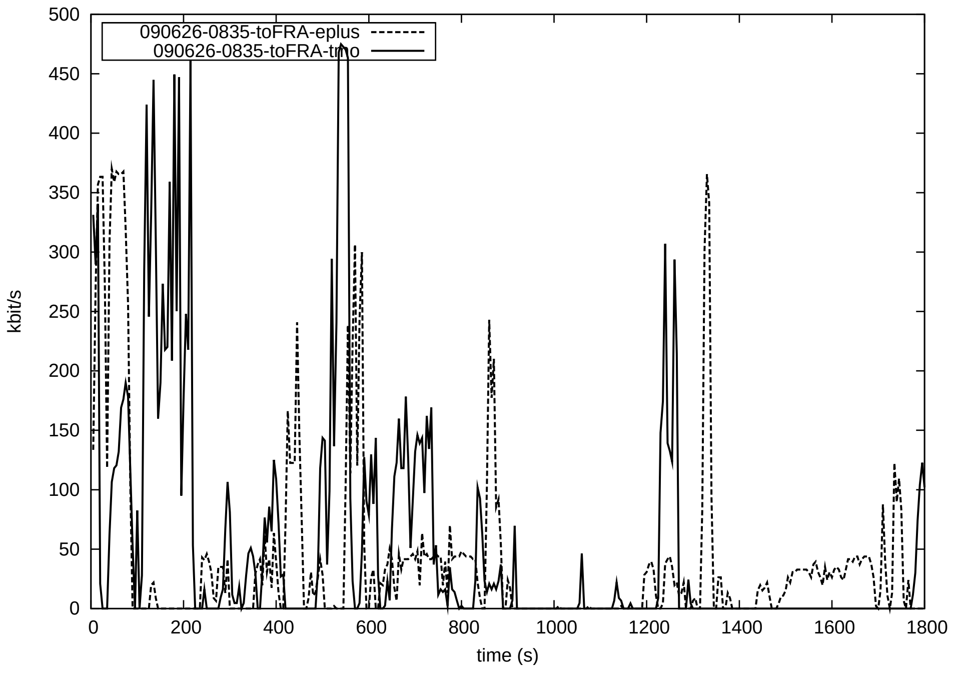


Figure 3.9: Measurement (08:35h), Mannheim to Frankfurt, EPlus and T-Mobile

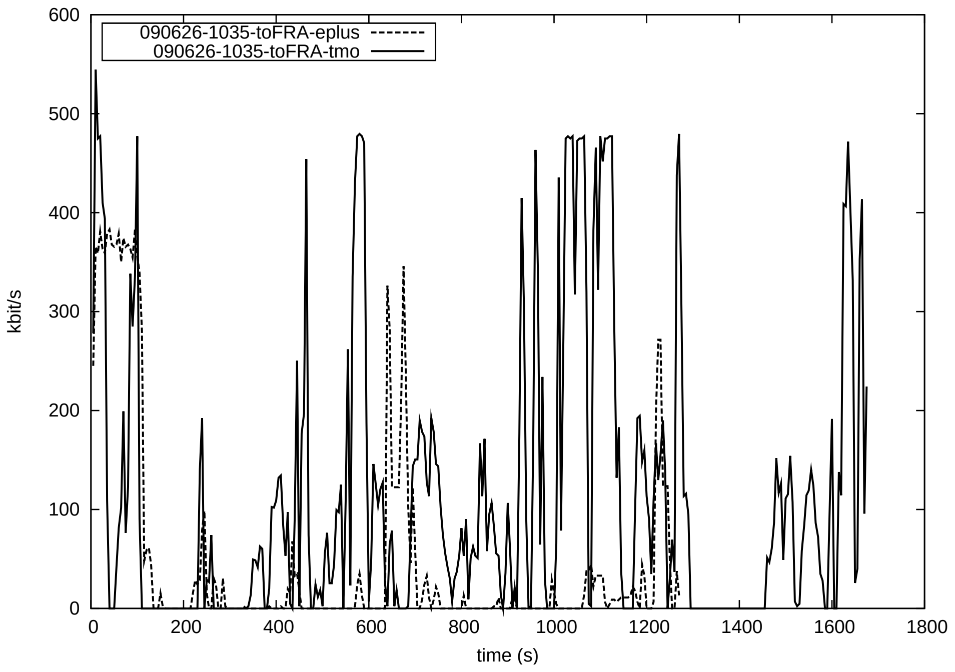


Figure 3.10: Measurement (12:35h), Mannheim to Frankfurt, EPlus and T-Mobile

The following results were all measured on several train rides between Mannheim Hauptbahnhof (central station) and Frankfurt Airport on June 6, 2009. Figure 3.9 shows the achievable throughput for the journey from Mannheim to Frankfurt, starting at 08:35 h for EPlus and T-Mobile. Figure 3.10 is shown as a comparison to highlight the differences in achievable throughput for the same train track, same direction, same measurement setup, but at a different period of time, i. e., starting at 12:35 h.

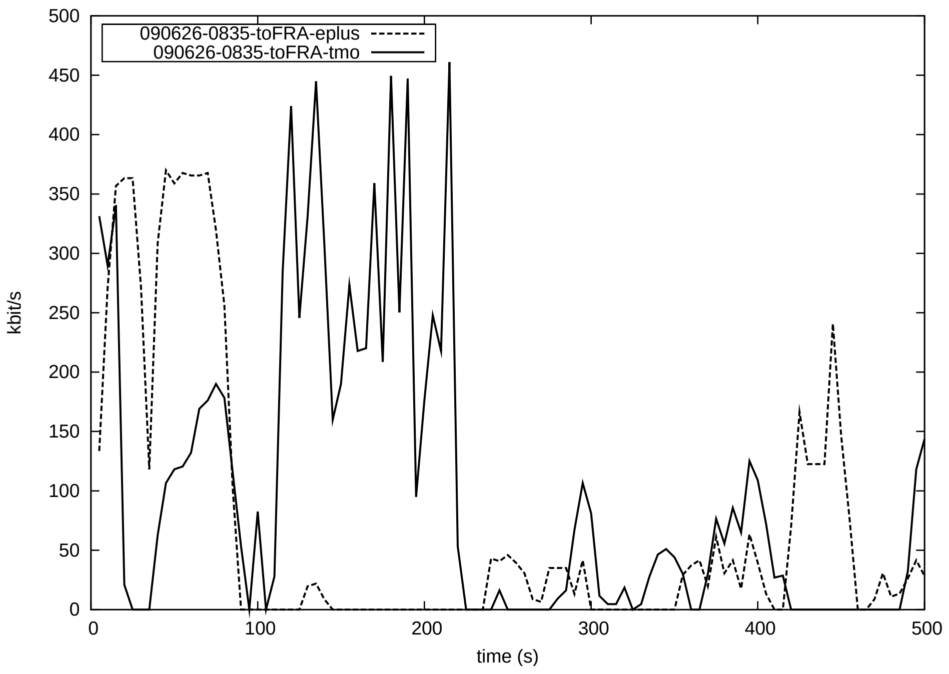


Figure 3.11: minutes excerpt of 2009-06-26, Mannheim to Frankfurt, EPlus and T-Mobile

Figure 3.11 shows a detailed excerpt of the measurement Mannheim to Frankfurt, starting at 08:35 h for EPlus and T-Mobile, to give a more detailed insight of the throughput changes. The achievable throughput is varying and is doing this at short time intervals, e. g., for T-Mobile the throughput is varying between 150 kbit/s and 350 kbit/s in the time from 0 s to 100 s and dropping 0 after this.

Figure 3.10 shows also another impacting factor: failures of the mobile equipment, such as the mobile phone or the connection between the mobile phone and the computer. At time 1250 s, the EPlus throughput drops to zero, as the mobile phone wasn't able anymore to establish a UMTS data connection with the mobile wireless network. Only a full reboot of the phone helped to get it working again. We observed equipment failures at multiple time, on each of the used mobile phones, and also on the connection between the phones and the computers. At some occasions it took several minutes (up to 10 minutes) to re-enable the data connection via the phones, which lead to longer breaks in the measurements. This did influence the measurement results, as there was no achievable throughput measured for this particular operator during this period. This reflects reality in the sense if the equipment fails during the measurements, it will also fail under ”normal” usage by a user.

Another interesting comparison is the achievable throughput of the same operator but at a different time of the day. Figure 3.12 shows the result of the measurement from Mannheim to Frankfurt starting at 08:35 h and at 10:35 h. The two measurement plots do not match in most parts and, for instance, for time period 900 s to 1200 s there is complete mismatch. The run starting 08:35 h achieved almost 500 kbit/s, while there is no throughput at all for the run starting at 10:35 h. This may be result of a temporary drop-out of the mobile wireless network, or as there are multiple active users in a UMTS cell (see also 3.1.2 for considerations about the factors). However, from the observation of the measurement software and also for the human

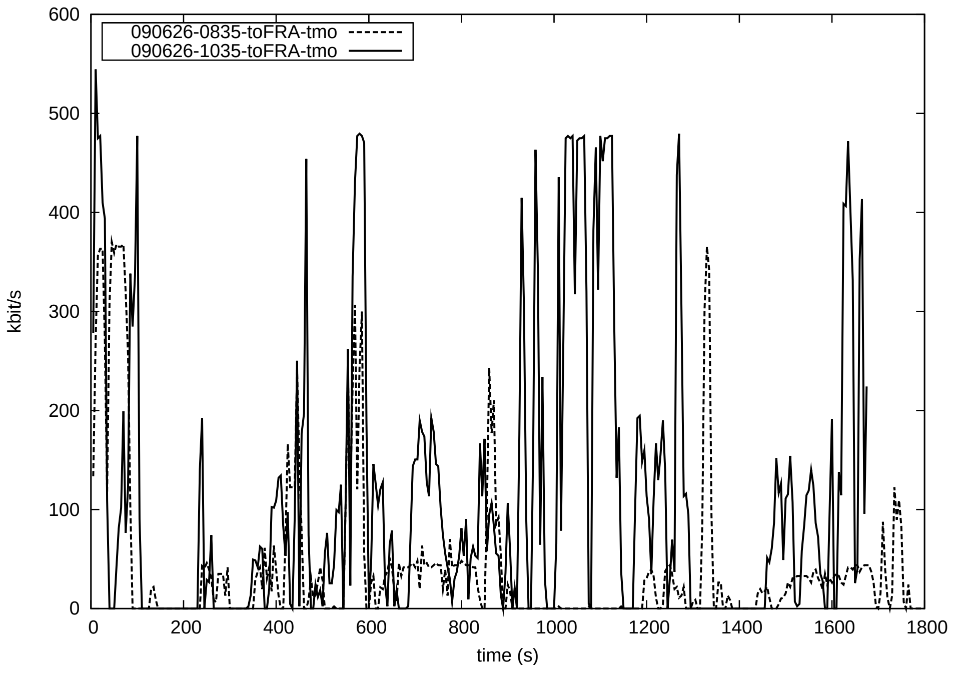


Figure 3.12: Achievable Throughput for some operator, same seeting, different time

observer it is impossible to judge the real impacting factor. It shows the reality of the volatile character of the mobile wireless.

3.4.2 Measurement Analysis and Statistics

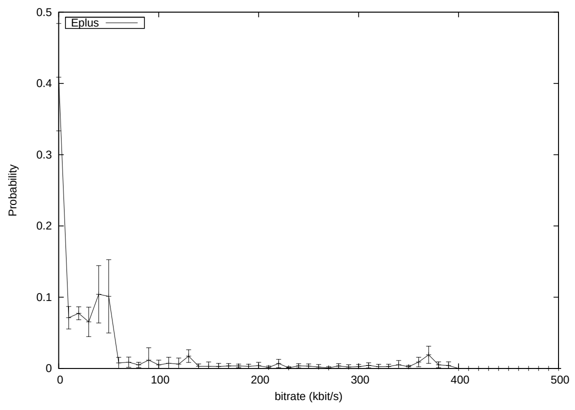
The earlier sections described the measurement methodology and a representative set of measurement results. The results were measured as time series $f(x) = \Theta(t)$ and not usable in this form for out further con sideratios, as we need other metrics, based on the measurements, to later on investigate the behavior of the proposed system. For the further elaborations on what applications can expect on network performance, we use three probability distributions to describe the behavior of the deployment for a given network operator:

achievable throughput What is the probability that a specific throughput value $\Theta(t)$ can be achieved.

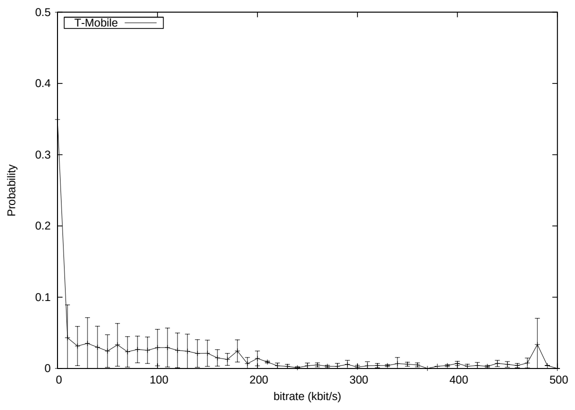
uptime What is the probability a link is up for a specific period of time for a given network operator. A link is up or operational, if the achievable throughput is not zero in this time, i. e., the link is operating with a throughput of $\Theta(t)$.

downtime What is the probability a link is down for a specific period of time for a given network operator. A link is down or not operational if the achievable throughput is zero in this time.

All three probability distributions are a first adequate approximation of the UMTS link behavior, without the intention to be authoritative for all cellular networks, all deployments (i. e., operators), all train tracks, etc.



(a) Eplus Throughput Distribution



(b) T-Mobile Throughput Distribution

Figure 3.13: Throughput Distribution

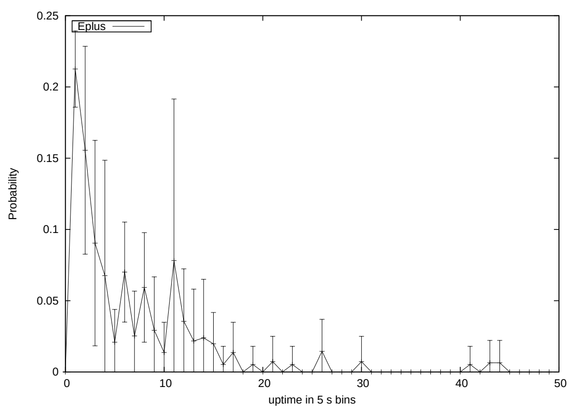
The time series measurements are transformed in probability functions, by quantifying measured throughput, uptime, and downtime into fixed bins. For each bin an average value and the standard deviation are calculated and the bin stores the probability of reaching the actual value of the bin. The average value per bin is used for further calculations and the standard deviation is used to judge the quality of the measurements. For the throughput we use 10 kbit/s base bins, for the correlating uptime and the downtime we use 5 s bins, as this is the measurement interval used during the measurements (see Section 3.4).

The quantified probability functions per operator are shown for the throughput in Figure 3.13, for the uptime in Figure 3.14, and the downtime in Figure 3.14. The standard deviation of each probability function, shown as the vertical bars for each bin, also reflect the variety of the measurement results in Section 3.4.1. There is rarely a measurement run that is resulting in the same measured throughput and up/downtime for the same operator and also for the same direction of travel in the train, nicely illustrating the volatile character of such mobile wireless system. The standard deviation for the measurements is in the range between 40 % and 200 %. This leads to the conclusion that the UMTS environment is so volatile that we cannot make a statement on the to be expected link throughput, uptimes, and downtimes in the general case.

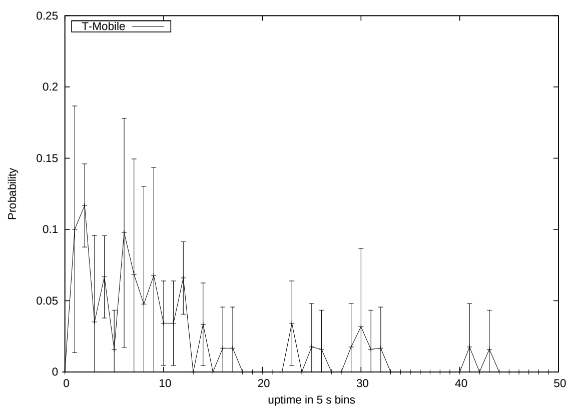
The achievable throughput for Eplus is in general lower than the one for T-Mobile, as shown in Figure 3.13. This observation is in line with the general network deployment status of each of the operator, as T-Mobile is pushing more towards a performant network in many places, while Eplus is deploying UMTS/HSDPA only in larger cities or areas with dense population (metropolitan areas).

For the uptime distribution in Figure 3.14 Eplus has more times with a short uptime period, while T-Mobile has in general a longer uptime period (due to the better network deployment).

Both operators have almost the same downtime behavior, which might lead to the conclusion that both suffer from the same environmental challenges, such as tunnels, road cuts, and are also deploying the better UMTS technology in correlation with the human population at each area.

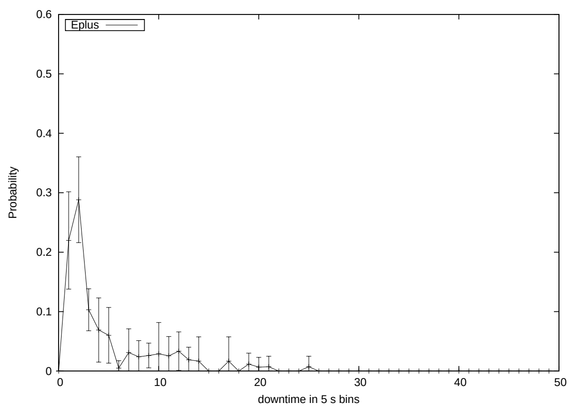


(a) Eplus Up-Time Distribution

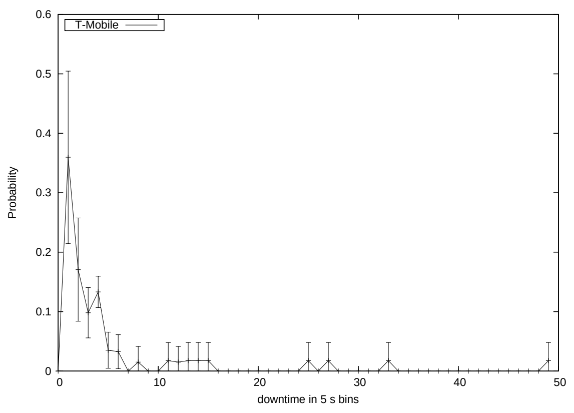


(b) T-Mobile Up-Time Distribution

Figure 3.14: Uptime Distribution



(a) Eplus Downtime Distribution



(b) T-Mobile Downtime Distribution

Figure 3.15: Downtime Distribution

3.4.3 Evaluation of the Statistical Data

We use an negative exponential function, as a first adequate approximation to transform the statistical probability in a form that is usable for computer simulations.

$$P(x) = \lambda e^{-\lambda x} \tag{3.1}$$

The value of $\frac{1}{\lambda}$ denotes the mean value of the negative exponential function and is to be determined for our further elaborations. Each λ for each operator and probability distribution is determined by applying Gauß' least-squares method. The graphs consists of n measured value, i. e., the reading points, pairs $(x_i; y_i)$ with $(i = 1, 2, ..., n)$. We define the distance v_i of the reading point $P_i = (x_i; y_i)$ from the best-fit curve $y = f(x)$ that is to be determined as:

$$v_i = y_i - P(x_i) \tag{3.2}$$

Operator	$\lambda 1_{bw}$	$\lambda 2_{bw}$	$\lambda 1_{up}$	$\lambda 2_{up}$	$\lambda 1_{down}$	$\lambda 2_{down}$
T-Mobile	0.00435	0.00898	0.14453	0.14410	0.021532	0.032654
EPlus	0.007	0.018644	0.08973	0.234367	0.048834	0.0296241

Table 3.4: Link Configuration Parameters; $[\lambda_{bw}] = \frac{s}{kbit}$, $[\lambda_{up}] = s^{-1}$, $[\lambda_{down}] = s^{-1}$

The goal is now to minimize the sum of squared distance in formula 3.3.

$$S = \sum_{i=1}^{n} v_i^2 = \sum_{i=1}^{n} (y_i - P(x_i))^2 \to \text{minimum} \tag{3.3}$$

However, as the least square method is defined for linear functions only, we need to transform the negative exponential function 3.1 in a linear function, by applying equation 3.4.

$$\ln P(x) = \ln\left(\lambda e^{-\lambda x}\right) = \ln\lambda - \lambda x \tag{3.4}$$

We now substitute $u = x$, $v = \ln P(x)$, $c = -\lambda$, and $\ln\lambda = d$, and come to this linear equation suitable for the least square method:

$$P(x) = \lambda e^{-\lambda x} \xrightarrow[v=\ln(P(x)),d=\ln\lambda]{u=-x,c=\lambda} v = d + cu \tag{3.5}$$

The equation 3.5 is now used for the least square method in the equations 3.6 and 3.7

$$c = \frac{\sum_{i=1}^{n} u_i v_i - n\bar{u}\,\bar{v}}{\sum_{i=1}^{n} u_i^2 - n\bar{u}^2} \tag{3.6}$$

$$d = \bar{v} - a\bar{u} \tag{3.7}$$

The inverse transformation is $\lambda_1 = -c$ and also $\lambda_2 = e^d$.

Table 3.4 shows the calculated values out of the equations refe:linear-regression-a and Equation 3.7, after the inverse substitution of c and d to their respective λ values. All $\lambda 1$ and $\lambda 2$ values may intuitively look not suggesting a negative exponential function for the probability distributions. However, it should be noted that the actual data points out of $P_{bw}(x)$, $P_{up}(x)$, and $P_{down}(x)$, have a standard deviation of 200 %. This is also reflected in the differences between the corresponding $\lambda 1$ and $\lambda 2$ values, as they are overlapping given the error indicated by the standard deviation.

3.5 Discussion

The intention of this Chapter is to give an insight to today's mobile wireless networks and what applications running on top of this network can achieve in terms of throughput (absolute throughput and temporal availability). The Chapter is very much focused on the UMTS based systems of two German operators, i. e., Eplus and T-Mobile, and measurements performed on a high-speed train track between Mannheim and Frankfurt. This may be seen as a limitation, but in general this represents a realistic set of what a mobile wireless user can expect. Eplus and T-Mobile have been selected, as both of them represent a class of mobile wireless operator in Germany, but also for other European countries. Eplus is a smaller operator with a less ambitious network deployment, e. g., UMTS only in cities, while T-Mobile is Germany's largest mobile operator with UMTS deployment in lot more places than EPlus. There are two more operators, O2 and Vodafone. O2 can be seen on the same scale as EPlus and Vodafone can be seen equal to T-Mobile.

We focus on a specific technology for performing our study, deliberately neglecting other, relating wireless technologies, such as, Wireless LAN (WLAN) or Worldwide Interoperability for Microwave Access (WiMAX) (amongst other technologies in that space). Both technologies do not provide the same wireless coverage in terms of geographical spread over a country, i. e,. they are not deployed and used nation-wide, for different reasons. WLAN is not made for this purpose, due to the limited radio range, while WiMAX has not made the big step into the market (yet). However, both will more less exhibit the same behavior than UMTS based wireless networks: the achievable throughput per user depends on factors, such as, number of concurrent users in a cell, moving pattern of a user, distance of the user to the base station, etc. A good example of WLAN behavior for moving users is given in [26].

The gained measurements and statistical evaluation results are for sure not representative for all locations in Germany or even beyond. However, they indicate a trend when using Internet applications over mobile wireless. Internet access is almost possible anywhere in Germany, but the *quality* of access to the Internet varies. The *quality* is meant to reflect on how fast and stable you can work on the Internet and must probably be put in relationship to what the average user is used to from fixed line Internet access.

One might argue that the future of mobile wireless will change the situation dramatically, as new radio technologies will bring way more achievable throughput to the users. However, considering recent studies on throughput performance of the UMTS/3G Long Term Evolution (LTE) suggest that the fast and reliable Internet access at any time and under any condition is still a far out dream. For instance, [92] shows bitrates for a LTE radio technology, the next generation of UMTS, in the range of 3 Mb/s at the cell edge for a *pedestrian*, i. e., at walking speeds. However, there seems to be no good reason why mobile wireless networks won't be used by users travelling at speeds much higher than walking speeds. On the other hand, each radio technology is still subject to physics, i. e., the radio is still suffering from challenging landscapes, skyscrapers, etc.

There is to our best knowledge a single related work in the area of measuring UMTS networks in real environments: Liu et al [93] performed application level measurements on top of TCP in several UMTS mobile wireless networks in Hong Kong. Their key findings are larger achievable bitrates in general (due to the measurements in Hong Kong), the preemption of data transfers by voice and video calls, and that it is impossible to predict the actual capacity of a particular cell based on known models.

4 Conceptual System Design

"The System must tolerate wide network variation." [RFC 1122]

4.1 Problem Statement

Most applications have very specific requirements to the network or make many assumptions about how the network is behaving. For instance, most file download programs, such as web browser, do assume a constant TCP connection to the server. However, after a drop-out of the TCP connection, the transfer is not resumed, despite the fact that there is still a need to download. On the other hand, users of these applications have a specific behavior of their used applications in mind. For instance, most of the users are accessing the Internet via fixed line network access, based on, for instance on ADSL or FTTH , with gross bit rates in the range of Mbit/s. Assuming an ADSL link with 6 Mbit/s and an average web page download size of 1 MByte, it takes 1.333 s to retrieve a web page[1]. It takes already a minimum of 20.833 s on a UMTS link with 384 kbit/s, which is approximately 16 times longer than on the mentioned ADSL link. Browsing the web via WWW can be considered as a more relaxed application, where the user tolerates longer delivery times in a certain range.

However, there are many other applications that either do not tolerate low network resources, such as lack of bandwidth, or are prone to volatile network conditions, such as variations of the achievable throughput, delay, or packet loss rate. One application is Voice over IP (VoIP) that requires a minimum achievable throughput (depending on the used codec), low delay and a minimal packet loss (as each packet carries voice fragments). VoIP can be used over many access technologies, as the achievable throughput required by the application is in the range of 100 /kbit/s or below, there are other applications, such as video streaming, that require in general a much higher achievable throughput from the network – depending on content type and image quality – a sustained bitrate in a range from 300 kbps to 16 Mbps, which can be delivered easily and reliably only via wireline access networks.

In general, there is a dependency between the networking needs of the applications and what the network delivers (all parts of the network, but the access networks have been bottlenecks for a long time), i. e., each area is partly driving by the other area. For instance, video streaming applications have not been used in the early days of the Internet, as the access technologies

[1]The average web page site has been determined on a set of news web sites, such as, cnn.com, spiegel.de, stern.de, and nyt.com, by downloading their start page, with text and graphics, but without any video. These values are not representative, but are used for an illustrative purpose and were taken in November 2010.

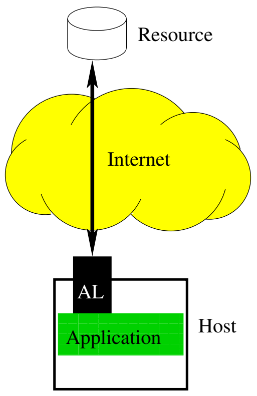


Figure 4.1: Single-homed host

(mainly modems with 48 kbit/s) did not allow the usage of this service. On the other hand, today video streaming is one of the main drivers for the continuous updates of access technologies (e. g., with the introduction of High Definition TV (HDTV) an major uplift of the ADSL technology Very High Speed Digital Subscriber Line (VDSL) was introduced). We see a similar movement for wireless technologies, such as Wireless LAN (WLAN) (e. g., introduction of 300 Mbit/s) and UMTS (e. g., introduction of HSDPA with 7.2 Mbit/s, cf. Section 3.1.2).

The applications assume that they can achieve a certain throughput $\Theta(t)$ over the network, that is in an acceptable range, i. e., being stable within limits and not changing several orders of magnitudes within a short time, and not dropping below a minimal threshold of $\forall t : \Theta(t) \geq \Theta_{min}$. This minimal threshold can be a hard limit of the application, such as the bitrate of the video stream to be delivered, or can be subjective one, as in the case of web access, where the patience or impatience of the user determines this threshold.

Most of today's fixed-lines access technologies and the backbone network behind can fulfill the application's needs. However, assuming that the actual bottleneck is the access-line itself, we can see that the line's bitrate $\hat{B}_\ell$ will be stable over a long period of time (i. e., in the range of hours): $\hat{B}_\ell(t) \approx \hat{B}_\ell \forall t$ with $\hat{B}_\ell$ being the *nominal bitrate*. The achievable throughput on a particular link $\Theta_\ell(t)$ for the application can be lower than $\hat{B}_\ell$, as there are many more impacting factors to the throughput, such as, for instance, the end-to-end capacity of the whole Internet path, capacity of the server to deliver, the aggressiveness of the application, etc.

However, there are cases where users try to use their well-known applications in environments where the Internet access is not as fast and as stable as they are used to. For instance, when using the Internet via a wireless technology, the achievable throughput varies a lot (cf. Section 3.4), or in many areas there is just no high-speed Internet access, i. e., many rural areas have only modem-speed Internet access via traditional dial-up or limited wireless, such as GPRS only. Such an access-line is providing a *resource constrained environment* (see Definition 1.3).

Figure 4.1 shows a typical access-line configuration of a host connected to the Internet. An application on that host accesses on resource in the network. This resource can be of any type, such as, a web server or video streaming server. A host has typically only one active (i. e., used) IP interface, despite the fact that most of today's hosts have multiple interfaces. A typical configuration (as of today (2010)) for state-of-the-art laptops includes Ethernet, Wireless

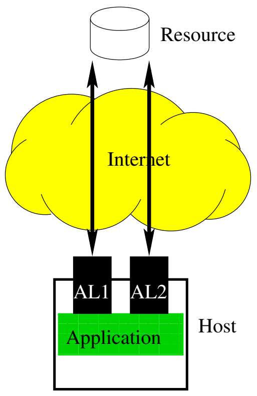


Figure 4.2: Multi-homed host

LAN (WLAN) and Bluetooth, while mobile devices (e. g., mobile phones or smart phones) include UMTS, WirelessLAN (WLAN) and Bluetooth. However, despite the fact that multiple interfaces are available, these interfaces are rarely used *simultaneously*. There are two reasons why interfaces on an IP hosts are not used at the same time: this use case is still not yet supported by the operating systems (e. g., missing transport protocols for this or issues with automatic interface configuration (DNS)) and an economic constrain, as each access-link can require a subscription to a network operator.

Assuming negligible costs for each subscription, the first natural step to cope with a single *resource constrained* access-line is to use host multi-homing, as depicted in Figure 4.2. Host multi-homing allows the host, thus the application, to access the network via multiple interfaces simultaneously This allows the host to aggregate the achievable throughput of each interface for the application's needs. Each interface needs to be connected to the Internet, which typically incurs some costs for a subscription to an Internet Service Provider (ISP) . This costs is charged to the user and most users will have only a single ISP contract for their mobile devices, simply for cost reasons. Another, obstacle for host multi-homing is that each application has to be aware of this and implement multi-homing techniques, unless there is wide support for Multipath TCP (MPTCP) [94], which will enable applications to make use of multi-homing without large modifications (other than on the Unix socket level). There is also an early work on using multiple TCP connections to cope with multi-path transport [95].

The only obvious solutions for host with a resource constrained access link are either to upgrade the link (if possbile at all) or to refrain from using a specific service.

4.2 Cooperative Internet Access

Let's consider a situation where multiple users (with the computers/hosts) are in the same physical location, e. g., in a train or withing a building, for a some longer time period, whereas longer refers to minutes or hours, but not seconds. We assume that each host has at least one interface that connects to the Internet, the so-called *access-link*. We further assume that each host has a second interface that can be used to find other hosts that are interested in a cooperative Internet access, that can be used to coordinate these hosts, and that can be used to exchange information elements freely between those hosts. This second link is called *sharing-link* (marked as SL in the following figures) and it provides reliable high-speed communication between the hosts in the group free of charge, but does not have direct connection to the Internet.

The hosts can now use the *sharing-link* to coordinate the access to the Internet via the *access-links*, either with the goal of letting each hosts using another hosts access-link resource for its own purpose (resource pooling), or by even coordinating the retrieval of data from the Internet.

A key property of the cooperative Internet access is that information elements that are needed by each participating host are retrieved only once from the Internet for the whole local system and not for each participating host. For sure, there are information elements that are not needed by all hosts, but only by a single host.

We assume that the applications used work with information elements. These information elements are the smallest atomic unit the application can work one, where these atomic units do not need to have to be of equal size or to be of equal content or content type. Working with information elements allows to completely abstract from the actual transport of these elements across the network, i. e., it ensures independence from used transport protocol. IP version, Internet path taken, etc; and also which node actually handles, i. e., downloads or uploads, the element. This allows to delegate the network handling of each information element to other entities in the network, such as, in our case the delegation to other peers/hosts, replication of such elements in various places, and eventually the fulfillment of our goal to cooperatively access a service.

> *An application on a host has the need to access an information entity in the network, whereas the path/network access to this entity involves the usage of a resource constrained network path. The host itself is unable to overcome the limitation of the resource, but a set of hosts, being in physical proximity can aggregate their network access resources to overcome it. Next to the aggregation of network access resource, the optimization of the information retrieval is key to the Cooperative Internet Accessconcept. Instead of retrieving equal information elements of an information entity for each application instance separately, equal information elements are only retrieved a single time.*

The network topology is characterized by having two networks connected with each other over at least two links, where there is at least one access-link and one sharing-link, with a node in between, where

1. (a) the combined link bandwidth of all that connecting links is limited
   (b) the connecting links have at least in one network different nodes
   (c) the combined bandwidth of the network connecting links is much lower than the bandwidth of the links within each of the networks itself.
2. the sources and destination of the stream is at different sides of the combined bottleneck set of links.

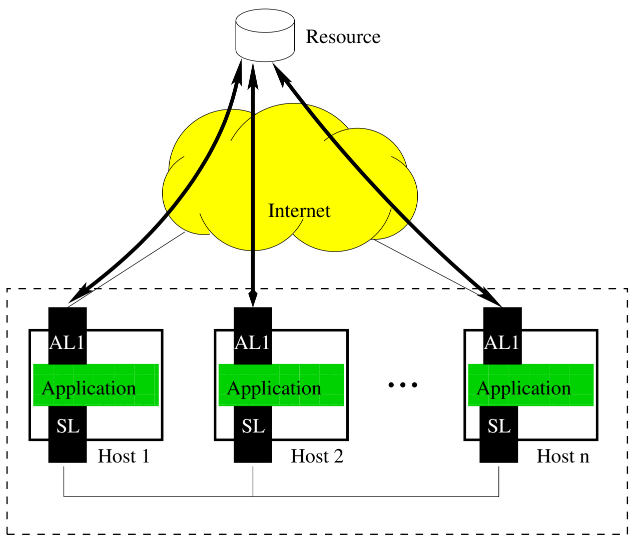


Figure 4.3: Single-homed hosts accessing cooperatively a single resource

Two use cases for this situation are shown in Figure 4.3 and Figure 4.4. The dotted box around the hosts indicates that they are all in the same physical location.

4.2.1 Case 1: Common Interest in Single Resource

Figure 4.3 shows a number of users using the same application and that are interested in the same resource in the Internet. An example for such an application is peer-to-peer video live streaming, where multiple users are watching the same video[2]. Peer-to-peer video live streaming requires a stable and performant network connection to streaming the video in acceptable quality to the users. Since the network requirements are high and video streaming imposes also real-time or at least near real-time requirements to the network environment, the access-links (labeled AL1 Figure 4.3) of the single hosts may not be able to transport the data to the hosts via a single AL1.

Since all hosts are using the same application and are interested in the same resource, they could team up to coordinate their access to the resource and to eliminate the need for each host to retrieve the whole resource on its own. The coordination of which host is handling which piece of the resource, i. e., which information element, is communicated via the sharing-link, as well as, the exchange of the information elements amongst the participating hosts.

4.2.2 Case 2: Multiple Resources with Overlap

The second case is where multiple hosts are agreeing to use cooperative Internet access, but they do not use a single resource, as in Section 4.2.1. An example for this case is Web access via Hyper Text Transfer Protocol (HTTP). All applications are primary the same, i. e., a web browser but the resources accessed a varying a lot between, e. g., web sites or file download.

[2]We use the term video but in the sense that this includes live streaming of TV programs, video on demand, etc

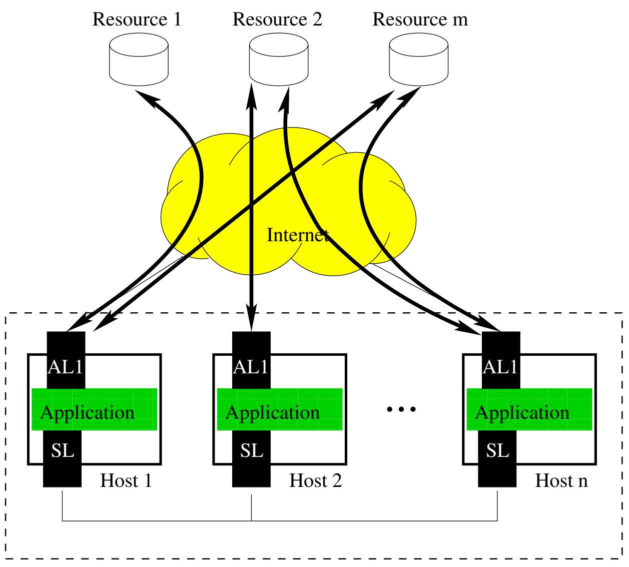


Figure 4.4: Single-homed hosts accessing *m* resources (overlap)

For the most popular usage of web access, i. e., browsing web sites, we can see that most users will use quite a number of websites (see Section 6.1.3), but some web pages are being used by multiple users. Such web pages, are the most popular web pages in country or region. For instance, in Germany, as of today (2010), the most popular web pages used by many people are facebook, twitter, and news sites (spiegel.de, stern.de, etc). The users will access varying parts of the web sites, but many elements of a web page are probably the same, such as, common graphics, advertisements, text, or even videos (see Section 6.1.3 for more details).

Figure 4.4 shows such case, where the applications access multiple resources with an overlap. The applications can coordinate their access to the resources in such a way that information elements that are commonly used by multiple application instances, are retrieved only once via the access-link and redistributed multiple times via the sharing-link. This is helpful in scenarios where the access-links are resource constrained.

4.2.3 Case 3: Multiple Resources without Overlap

A third case, typically explored by the similar earlier works (e. g., PERM [32]) , aims at reusing idle network capacity of neighboring nodes without coordinating on the information element level. The main goal is to maximize the nodes throughput without considering other hosts needs in terms of reusing already retrieved information elements.

Figure 4.5 depicts that nodes can access resources they need via the access-links of other nodes. This approach seems to be beneficial for all participants in situations where there is idle bandwidth available on the access-link. This is mostly the case in fixed-line access networks. Imagine a case where multiple users are living in the same area and can use a local network (e. g., LAN, WLAN, or whatever) to communicate locally. This can be a small town or settlement, where the inhabitants are using mobile wireless networks to connect to the Internet. Instead of using only their own link, that is probably limited due to network limitations or monetary limitations (e. g., better network access may cause higher costs not bearable for an individual),

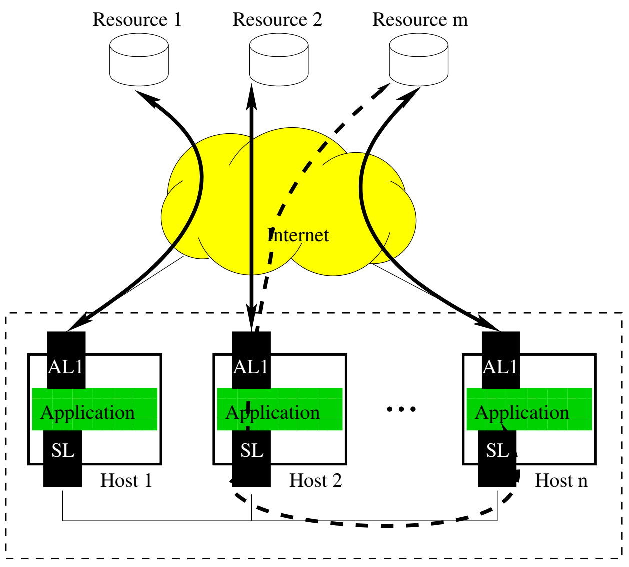


Figure 4.5: Single-homed hosts accessing *m* resources (disjoint)

all participants can team up for cooperative network access. Each participant devotes its access-link to the community, and the sharing-link is used to coordinate the network access and to locally distribute the information elements, or just to trade throughput for MPTCP.

One way of dealing with this situation is to use multi-path protocols and to allow the applications on one host to use the access-link of the other hosts for their multi-path communication with the resource in the Internet. However, this requires connection sharing, i. e., each host has to allow the other hosts to re-use the IP interface with its own flows. This is shown as a dashed-line in Figure 4.5.

4.2.4 Working Example for Cooperative Internet Access

This section discusses how Cooperative Internet Accessworks with a simplified example, based on the use case depicted in Figure 4.4. We assume that 2 nodes are connected via the sharing-link and they are using each an access-link provided by 2 operators. The access-links of the operators are exhibiting a disjoint behavior, i. e., the access-link of each operator does never exhibit a similar characteristics of the achievable throughput. The achievable throughput for access-link 1 (operator 1) is denoted by $\Theta_1(t)$ and for access-link 2 (operator 2) it is $\Theta_2(t)$. For the purpose of illustrating the principle of operation, we assume that the transmission capacity $C_i(t_1 - t_0)$ of each link in a time period of Δt is always equal to the size of one information element $|I|$, see Equation (4.1) and Figure 4.6 for the times.

$$C_i(t_1 - t_0) = \Theta_i(t)(t_1 - t_0) = |I| \tag{4.1}$$

The network is an ideal network, i. e., without packet loss, delay, etc, and allows to send requests from a information element receiver to the information element sender in no time. The nodes have perfect knowledge about when a link will be up and ready to transmit.

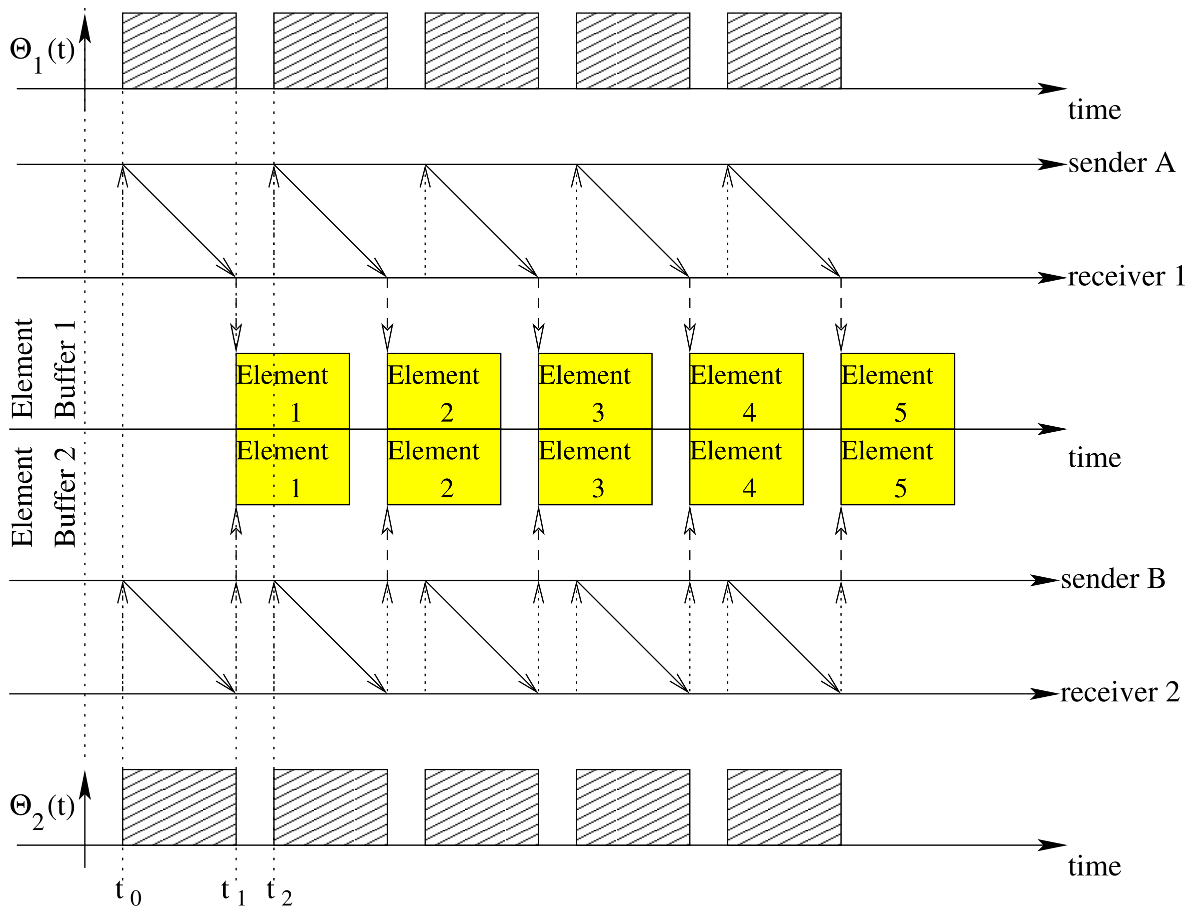


Figure 4.6: Undisturbed information element retrieval

The nodes access a single resource, provided by multiple sources (i. e., *sender A* and *sender B*) in the Internet that provides information elements of fixed size $|I|$ (e. g., such as chunks in peer-to-peer file-sharing). The information element set starts with the index 1 and is increased by one for each new element. Dotted lines from receiver to sender are the requests to the sender for information elements. Bold lines from the sender to the receiver is the transmission of an information element. Dashed lines to the element buffer show the point of time when an information element has been retrieved and can be placed in the element buffer for the next steps in processing. We use a clocked available throughput function $\Theta_i(t)$ that is delivering the capacity between t_0 and t_1 and a break of $t_2 - t_1$ (see Figure 4.6).

The "undisturbed" case, when there is always sufficient achievable throughput to retrieve an information element, is shown in Figure 4.6. Each node is retrieving the information elements on its own, whenever the access-link is up and running. Retrieved information elements are placed in each nodes buffer, labeled as *element buffer 1* and *element buffer 2*. As a results, node 1 (on top) and node 2(at bottom) are each able to retrieve 5 elements.

Now we assume that clocked achievable throughput of the access-links cannot be used when available. This is depicted in Figure 4.7 by having used slots (the boxes filled with a pattern) and unused slots (the empty boxes). Here we can see that node 1 is able to retrieve 3 elements vs. only 2 for node 2. However, considering all available slots of both access-links, i. e., $[t_0 : t_1]$,

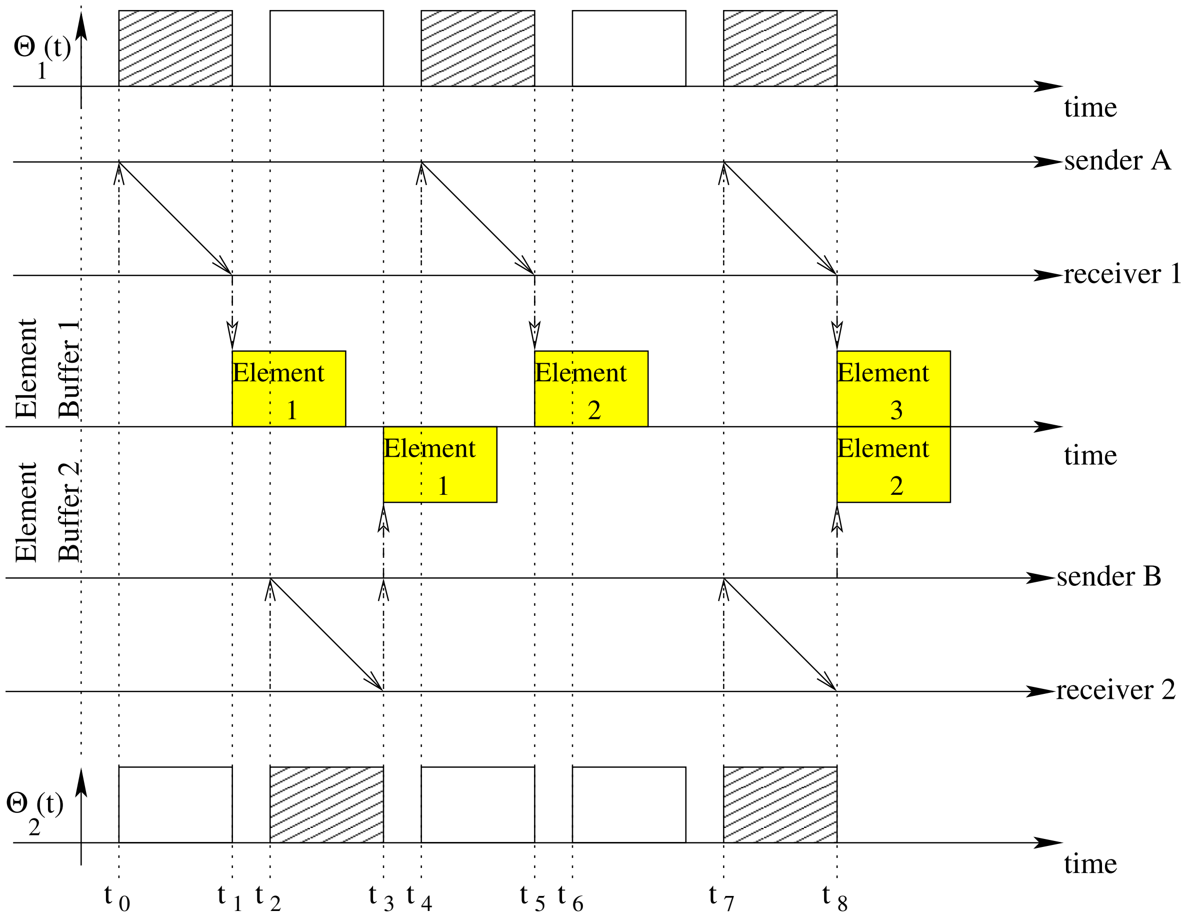


Figure 4.7: Disturbed information element retrieval without Cooperative Internet Access

$[t_2 : t_3]$, $[t_4 : t_5]$, and $[t_7 : t_8]$, it is obvious that there would be sufficient achievable bandwidth to retrieve all information elements in the same time, as in Figure 4.6, if the nodes would coordinate the information element retrieval.

In Figure 4.8 we can see the same network situation as in Figure 4.7, but where the nodes fill the joint element buffer in a coordinated and cooperative way. In this simplified example, the nodes assign information elements in a subsequent way to the next link that is coming up. At $t = t_0$ node 1 starts to load element 1 as its access-link is up, while the access-link of node 2 is down. Vice versa this is done at $t = t_2$ and $t = t_4$, as either $\Theta_1(t)$ or $\Theta_2(t)$ contributes throughput to the system. At time $t = t_6$ we observe that no link is used to retrieve an information element, while at $t = t_7$ both links are used simultaneously. The elements are retrieved only once for the whole system and placed in a joint element buffer that is accessible by both nodes. At the end of the time period the joint element buffer contains all 5 elements.

There are two interpretations for the unused slots: Such a slot can represent a drop of the access-link's bitrate to zero, due to a congested link, a link failure, or changes in the environment of a wireless link. Second, by using Cooperative Internet Accessthe throughput consumption on a access-link is lowered, leaving idle slots for saving network resources or as idle resource to be used by other applications.

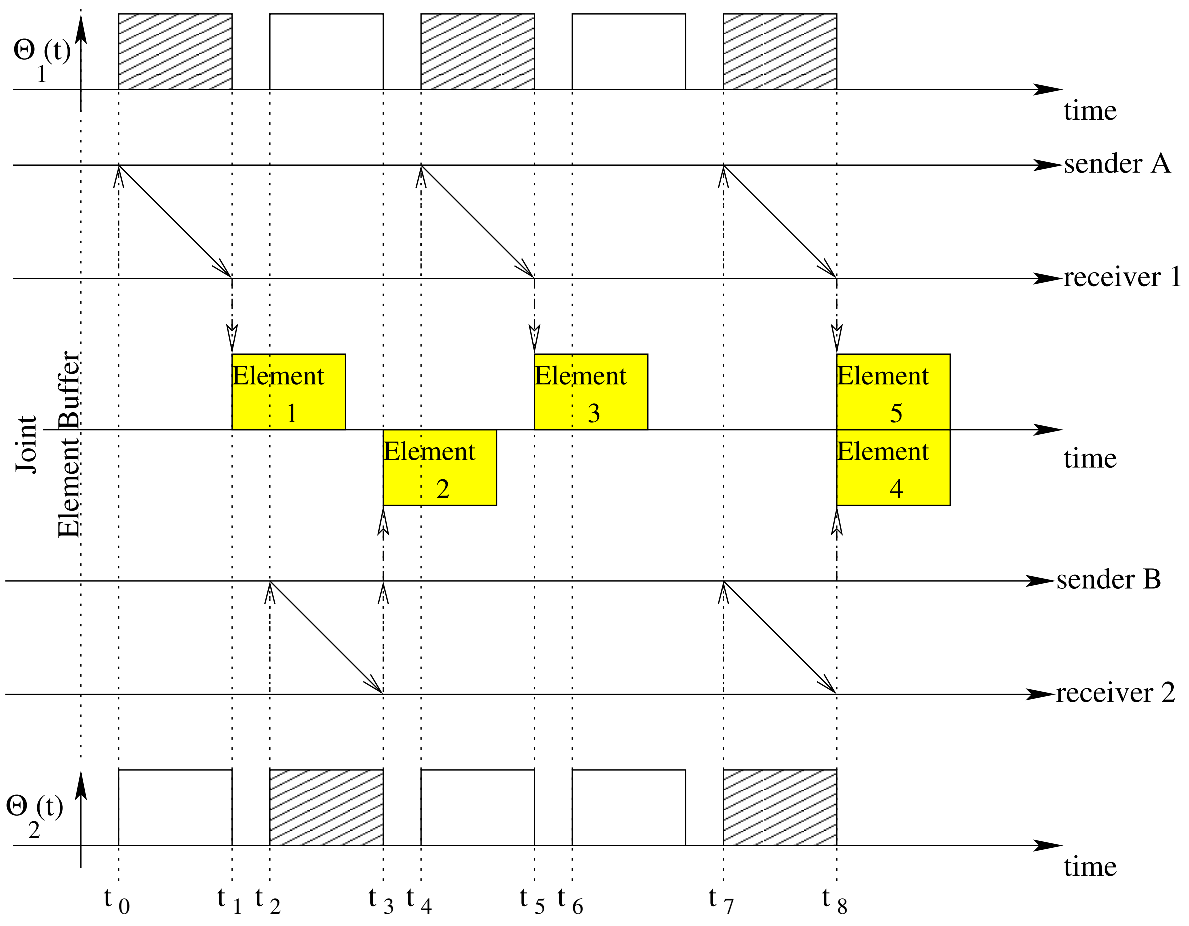


Figure 4.8: Information element retrieval with Cooperative Internet Access

4.2.5 Characterization of the Access-Links

The cooperative Internet access scheme requires to characterize the access-link for the further discussions and also algorithms, as the links are not behaving as used to in non resource constrained environments. Cooperative Internet Accessdoes not assume a specific link behavior per se, such as, whether the links are fixed-lines or wireless, symmetric or asymmetric, shared or non-shared medium, but the behavior of links influences the overall behavior. This section elaborates how Cooperative Internet Accesscharacterizes an access-link.

In Section 1.2 we define the general property of an access-link and here we discuss more the technical characterization. For Cooperative Internet Access, an access-link is:

- a link that connects the node to the Internet via an access network of any type (wireless or wired link) or particular technology (e. g., UMTS);
- a resource that can be described by:
  - its nominal bandwidth $\hat{B}_\ell(t)$, which denotes the maximum bitrate of this link.
  - a notion of the achievable throughput $\Theta_\ell(t)$, which is usually $\Theta_\ell(t) \leq \hat{B}_\ell$

  - a notion of the stability of the link, i. e., how the probability of a link for being up vs. down is.

- a single resource connected to an IP interface.

We characterize a link with several probability functions that describe single aspects of a link behavior in the following paragraphs and provide a summary in Table 4.1.

We define two states for a link: A link is up, if data can be transferred via that link. A link is down, if no data can be transferred. This can lead to confusions with a particular link technological notion of up or down. For instance, Ethernet calls a link up, if there is an active sender/receiver pair. However, from our perspective the link would be still down unless there is an IP address and connectivity to the Internet. The probability function $P_\ell(up)$ denotes the probability that a link is up and can take only two values:

Table 4.1: Link behavior and associated probability functions

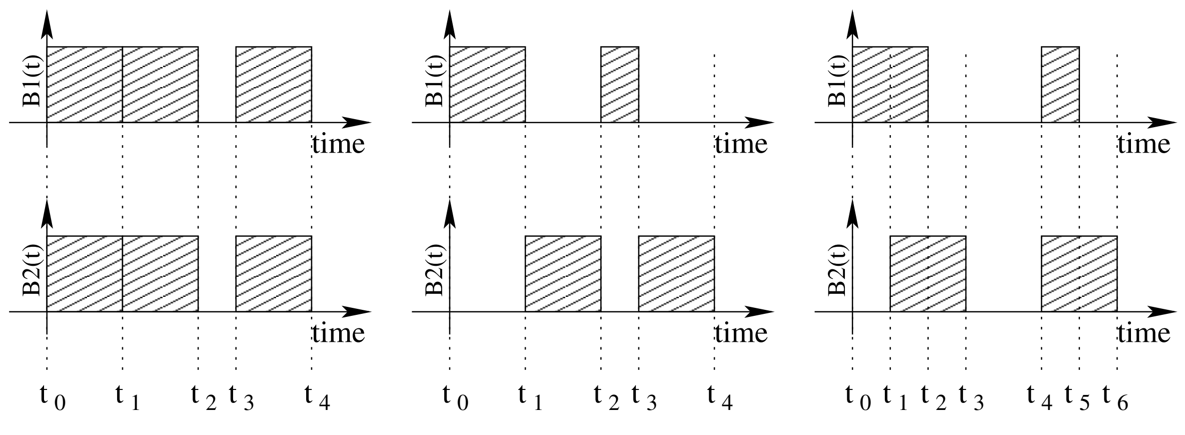
Behavior	Name	Description
Achievable throughput	$P_\ell(\Theta)$	achievable throughput for the application
Up-time	$P_\ell(uptime)$	how long a link is up
Down-time	$P_\ell(downtime)$	how long a link is down

Each link can be further characterized by a probability of achievable throughput $P_\ell(\Theta)$, by a probability $P_\ell(uptime)$ describing how long a link will be up and delivering throughput, and by a probability $P_\ell(downtime)$ describing the link down duration. Let the link be in state up, then $P_\ell(\Theta)$ gives the achievable throughput for a particular event of $P_\ell(uptime)$. Equation (4.2) shows the range a result can take. Table 4.1 summarizes all probability functions used to describe a link.

$$0 > P_\ell(\Theta) \geq \hat{B}_\ell \tag{4.2}$$

We further use a time based classification of the achievable throughput $\Theta_\ell(t)$ of a particular link. The time based classifications is used by Cooperative Internet Accessto judge whether a set of links is joint or disjoint. Two links are joint if they exhibit the same temporal behavior of achievable throughput $\Theta_i(t) = \Theta_j(t) \forall t; i \neq j$, as shown in Figure 4.9(a). Joint links will have no achievable throughput or have achievable throughput for the same or similar period of time, while disjoint links will no common behavior, as shown in Figure 4.9(b). There can be links that are partially joint, as shown in Figure 4.9(c). The link behaving can be used in the scheduling decision, e. g., 2 urgently needed information elements can be scheduled on 2 disjoint links, avoiding a loss of both in any case.

The system will operate in a real deployment where even joint links are not behaving the same but exhibit a *similar* behavior: There will be very likely no perfect joint links, as there are slight differences even between actually joint links. For instance, the used hardware interface (e. g., different vendors) might lead to a drop of the radio signal at one device much earlier than at



(a) joint links (b) disjoint links (c) partially joint links

Figure 4.9: Classification of Link Behavior

another device. This will result in that these links are classified as partially joint links, even though they should be classified as joint links.

The exact definition of the method on how to determine if links are joint or dis-joint is for further study. Existing methods of signal analysis are applicable here (cf., [96]) and can be reused in an implementation.

4.2.6 Characterization of the Sharing-Link

The sharing-link has to be much more stable than the access-link, as it is the ”backbone” of the local system where all nodes connect to and exchange messages amongst each others. The sharing-link is assumed to be stable over the operational time of the system, i. e., the link is established and is stable with respect to its characteristics (e. g., nominal bitrate and latency) The link provides high-speed data exchange between the participating nodes and is free of charge. The participating nodes connect via the sharing-link and communicate directly between each other.

The nodes in the local system need to detect other nodes which are in the same physical context, i. e, the need be in physical proximity. Therefore, the sharing-link consists of a single link-layer broadcast domain, such as Ethernet or WLAN radio coverage, which ensures that only nodes connected to this link can actually participate and not nodes which are 1 IP hop away, as these nodes may be only reachable via a resource constrained environment.

4.2.7 Information Element vs. Flow/Packet

Cooperative Internet Accessworks in the level of information elements to trade data between the participating nodes with the local system and to schedule data for the retrieval via the access-links. Other approaches, such as PERM [32], CUBS [33], or MOPED [34] work on the flow level for the ”data” to link assignment (cf. also Section 2.3 and also Section 2.8).

The flow-based approaches can assign a new flow to an interface on the origin node. This interface determines which route the flow will take and thus also which access-link is used. The flow cannot be reassigned to another access-link if, the flow is not retrieving the data with the expected achievable throughput. This caveat is fine for non resource constrained environments, as the expectation is that the achievable throughput will not change drastically, and there is no remedy to this, if the throughput changes. The application on top of the flow will be stuck with the low throughput.

A new approach which allows to move flows from one interface to another one is Multipath TCP (MPTCP) [21]. MPTCP can reassign flows from one interface to another, but requires a node to have multiple interfaces connected to the Internet at the same time.

None of the approaches known from the related work allows what Cooperative Internet Access-demands:

system-multihoming each nodes needs a single access-link only, but the system of the local nodes is multihomed;

multiple resource replicas typically there is a single replica of the resource the nodes want to access, where multipath (flow solutions) can use multiple access-links to a single node in the Internet. However in many peer-to-peer applications there are **multiple replicas of the same resource** available at different nodes in the Internet. This cannot be handled by scheduling flows the multiple access-links, as the multipath flow approaches are still and end-to-end protocol between **two nodes**;

sharing of elements a benefit of Cooperative Internet Accessis the cooperative retrieval of elements and sharing of these between the participating nodes. This is not possible on the flow level, but needs a clear distinction on the element level.

delegation Cooperative Internet Accessallows to delegate the retrieval of elements to other nodes. A delegation of flows is not possible, as flows are bound to a local interface.

4.2.8 Applications and Cooperative Internet Access

There are a number of applications used and for some of them Cooperative Internet Access-can be used to enhance their behavior in resource constrained environments. We list a (non-exhaustive) number of application types used in the Internet (May 2010) and discuss which of them can be used together with Cooperative Internet Access. The applications are classified as:

remote access Applications that allow to access the command line or shell of a computer via the Internet. Examples are, e. g., ssh [97] or telnet [98]. Remote access applications use a single flow of data between client and server.

Web Browsing (WWW) Web browsing is the most common type of application used in the Internet (as of today (2010), cf. slide no. 15 of [99]) Web browsing relies on the Hyper Text Transfer Protocol (HTTP) [RFC 2616], variations of Hyper Text Markup Language (HTML) [HTML] and other media types, such as video or images. A single web page is usually made up of several self-contained elements (atomic units), such as text, images, videos, etc.

RPC Remote Procedure Call (RPC) is used to implement computer procedures on remote systems. An example for a web-based RPC protocol is the Simple Object Access Protocol (SOAP) [RFC 4227]. Most of the protocols are transaction based and will results in a flow of data being sent to the server and a flow of data returned to the client.

VoIP Voice over IP (VoIP) implements telephone voice service over IP-base networks. It typically relies on the Session Initiation Protocol (SIP) [RFC 3261] for session signalling and on the Real-time Transport Protocol. (RTP) [RFC 3550] for media transport. VoIP protocols allow typically the implementation and usage for video conferencing, text messaging and also instand messaging. The media transport is a flow of small RTP packets exchanged between the SIP peers.

video streaming Video streaming is used for live streaming of TV events or video on demand and relies on Hyper Text Transfer Protocol (HTTP) transport, i. e., progressive download, or on dedicated protocols, such as the Real-Time Streaming Protocol (RTSP) [RFC 2326] as signalling protocol and on RTP for media transport (amongst many others). The media transport is a flow of RTP packets or a bit flow over HTTP, from the video server to the client.

P2P file-sharing peer-to-peer file-sharing uses peer-to-peer technology to share files amongst a group of users. An application example used is bittorrent. peer-to-peer file sharing uses atomic units, the so-called chunks, that are exchanged between the participating peers.

P2P streaming peer-to-peer video streaming implements a peer-to-peer version of video streaming, re-using principles of peer-to-peer file-sharing. peer-to-peer video streaming uses atomic units, the so-called chunks, that are exchanged between the participating peers.

filetransfer Protocols for file transfer, such as the File Transfer Protocol (FTP) [RFC 959] or HTTP are used to download or upload files from a server. Some file transfer protocols allow that the file is retrieved in self-defined segments, which can be considered as atomic units (cf. the REST command of FTP in [RFC 959]).

We classify now the above mentioned applications accordingly to the below application properties, as Cooperative Internet Accessrequires a certain application properties, such as the use information elements in the application. Table 4.2.8 lists the applications vs. the properties.

Application Property 1 *Time-critical: The delivery of the applications data is bound to real-time or near real-time constraints. The application will fail or not function properly if the application data is not delivered within time limits.*

Application Property 2 *Interactive: The application requires to be responsive to a challenge and response pattern. This requires that the response is delivered in a timely manner to the response.*

No.	Property	remote access	WWW	RPC	VoIP	streaming	P2P file-sharing	P2P streaming	filetransfer
1	time-critical	✓	✗	○	✓	✓	✗	✓	✗
2	interactive	✓	✗	✓	✓	✗	✗	✗	✗
3	data volume	low	medium	medium	low	high	high	high	high
4	inf. element prop	✗	✓	✗	✗	✗	✓	✓	✓

Legend

✓	applicable
○	partially applicable
✗	not applicable

Table 4.2: Selected Application Properties

Application Property 3 *Data volume: A relative approximation about the data volume caused by that application. For instance, remote access requires less data volume compared to video streaming.*

Application Property 4 *Information element: Does the application have information elements, i. e., atomic units, on the application level. For instance, peer-to-peer video streaming uses chunks as atomic units, while video streaming uses a stream of RTP packets and does not have information elements.*

Only the applications in support of information elements, e. g., WWW, peer-to-peer file-sharing, peer-to-peer video streaming, and filetransfer, can be handled by Cooperative Internet Access. Out of the above, we chose two applications for the further considerations that have information elements: peer-to-peer video streaming as this application is time-critical and as counterpart of a non-time-critical we chose WWW. Peer-to-peer streaming is discussed in Section 5 and WWW in Section 6.

4.3 Conceptual Design

This section describes the conceptual design of Cooperative Internet Accesssystem. Cooperative Internet Accessrequires the extension of existing applications, but no change in the operating system or in any part of the networking subsystem.

The dotted boxes of Figure 4.10 show the elementary building blocks of an application. The *element retrieval or exchange* is the communication subsystem of an application that is in charge of sending and receiving of data to and from peers/servers in the Internet via the access-link. Data that is sent or received by the application passes through the *element buffer*, which passes

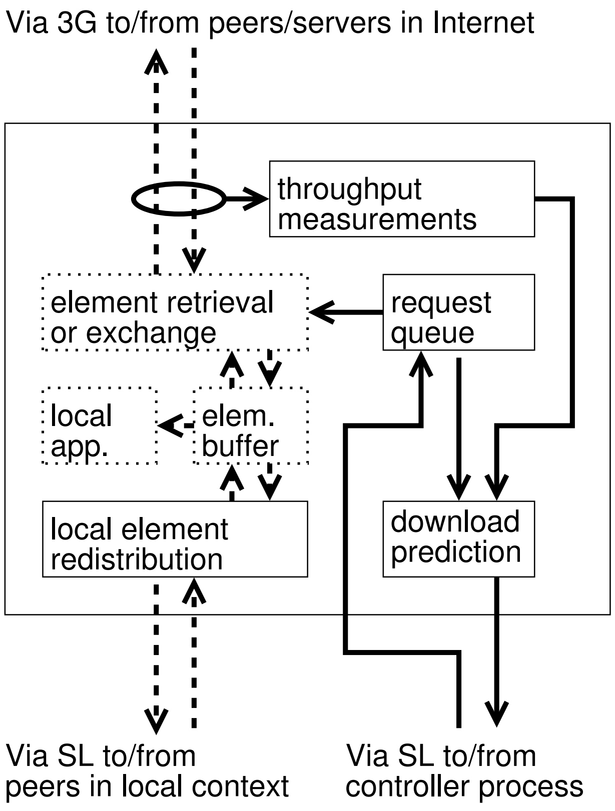

Via 3G to/from peers/servers in Internet
throughput measurements
element retrieval or exchange
request queue
local app.
elem. buffer
local element redistribution
download prediction
Via SL to/from peers in local context
Via SL to/from controller process

Figure 4.10: General Node Architecture

over or receives data from the local application. The *local application* is the actual processing entity in the application (e. g. the rendering engine of a web browser).

For Cooperative Internet Access, the application is extended with the bold line boxes elements shown in Figure 4.10 to add the required Cooperative Internet Accessfunctionality. The left hand-side of the node architecture belongs to the data plane of Cooperative Internet Access, i. e., where the actual information elements are handled. The right hand side of the node architecture belongs to the control plane, i. e., the building blocks are required to operate Cooperative Internet Access.

The *local element redistribution* module interconnects the participating Cooperative Internet Accessnodes for their local information element exchange (Cooperative Internet Accessdata plane) via the sharing-link. The information elements are read from or written to the *element buffer* of the local application instance, depending on whether they are being sent or received from the node.

The control plane of Cooperative Internet Accessare the building blocks on the right hand side in Figure 4.10. These blocks are added to the node and interact with the actual applications and also with the networking stack of the host operating system. The *throughput measurements* module is constantly collecting passive measurements about the achievable throughput on the access-link in the upload direction (i. e., from the node to the Internet) and in the download direction (i. e., from the Internet to the node). Those throughput measurements are reported to the *download prediction* module where they are processed and reported to the *controller* *process*. The *request queue* module stores all requests for information elements that have to be made by the node and it controls the *element retrieval or exchange* module. The *request queue* module reports also the current requests to the *download prediction* module. The *request queue* module receives its information from the *controller process*. The controller process is shown as being outside the node, as a meta process, as the controller itself can be either on a different node, if there is a single centralized controller, or it can be distributed amongst all participating nodes. The controller process is logically separated from the general node architecture.

The following sections describe the modules in detail.

4.3.1 Throughput Measurements

Cooperative Internet Accessaims at overcoming the limitations of particular links in a cooperative way with other nodes. The first step to understand if a link is resource constrained is to measure the achievable throughput of the link $\Theta_i(t)$.

The achievable throughput is measured for the uplink and the downlink traffic at a fixed common sampling frequency f_s. The measurements samples are stored for a limited time period, i. e., a moving window lasting from now to δ seconds in the past, as only a recent history of the link's throughput is of interest for the system. The maximum measured throughput in the $\left[\hat{\Theta}_i\right]_{\delta}^{\text{now}}$ is also recorded, as well as, the nominal link bitrate $\hat{B}_\ell$. $\left[\hat{\Theta}_i\right]_{\delta}^{\text{now}}$ can actually lower than $\hat{B}_\ell$, as the access-link itself may not be the limiting factor, but a bottleneck link down the path in the network.

The module additionally collects the current link status, i. e., if the link is up or down. A link is reported as up, if the link layer indicates the link to be up and if there is network layer configuration of that link (including support for the network layer, such as name resolution services). The network layer information may include an IP address, default route, DNS servers. The measurement module can trigger a short test sequence to see if the link is properly working (e. g., sending of ICMP messages).

The throughput measurement module connects to the download prediction module and reports all measured parameters ($\Theta_i(t)$, link up or down, $\left[\hat{B}_i\right]_{\delta}^{\text{now}}$, and $\hat{B}_\ell$).

4.3.2 Request Queue

Typically, applications do not consider when to up or download their information elements via the Internet. They typically assign the elements to the element retrieval or exchange module and wait for the completion. For Cooperative Internet Accessthe *element retrieval or exchange module* is augmented with the *request queue*. The request queue determines which information element will be retrieved via the access-link with what priority. The request queue module instructs the *element retrieval or exchange module* to work on a specific information element.

The request queue is a priority queue, meaning that elements at the queue's head are handled with the highest priority. The queue is controlled by the controller process. The controller process controls which information element requests will be stored in the request queue or if the queue entries need to be changed. The dimensioning of the request queue's size depends on the used application is defined in the respective section.

The request queue module is connected to the download prediction module. The queue module reports the current queue entries (i. e., which information elements are queued) and the size of the information elements to be handled.

4.3.3 Download Prediction

The download prediction module combines the information received from the throughput measurements module and the request queue module. The download prediction has to judge when the current ongoing upload and/or download will be finished and when the other elements in the queue will be finished.

This module uses the measurement results from the measurement module to compute the probabilistic metrics outlined in Section 4.2.5. These metrics are used to predict the future development of the achievable throughput and in this way to estimate the completion time of each information element entry in the request queue. For each information element request, the prediction lists the estimated completion time and the probability that this completion time will reached. A predictor may list multiple completion times with varying probability values for a single information element request. A predictor may also give a general guidance about the expected achievable throughput for an interval and the probability that this estimation will occur.

The Cooperative Internet Accessconcept does not enforce the usage of a single, standardized download prediction algorithm between all nodes. In fact, each node can deploy and use its own algorithms. It is only required that the output of the algorithm is interoperable with the output of all other algorithms, i. e., the output must fit to the metrics described in Section 4.2.5.

4.3.4 Local Element Redistribution

The element buffer, containing the information elements for the local application, is filled by information elements retrieved via the access-link (via the element retrieval or exchange module), and via the sharing-link with information elements from other local Cooperative Internet Accessnodes. The local element redistribution module connects to the other local Cooperative Internet Accessnodes and takes care that the information elements retrieved by the node itself are sent to the other nodes in need of the information element or retrieved from other nodes. The local element redistribution keeps also track about all locally participating peers, i. e., the group membership management.

4.3.5 Controller Process

The earlier sections described the modules that are needed on any participating node, but left out the controller process. The controller process is the focus point of the system, where all information from the peers is collected and processed, the scheduling of the information elements is performed, and from where the request queues of the local nodes are maintained.

The controller keeps a list of access-links and the nodes offering these access-links. There may be more access-links than nodes, as a single node can offer multiple access-links. The controller does not consider nodes in its computations, but uses the access-link as working unit.

The functional tasks of the controller process are:

situation management Using Cooperative Internet Accessmay not be useful in every situation, as nodes are mobile, connecting to the Internet at different times with changing access-links/types. For instance, when the node is connected to a ADSL line, it may not beneficially to use Cooperative Internet Access, as the line will be very likely not resource constrained. The usage of Cooperative Internet Accessin that situation will cause a not needed overhead. Therefore, it is necessary for the system to detect the situation. Second, the system should detect which of the links is used as access-link and which other links, if any, can be used as sharing-link.

group management Cooperative Internet Accessnodes that want to participate need to decide in the **situation management** if they are going to participate in the system. A node that is participating in Cooperative Internet Access, announces its presence to the system, what resource it wants to access (e. g., which video channel), and the link resources the node can commit (e. g., number of access-links and type of link). The controller collects this information and monitors the presence of the nodes throughout their participation. Nodes that want to leave to the system also announce this or the controller will notice that they have left when checking the peers for their liveliness.

link measurements The controller process collects the measurement results of the links from the download predictor. The measurements are stored and used by the **information element scheduler**.

information element requests The controller process collects the requests for information elements from the various participating nodes and uses this for the **information element scheduler**. This step can be be implicit, if the peers have a pre-defined set of information elements to be handled which are known a priory to the controller process (e. g., number of chunks in file-sharing). Nodes can add a priority value to the information elements, indicating their preference for handling.

status of information elements The controller process collects the status of the information elements assigned to the request queues on the nodes.

information element scheduler The main task of the controller process is to coordinate the handling of information elements with the local Cooperative Internet Accessnodes. The controller takes the list of information element requests and computes how these elements can be handled according to their priority with the current set of links. The elements are ranked according their priority and matched against the reported achievable bandwidth (and probability) of the links, to schedule the information element requests to a link or a set of links. The matching process depends on the used application. Section 5 and Section 6 discuss 2 applications and the resulting scheduling algorithms.

element assignment The controller process collects assigns after the information elements, after the **information element scheduler**, to the request queue's of the local nodes.

The functional elements described above can be grouped into:

overall system control The overall system control comprises all elements that are not specific for an application, such as situation management, link measurements and group management.

application specific system control The functional elements information element requests, status of information elements, information element scheduler, and chunk assignment are specific for each application that uses Cooperative Internet Access.

This differentiation leads to the separation between an overall system control that is application independent and a controller instance per application used. The application specific controller instance is required as the semantics of each application is different in terms of types of information elements, priority of handling of information elements, dependencies between information elements, and time constraints.

4.4 Discussion of Design Options

Section 4.3 discussed the conceptual design of Cooperative Internet Accessbut left open the implementation design options. This section discusses the design options for Cooperative Internet Access.

4.4.1 Detecting a Constrained Resource Situation

The first step for a node is to detect if it is in a resource constrained environment. This cannot be done in general as it depends on the applications used on the host and is therefore partially discussed here and also in Section 5 and Section 6.

The general design options are:

- the user can ”hit a button” to tell the system to enable Cooperative Internet Access, if the users realize such an environment;
- the application can be adapted to perform a situation check on its own, such as some applications, e. g., as Skype [100] does with their voice codec settings. Skype adapts the voice codec bitrate or switches the codec if the achievable throughput is too low.
- the application could register a minimal achievable throughput with the Cooperative Internet Accesssystem and the system decides on its own to start working.

4.4.2 Setting up the Sharing-Link

The sharing-link needs a working IP address configuration, i. e., each interface has to have at least one IP address (independent whether IPv4 [RFC 791] or IPv6 [RFC 2460] is used), before the Cooperative Internet Accesssystem can start to work. If the sharing-link is made up of existing network infrastructure, such as a WLAN access point, the access point could also assign IP addresses to the nodes (e. g., via DHCP [RFC 2131, RFC 3315]).

The local Cooperative Internet Accessnodes will need to setup their own sharing-link infrastructure, in the case that there is no existing infrastructure. There are multiple ways of setting a local area network, such as Ethernet cross-connect for 2 nodes, an Ethernet switch for more than 2 nodes, self-provided WLAN base stations, or WLAN in ad-hoc mode. If WLAN is used, the nodes can use a pre-defined network name (e. g., ”Cooperative Internet Access”) to find other nodes that already have set up such a network, or to create its own network with that name.

However, it is very likely that the sharing-link will be self-provided by the local nodes and neither of them will either able or configure to provide IP addresses to other nodes. The users could agree on an IP address configuration scheme and manually configure it, but this may be a non-trivial task for the average user and thus prone to errors. The best way to cope with situations where there are assigned IP addresses or no assigned IP addresses is to rely on link-local addresses on their IP interface connected to the sharing-link, according to IPv4 link-local addresses [RFC 3927] and IPv6 link-local addresses (part of the IPv6 stateless adddress auto-configuration [RFC 4862]).

The sharing-link needs to provide high-speed network access so that it is able to carry the load of the information element redistribution and the signaling messages. The capacity of the sharing-link is not proportional to the number of participating nodes, but to the required throughput of the particular application. The dimensioning depends on the required application and is discussed in Section 7.4.2. The proposed use of WLAN suits most applications usable with Cooperative Internet Accessin resource constrained environments.

4.4.3 Finding Cooperative Internet AccessNodes and Services

Once the local nodes have setup the sharing-link (cf. Section 4.4.2) they can start to look for other Cooperative Internet Accessnodes being in physical proximity and the "services" they provide. The "services" refer to what type of application they are interested in to share the resources of their access-link.

The above process is commonly referred to as service discovery. The main difference is whether a centralized or a decentralized repository is used to store the information of services and to reply to query requests for the services. Service discovery has been specified in various protocols, such as, for instance, the Service Location Protocol (SLP) [RFC 2608] and Multicast DNS (mDNS) [mDNS]. SLP follows the centralized approach and mDNS follows the decentralized approach.

The centralized approach, shown in Figure 4.11, requires a server being located in the Internet and reachable by all nodes. This server will store information per current active Cooperative

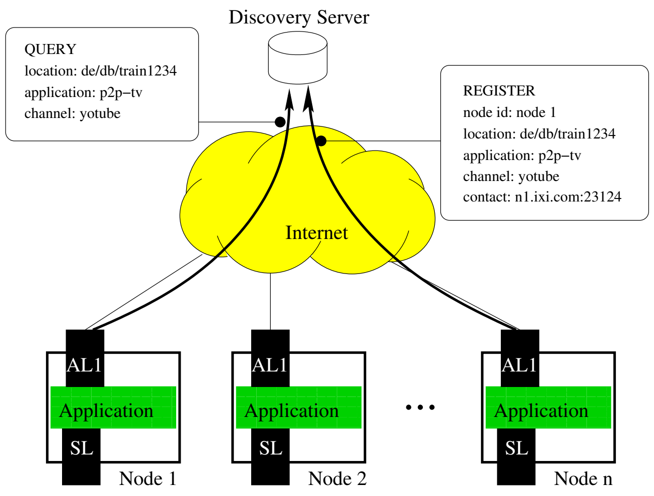


Figure 4.11: Server-based lookup for Cooperative Internet Accessnodes

Internet Accessnode, their location, used application, used resource (e. g. video channel), and how they can be reached by other applications. A Cooperative Internet Accessnode that searches for a Cooperative Internet Accessnode in its vicinity will query this server for neighboring nodes (shown as the QUERY in Figure 4.11).

The Cooperative Internet Accessnodes have to register the below type of information at the server, updating the information, and also removing the information:

node ID a unique identifier for a particular node that can be used as proof of ownership (if it is cryptographically safe, such as host IDs of the Host Identity Protocol (HIP) [RFC 4423]) for the information provided.

physical location the node has to provide an information about its current physical location. This information can, for instance, be based on coordinates obtained via the Global Positioning System (GPS), manually edited information by the user of the node, or other information sources such as RFID tags in trains indicating the current train number.

application the used application at the node, e. g., a peer-to-peer live streaming application.

channel the used channel of this application, e. g., what video channel being watched or what web site being visited.

contact the network contact information, i. e., IP address, transport protocol, and port number.

A Cooperative Internet Accessnode that searches other nodes in its vicinity will query the server with **location**, **application**, **channel**. The server will reply with a list of nodes for this query, if there is a matching entry, including the **contact** information for each node. The nodes will use the **contact** information to communicate with the other nodes.

However, the server-based approach has some severe drawbacks in the scope of Cooperative Internet Access. The server itself may be in need to be able to handled an enormous amount of requests, if the Cooperative Internet Accesssystem would find a wide audience. This may

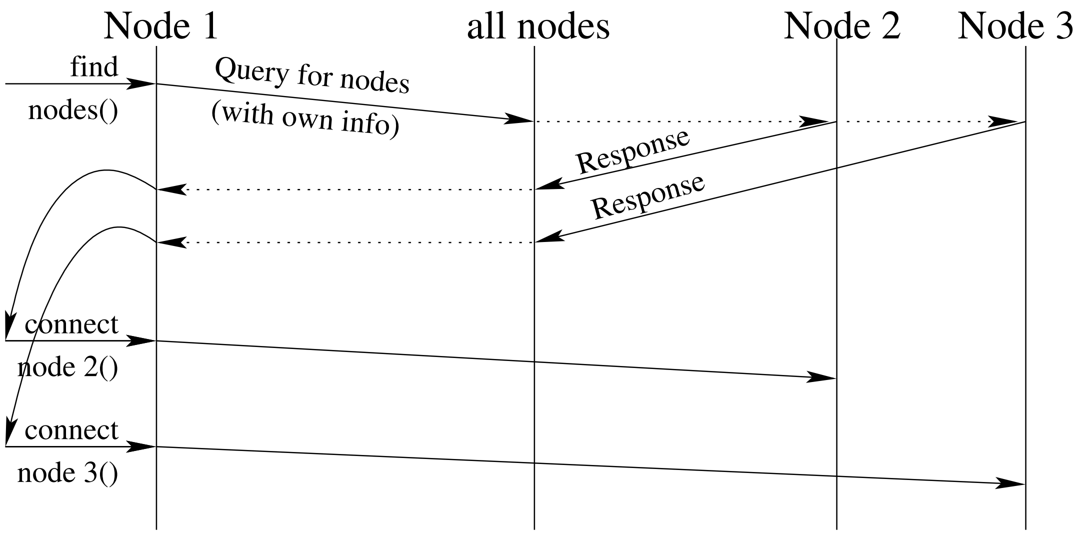


Figure 4.12: Link-Local Cooperative Internet Accessdiscovery flow chart

results in the need to deploy multiple servers with load balancing. Technically, this is not issue, but the operational costs would have to be covered by somebody. This seems to be unlikely for a system such as Cooperative Internet Access. Second, some of the information exposed to the server impose a threat to the user's privacy: the type of used application (e. g., file-sharing application), the used resource (e. g., filename, video channel), and the current physical location (which may include movement patterns). The location information provided by the system may not be accurate enough to judge if two nodes are in the same context (e. g., when do GPS coordinates match for the WLAN radio cell radius). There can be an ambiguousness of location information, i. e., how to represent any type of location, such as, sitting in a train in Germany, with train no. 1234. Another point is, if the information provided to the server is still valid at the time when it is being queried for, as nodes will move frequently or they are shut down by the user.

A decentralized approach would limit the information management (storing and querying) to the nodes in the local context only and not beyond this. The prerequisite for the decentralized approach is that the nodes have setup their sharing-link, as described in Section 4.4.2. Only nodes connected to the same sharing-link) should be able to receive and response messages. This ensures that only nodes being in physical proximity and thus in the same context, will be able to communicate. It may not beneficial to include nodes beyond the sharing-link, i. e., nodes that are one or multiple IP hops aways (via a router), as they are very likely not anymore in the same context.

Nodes connected to the particular sharing-link can either listen to broadcast messages sent or join a link-local multicast group to find other nodes. Broadcasting to all nodes on the link may be fine, as probably only the Cooperative Internet Accessnodes connect to this sharing-link, but to keep information distribution only relevant for participating nodes we prefer link-local multicast. With this multicast mechanism, only nodes that join a multi-cast group will receive and process the IP packets [RFC 1122].

Figure 4.12 shows the flow chart to discover Cooperative Internet Accessnodes in the same local context with 3 nodes (Internet Group Management Protocol (IGMP) [RFC 5186]) messages are omitted). Node 1 arrives and connects to the sharing-link and starts to send a query message to the a multicast address (”all nodes” in the figure) where all Cooperative Internet Accessnodes listen to. All nodes on the sharing-link reply with a response to the same multicast address, giving also other nodes the chance to learn about current set of active nodes and also to check if already known nodes are still alive. Node 1 can use the information provided in the responses to chose the appropriate nodes and to directly connect to them for the further operation, if needed.

The nodes have to provide this information in their responses:

contact the network contact information, i. e., IP address, transport protocol, and port number. This is syntactically the same information as provided by the nodes in the server case above, but semantically the can provide a different information, i. e., there is only the need to provide at least IP addresses that have a local significance, such as link-local IP addresses.

application the used application at the node, e. g., a peer-to-peer live streaming application.

channel the used channel of this application, e. g., what video channel being watched or what web site being visited.

One implementation choice for the decentralized approach is the Multicast DNS (mDNS) [mDNS] protocol. This protocol uses the link-local multicast address 224.0.0.251 out of the *Local Network Control Block* [RFC 5771]. Multicast DNS provides autoconfigured DNS in a link-local space, a service to resolve local host names, and a service discovery mechanism. Another implementation choice is the Message Bus protocol [RFC 3259] which is a link-local message oriented communication protocol.

The nodes can do not only need to discover other Cooperative Internet Accessnodes, but in fact do need to find nodes that offer the same application for the coooperative Internet access and use the same channel of that application (basically they need to find nodes using the same resource). This can be all done via the mDNS based discovery process. The nodes should also periodically query for other nodes to see if the nodes are still alive. Not every node has to send this query, as long as, it receives queries and replies from other nodes. This will limit the number of query messages to required minimum.

The more compelling approach for Cooperative Internet Accessis the decentralized approach, as this does not require a server to deployed and maintained in the Internet, it avoids the to far spreading of user private data (kept to local context for the decentralized approach vs. sending it to the server for the centralized approach). This approach ensures also by its link-local property, together with the sharing-link configuration, that only nodes in the same physical location and thus in the same context, can actually communicate with each other. Therefore, we always assume the decentralized approach to be used.

The next steps after other Cooperative Internet Accessnodes were located is to connect to them and to start the coordination and data exchange between the nodes.

4.4.4 System Coordination

The functional tasks of the controller process are described in Section 4.3.5 and cover the situation management, group management, collecting of link measurements, handling of information element requests, checking the status of the information elements, scheduling of the elements and finally the assignment of elements to the access-links. This section describes the implementation choices for the controller as system coordinator. The implementation options for the situation management are described in Section 4.4.1 and the group management is described in Section 4.4.3, as the are part of the overall system control and they are not repeated in this section.

The controller process requires to collect and process the information required to operate the Cooperative Internet Accesssystem. This can be done as at a central point or in a distributed way. However, it is impossible to rule out the centralized or decentralized design, as some applications may required the one or the other approach – this is highly dependent on the used application. For instance, the examined peer-to-peer video streaming application in Chapter 5 requires a centralized controller process, as the scheduling algorithm must be performed at a single node. The second examined application in this thesis, is cooperative web browsing in Chapter 6, and there the controller process is implemented in a distributed way.

However, each Cooperative Internet Accessnode has to implement the controller-process, independent of whether the centralized or decentralized approach is used: Even in the decentralized approach, every node can either become the host of the controller process, as the controller-process is elected. The nodes in the local system will select a node which hosts and runs the controller-process. There are such selection process in, for instance, IPv6 default router selection through router advertisements [RFC 2461], and super-peer selection in peer-to-peer networks [101].

Our selection process follows a simple principle that nodes connected to a power socket are preferred over battery operated nodes. In the case there are multiple nodes connected to a power socket, the node with larger memory is selected. In the case there is still a tie between at least 2 nodes, the node that arrived first in the system takes over. This requires to keep timestamps at the nodes and synchronized clocks.

We re-use the multicast-based group communication out of Section 4.4.5 for the communicating of the the control information. The information is sent to a multicast group, so that all nodes in the local context can receive the control information. This control information is:

- arriving nodes or gracefully leaving nodes (i. e., nodes being able to notify the other nodes about their leaving) in the system;
- link capacities information
- assignments of request queues
- status of request queues
- information element requests

The information and system state is usually stored in a single point when it comes to the centralized approach. But let's consider one system property of Cooperative Internet Access: Each node has the conroller-process logic implemented. Given that all nodes in the local context share the same set of control and state information we can define a failure procedure for the centralized approach. The nodes are required anyhow to periodically keep track of the participating nodes, as described in Section 4.4.3. The nodes restart the selection process, if the controller-process does not reply to two subsequently queries by other nodes and a direct message exchange with it.

4.4.5 Redistribution of Information Elements

Each node retrieves information elements via its access-link to its local element buffer. The retrieved information elements have to be shared with the other nodes in the local context via the sharing-link. This sharing is discussed in this section.

We have a set of nodes that participate in the same local context and that need to exchange the information elements. First of all, they have to announce the availability of a particular element and second they need to redistribute this element via the sharing-link.

One node could be assigned the role of an information element repository and distributor. This node would collect each information element retrieved by a node and would redistribute the element to the other nodes. Figure 4.13 shows this case where a controller node is collecting the information elements. Each node connects to the controller and uses this connection to exchange the information elements. Each node would have only a single connection to the controller which could also be reused for the control traffic. The advantage is that nodes joining later have a single contact point to retrieve data. The drawbacks are a single point of failure (if the node leaves the system), a potential bottleneck (if the machine is not that powerful), and that the information elements have to be transmitted multiple times within the system always crossing the sharing-link.

We assume n nodes in the system which retrieve i elements I_i and share it via the sharing-link. Each element is retrieved once. Each element will be transferred once to the controller and redistributed to all other nodes in need for this element. Each information element will be send $n-1$ times via sharing-link, i. .e., leading to a complexity of $O(n)$ for the information element exchange via the sharing-link. But a complexity of $O(1)$ for the number of transport connections, i. e., the number of transport connections for a node is independent of the number of nodes in the local context.

Another approach is a full-mesh between the participating node. A full-mesh requires each participating nodes to keep $\frac{n(n-1)}{2}$ number of transport connections to the other nodes, for bi-directional links. This results in a complexity of $O(n^2)$ in terms of transport links. The complexity of the information element exchange is the same as for the centralized approach, as every retrieved information element must be redistributed to all other nodes in the local context, i. e., the element must be $n-1$ times redistributed.

A mesh with a connection degree lower than full mesh, combined with the capability of each node to forward information elements to other nodes, would lower the complexity in terms of

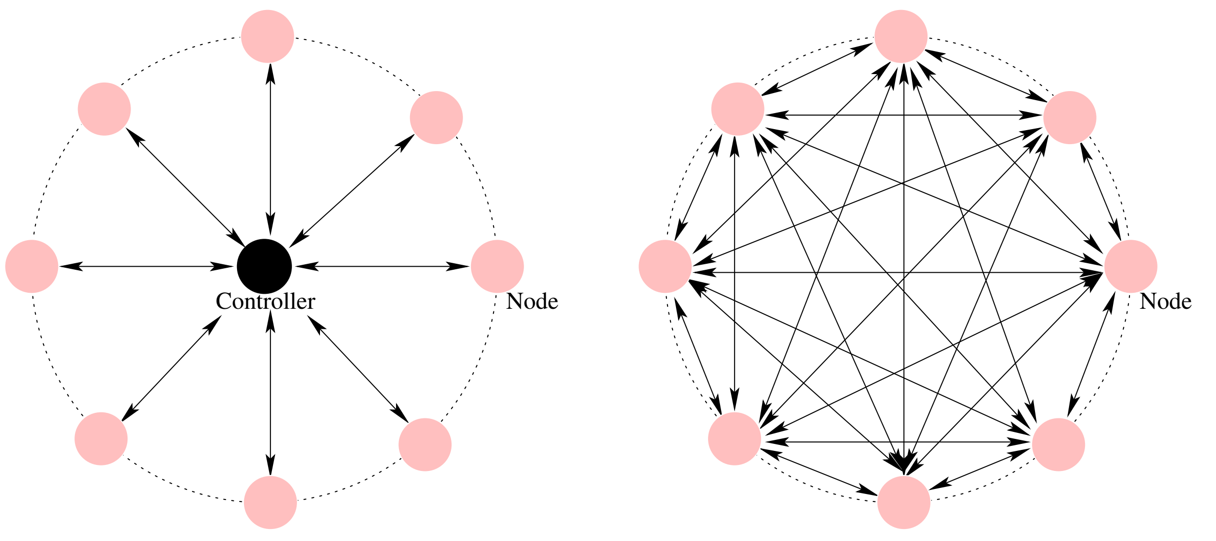


Figure 4.13: Information element redistribution on the sharing-link

required number of transport connections to other nodes. Nonetheless, the complexity in terms of transmitting the same information elements several times remains.

Multicast avoids the issues of sending the information element multiple times via the sharing-link. With multicast all nodes join a particular multicast group and send their data to this group. Each participating node receives packets send to this multicast group. Multicast faces deployment issues in large scale networks, but link-local multicast serves well in this case, as the multicast is distributed only on the local link and not beyond. The local link is in our case the sharing-link. Link-local multicast uses standard IP multicast [RFC 1122] with a multicast addresses out of the *Local Network Control Block* [RFC 5771].

With multicast, an information element and the information that did arrive is sent to a multicast address, where all participating nodes are listen to. However, information elements will exceed the maximum transport unit (MTU) for most common link-layer technologies. For instance, the MTU for Ethernet is 1500 byte [102] and an information element can have several kilobytes.

However, multicast is a unreliable transport mechanism, as the data sender has no control whether a multicast data packet made it to the intended receiver(s). The Reliable Multicast Transport (RMT) [RFC 2887] framework defines a reliable version of multicast and with the NACK-Oriented Reliable Multicast (NORM) Transport Protocol [RFC 5740]. a way to add reliability to multicast transmissions. The use of NORM ensures that each information elements is transmitted only once via the sharing-link, unless there are receivers that fail to received the full element. In this case, they need to retrieve the missing parts of the element by using the Negative-Acknowledgment (NACK) mechanism [RFC 5401].

Figure 4.14 shows a flow chart for a simplified RMT example. Node 1 retrieved an information element and is now ready to send this to the nodes (inf. elem. ready()). The information element is fragment to the multicast packets (pieces 1 to 3) and subsequently send to the multicast group address (dist. group). Node 3 receives all pieces, but node 2 misses piece 2 at t_1 and sends a

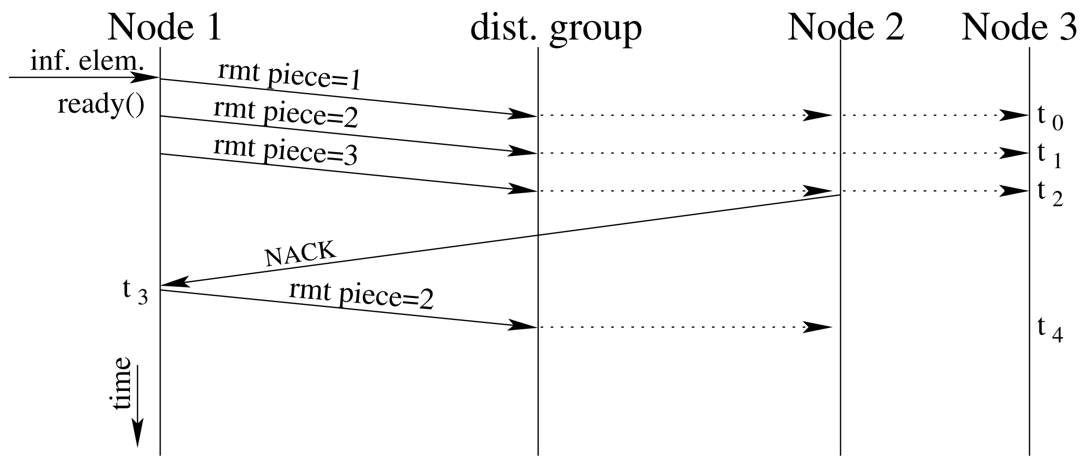


Figure 4.14: NORM flow diagram (simplified)

NACK to node 1 at t_2. Upon receiving the NACK at node 1 the piece 2 is resend (t_3) and finally received at node 2 at t_4.

The NORM based transmission scheme will limit transmission frequency of each information element, as the element is transferred only once in the ideal case of no packet loss. In the case of packet loss, only the lost packets will be retransmitted.

5 Peer-to-Peer Video Streaming in Constrained Environments

5.1 Challenges for Peer-to-Peer Video Streaming

This section and the following base on the description of peer-to-peer video streaming systems in Section 2.6.

5.1.1 Assumptions

This section sets the assumptions made throughout the remainder of this Chapter. The assumptions are necessary, as we want to focus the elaboration on effects caused by the Cooperative Internet Access system and not on general peer-to-peer system issues.

We assume that the local peer has sufficient remote peers and those remote peers are "able" to fulfill its chunk requests. We assume further the overall stem of the peer-to-peer stem is operating fine, i. e., the stem can distribute the content to the peers without issues. There is, for instance, no issue with the content source and the chunk dissemination.

The general assumption in peer-to-peer video streaming system design seems to be that the uplink of a peer's *access-link* is the limiting bottleneck, but that the downlink is able to handle the download of chunks, for instance, in PULSE [67]. In general, the limiting factor for a peer-to-peer system is the upload capacity of all participating peers, i. e., all peers can only have a cumulative download rate that is equal to the cumulative upload rate of all peers. However, the downlink may be already the bottleneck for some peers, as pointed out earlier in Section 3.

Breaking this down to a single peer that is situated in a resource constrained environment, we can define, analog to [103], that such a peer will operate either in an underload regime if $\forall t : \frac{\Theta_\ell(t)}{B_\mathrm{v}} < 1$ (less achievable bitrate than the video rate) or in a critical regime if $\forall t : \frac{\Theta_\ell(t)}{B_\mathrm{v}} = 1$. With the critical regime, the peer may still be able to obtain a sufficient number of chunks, so that the video can be displayed correctly. But in a underload regime, the peer will not be able to retrieve all chunks in time to display the video.

We further assume that the video is delivered with a video bitrate of B_v and that each generated chunk out of the video is of fixed length *ChunkSize*. The node can retrieve the chunks as long as $\forall t : \frac{\Theta_\ell(t)}{B_\mathrm{v}} \geq 1$ is true. The video chunks are generated at a fixed frequency f_C and with a fixed size *ChunkSize* and the video rate B_v:

$$f_\mathrm{C} = \frac{B_\mathrm{v}}{\mathit{ChunkSize}} \tag{5.1}$$

5.1.2 P2P Video Streaming in Resource Constrained Environments

Section 3.4 elaborates about the achievable throughput in UMTS deployments but the considerations in this section are in general true to all resource contraint environments, independent of wireless or fixed-line, moving users or non-moving users. Considering these measurements and the bandwidth requirements of the state of the art in peer-to-peer video streaming, we see that running peer-to-peer live streaming in such a resource constrained scenario is hardly possible or even impossible if the achievable throughput is below the minimum required throughput for longer times. The achievable throughput is not large enough to operate the peer in an at least critical regime, as most peer-to-peer systems required approximately 600 kbit/s to stream the video [5], as of today (2010); even lower video bitrates of 300 kbit/s (with a video quality) will be almost impossible to be streamed.

The peers will simply fail to download the chunks in a timely manner and consequently fail to display the video to the user, as there is not sufficient data. The peers will be able to joint the stream during periods of relative high throughput, e. g., in train stations, but in general the peers will not be able to stream the video.

The achievable throughput of a peer's *access-links* in fixed-line or stationary mobile wireless (either WLAN or UMTS) is rather stable, compared to moving mobile wireless. For stationary scenarios peer-to-peer video streaming systems consider several factors, e. g., throughput for data retrieved from a peer (cf. also Section 2.6) to decide from which remote peer a chunk is retrieved. These factors assume that the network environment is stable, i. e., the achievable throughput will not drop for a long period below the video rate or the throughput will not vary by one or two orders of magnitude. However, in wireless scenarios, the achievable throughput of an access-link is dropping far under the video rate or far above the the video rate at very short time scales. This is typically not considered by the peer-to-peer video streaming scheduling algorithms.

A resource contraint environment basically will not allow the peer to received sufficient data to display the video, as such a peer would have to buffer for a very long time to have a sufficient amount of chunks for the video playout. However, peers in peer-to-peer live video streaming system cannot simply wait for an arbitrary amount of time to fill their video buffer for the play-out. These systems have a reconnection threshold (e. g., T_D in PULSE [76]) which defines the maximum allowed node lag for a peer before the peer has to reset it chunk retrieval operations. A peer which lags for more than T_D behind the chunk source will not be able to retrieve any chunks anymore, as this threshold marks the time when all other peers will simply discard unused chunks from their chunk buffer. The threshold seems to be typically in the range of 0.5 to 2 minutes.

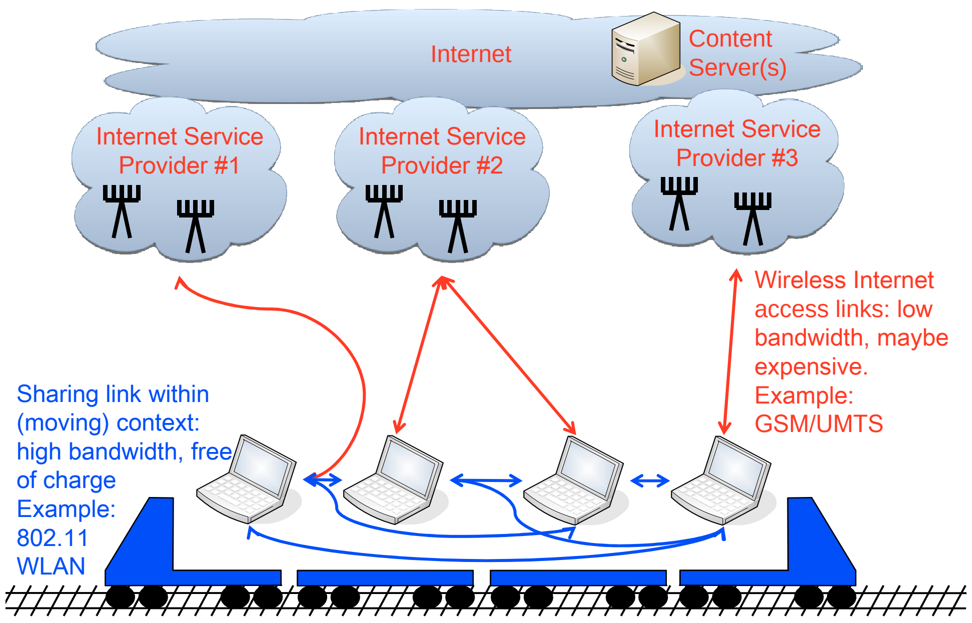

Internet
Content Server(s)
Internet Service Provider #1
Internet Service Provider #2
Internet Service Provider #3
Wireless Internet access links: low bandwidth, maybe expensive. Example: GSM/UMTS
Sharing link within (moving) context: high bandwidth, free of charge Example: 802.11 WLAN

Figure 5.1: Cooperative peer-to-peer video streaming use case

For nodes that have sufficient achievable throughput, the node lag is determined only by the dissemination time of a chunk from the source, via the peers, to a particular peer [104]. The chunk buffer is basically used to compensate the fluctuating dissemination times of the chunks. However, for peers with temporarily insufficient achievable bitrate the node lag will be slightly increased, while peers with constant insufficient achievable throughput will experience an ever increasing node lag, as they are unable to catch-up.

Technically, a once large node lag can be decreased again, if the achievable bitrate is greater than the video bitrate, but practically this is not possible, as the user expects the video to play in normal wall-clock speed. Catching up with the node lag would mean to play the video in fast forward until the video playback has caught up with the buffer node lag.

5.2 Cooperative P2P Video Streaming

The system we propose allows a group of L users in physical proximity (e. g., riding on the same train, see Figure 5.1) to access a live video stream in a joint effort.

We assume that every mobile device is equipped with two wireless interfaces: The so-called *access-link* (e. g., UMTS) provides Internet access. Its nominal bitrate $\hat{B}_\ell$ may be lower than the video stream's bitrate B_v. The actual achievable link throughput $\Theta_\ell(t)$ may change over time, i. e., $0 \leq \Theta_\ell(t) \leq \hat{B}_\ell$. The *sharing-link* (e. g., WLAN in ad-hoc mode) provides reliable high-speed communication between the nodes in the group free of charge, but no direct connection to the Internet.

The basic idea of the system is to download every chunk of the video stream over one of the access-links only once, and then redistribute it among the peers in the group using the sharing-link. The access-links are a precious resource and therefore their usage has to be coordinated *before* retrieval of a chunk is attempted. In this regard our system differs from other mesh-pull P2P streaming systems, which report the availability of chunks (e. g. by means of buffer maps) to their peers *after* they have been retrieved.

The instantaneous combined download capacity of the system is $\Theta(t) = \sum_{\ell=0}^{L-1} \Theta_\ell(t)$, and the system can only work stable if the average capacity is higher than the video bit rate: $\tilde{\Theta} \geq B_\text{v}$. Temporarily occurring drop outs have to be compensated by an reasonably dimensioned playout buffer, as in every P2P live streaming system.

The cooperative peer-to-peer video streaming uses the general Cooperative Internet Access architecture described in Section 4.2, relying on this to find other nodes, data exchange, etc; but requires extension to Cooperative Internet Access. For live streaming timely handling of chunks is the key: Chunks are generated at a fixed frequency (see Equation 5.1) and have to be available in the chunk buffer of a particular peer at the right time. This, together with the expectation that the peer's link will have an achievable bitrate lower than the video bitrate, calls for tight cooperation between participating peers.

The element that allows coordination between the peers is the buffer map. The buffer map lists all available and all missing chunks at a particular peer. The participating, local peers have to send their chunk maps to the controller process. Further, the peers have to send the the buffer maps of the active neighbor peers (i. e., the peers they have ongoing data exchange with) and passive neighbor peers (i. e. peers with no data transfer yet), to the controller process. The decision which chunks to fetch via the access-link is not made by the peer anymore, but by the controller-process. The peers refrain from requesting chunks from remote peers on their own, but delegate the chunk scheduling to the controller process. The controller process determines, based on the achievable throughput of a peer's link, the neighbors, and the available chunks of the neighbors, which local peer or local peers will be assigned to retrieve a particular chunk. However, the peers are free to decide on their own which chunks they upload to their remote peers. Once a chunk is retrieved, it is distributed to the other local peers via the sharing-link.

5.2.1 Scheduling the Chunks

The key to enable peer-to-peer video streaming in resource constrained environments is the coordinated retrieval of chunks by the participating local peers. The controller-process is coordinating the peers and is the focus point of the system. The controller process receives the following information per local peer and uses the information for the scheduling algorithm:

chunk map the current chunk map of a peer indicating the available and missing chunks. This information is sent periodically to see what chunks are missing in the system.

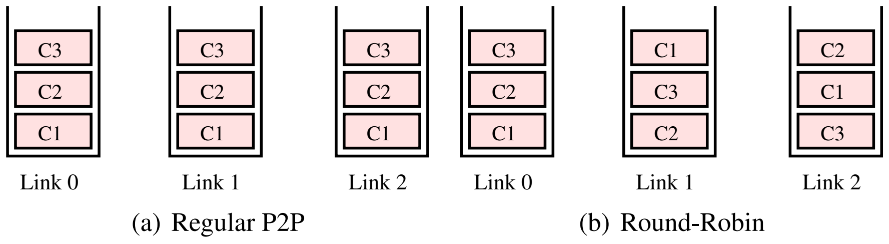


Figure 5.2: Chunk to link assignments

list of chunk maps of neighbors the chunk maps of the neighbors of that peer. This information is required to see if the local peer can actually retrieve a particular chunk from its neighbors, or if the neighbors are also lacking this specific chunk. It does not make sense to assign a chunk to a peer if there is no neighboring peer that can deliver that chunk.

achievable throughout The achievable throughput is measured by the peer per *access-link* and is needed by the controller process to judge the current state of the local peers and for the scheduler.

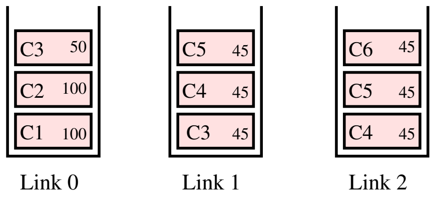
request queue For each pending request in the queue it reports several tuples *(point in time, likelyhood that chunk download will be finished by then)* to the controller.

The controller process uses this information in conjunction with the virtual chunk buffer (which implements also the trading window) to judge which chunks are the most urgent to be retrieved by the local peers. The missing chunks are sorted according to their urgency and stored in a First In – First Out (FIFO) list, so that urgent chunk are handled first by the scheduler.

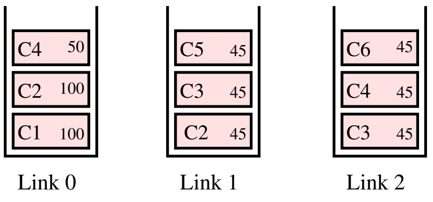
A first choice for the chunk to access-link scheduling, would be letting each link retrieve the same chunk at the same time, as shown in Figure 5.2(a). There are 3 access-links, *Link 0*, *Link 1*, and *Link 2* connected to local peers. The set of chunks to be retrieved is C_1, C_2, and C_3. This causes all links to be busy with a single chunk at the same time, probably downloading the same chunk 3 times, if all links deliver at the same throughput; or downloading no part of the chunk if all fail at the same time. As each link is a limited resource in terms of achievable throughput, this is a waste of resource and also neglects the fact that the links can be utilized in parallel to retrieve multiple chunks at the same time.

In fact, it would be better to ensure that a single chunk is not retrieved at the same time by multiple links, but possibly at different times, so that the set of currently retrieved chunks is diversified. This allows the scheduler to mitigate delivery failures of the required chunks, as a single access-link can:

1. fail completely, retrieving no chunks at all;
2. get very slow and deliver a chunk later as initially expected;
3. get very fast and deliver more chunks than initially expected;
4. fail at the same time where also other links fail (joint links).



(a) Estimation based



(b) Estimation based with sequential avoidance

Figure 5.3: Chunk to link with link probabilities

In the case that not all links retrieve the same chunk at the same time, the chances are either to have L-times different chunks, if all links deliver the chunk, instead of 1 but in multiple copies. If all links fail, no scheme will win. Chunks are typically assigned to multiple links, but at different ”times”, i. e., in different positions of the request queues, so that chunks will be retrieved with a certain probability by the local peers.

A strategy which is coping better with our scenario is a round-robin assignment, as depicted in Figure 5.2(b). Each link retrieves a different chunk, mitigating the shortcoming of the approach described above. Another scheduling approach would be the random assignment of chunks to access-links. However, round-robing and random do not consider the behavior of the links in the near future and is still prone to sudden link failures or drops of the throughput, as there is no redundancy of chunks. Let's, for instance, assume for chunk assignment made in Figure 5.2(b) that the achievable throughput of *Link 0* decreases to 0 while loading C_1. Let C_1 be an urgently needed chunk that is required to be downloaded, as otherwise the sliding window cannot progress. The other 2 links will keep downloading their chunks, but timewise the chunks will arrive in the chunk buffer in this order: $C_2, C_3, C_1, \ldots$. However, it may happen that link 2 is able to fetch C_3 but fail completely to retrieve more chunks. This will not change the retrieval order of the chunks, as Link 1 is retrieving the chunks in that order, but revert to the issues of retrieving all chunks via a single access-link.

Let's now assume that all links are disjoint in their behavior (e. g., each of them is from a different operator) and that we can estimate the achievable throughput and the probability to reach that throughput value for each link for a limited time in the future. Let's further assume that *Link 0* can deliver 2 chunks with a delivery probability of 100 % and the third chunk with only 50 % probability. *Link 1* and *Link 2* can deliver any chunk with a probability of 45 %. Using this knowledge, the chunks can be assigned as shown in Figure 5.3(a) with the right hand side values in the chunk boxes indicating the probability of a chunk being retrieved via the link. Each chunk is assigned to one or more links, until the summarized delivery probability of a chunk reaches a threshold value (here in this example it is 90 %). This causes that C_1 and C_2 are assigned to *Link 0* only, but all other chunks are assigned to multiple links. This allows to retrieve more chunks successfully while obeying the time critically of the peer-to-peer video streaming, as compared to the prior chunk to link assignments, as the scheduler can assign chunks to access-links according to the probability that a link is delivering a chunk.

The chunk to link assignment shown in Figure 5.3(a) has the drawback that C_1 and C_2 are lost, if the prediction of the chunk delivery probability is wrong. Let assume that *Link 0* can retrieve C_1 as indicate by the probability estimation, but that the estimation for the next chunks (here C_2 and C_3 are wrong – in the worst case they will not be delivered at all). C_3 can still be delivered via *Link 1* with a success probability of 45 %, but chunk C_2 will be lost for sure. This can be circumvented if a direct subsequent chunk is not only assigned to the same link, or is not assigned to joint links. Joint links will react similar and thus subsequent links assigned to them may be delivered too late or not at all. However, it might be impossible to fulfill this precondition in a real scenario, as only joint links may be up and running, while all other links are down.

It is also important to note that the FEC allows to skip certain chunks [105], but it also important to retrieve subsequent chunks as much as possible. However, retrieving subsequent chunks via one access-link may be difficulty as a link failure on this link may impact a number of subsequent chunks which are important to have for the FEC. In ADSL networks, a technique called *interleaving* is used to mitigate the impact of link issues to the delivery of subsequent cells. Instead of sending cells in their strict order, they are reorder (interleaved) to mitigate line distortions. We apply the *interleaving* technique but not on a single access-link, but across multiple access-link to improve the system resilience against chunk losses.

Figure 5.3(b) shows the chunk to link assignment where a subsequent chunk is not assigned to the same link in the first place, but probably at later stages. A chunk is assigned to an access-link and the next chunk is assigned to the following link. This will result in the assignment of C_1, C_2, and C_3 to *Link 0*, *Link 1*, and *Link 2*, respectively. This avoids subsequent chunks to be assigned to the same link. However, in a second run C_2 is assigned to *Link 0* as the retrieval success probability A_c is below the threshold of 90 %. The assignment will ensure that most of the chunks are retrieved by the system, given that the links are disjoint in their behavior. This result of chunk to link assignment follows the later described algorithm 1.

5.2.2 Deadline Scheduler

Regular chunk schedulers of peer-to-peer video streaming systems assume that each peers runs at least with a *critical regime* [103], i. e., $\forall t : \frac{\Theta_\ell(t)}{B_v} \geq 1$. However, we consider scenarios where the system will need to operate with low capacity $\frac{\Theta(t)}{B_v} < 1$ (for a short time) and also very high capacity $\frac{\Theta(t)}{B_v} \gg 1$ and with frequent transition between these extremes. The most challenging part is the coordination of chunk retrieval among the multiple links. Therefore, we propose a new algorithm that combines deadline-based chunk scheduling, throughput estimation for the *access-link* and a bin-packing algorithm with First-Fit-Decreasing (FFD) (see Algorithm 1). First-Fit-Decreasing refers to that chunks are assigned to links that can deliver within the chunk's delivery deadline (see Equation 5.2) with a high likelyhood first; Further chunks assignments will use a decreased level of likelyhood of delivery.

A regular peer-to-peer video streaming scheduler (as discussed in Section 2.6.5) can use the order of the chunks to determine which chunks are retrieved first. For instance, all chunks in the sliding window are retrieved in a sequential order while the remaining chunks in the trading window are retrieved following a random order or Least Recently Used (LRU) order. For peer-to-peer system running in non resource contrainst environments, retrieving chunks following LRU is to be preferred, as chunks not that distributed within the system have to be prioritized in the distribution above other more distributed chunks. However, in our scenario, throughput is a scarce resource and the peers will need to retrieve chunks according their priority needed for the local system. A first natural choice to define a priority is the order of the chunks, i e., chunks with a lower chunk ID have a higher priority over chunks with higher chunk IDs (increase chunk ID implies a decrease in retrieval priority). This does not give any guidance when a particular chunk is actually needed in the local system, but just the order. Furthermore, future peer-to-peer video streaming systems may use variable sized chunks where some chunks are smaller and can probably retrieved much faster than other chunks. This would also not be considered by solely considering the order of the chunks.

To give the Cooperative Internet Access system the ability to determine at which point of time a chunk is needed by the system and also to correlate the chunk's delivery deadline with the achievable throughput of the access-links (in conjunction with the chunk size), we use a deadline-based chunk scheduler. The scheduler calculates a chunk delivery deadline d_c for each chunk C out of the current trading window. This deadline is determined by the current lower window edge C_ρ of the sliding window and the current chunk C in Equation 5.2.

$$d_c = \frac{\text{ChunkSize}}{B_\text{v}} \cdot \left(C - C_\rho\right) + t_\text{absolute} \tag{5.2}$$

The algorithm considers the list of still missing chunk IDs C, which are sorted according to the chunk's respective playout deadlines d_c. Each chunk is assigned to one or several chunk retrieval request queues Q_ℓ, which correspond to the set of active links L, in order to assure a sufficiently high probability of successful download. The algorithm runs R rounds (line 1) and in each round each request queue is assigned another chunk request. L' denotes the links that have not yet been assigned a chunk request in this round (3). We search for the most urgent chunk which has not yet been assigned often enough for a high success probability (6), and look for candidate links L'' that do not yet have this chunk in their request queue (11). If a specific chunk already has been assigned to all links but the estimated download success probability is still below the threshold, there is not much we can do; we just continue with the next chunk (15) and the video will probably stall. If we find one or several candidate links we search for the one ℓ' that can retrieve the chunk with the highest success probability $P_\ell(d_c)$ within the chunk's respective deadline d_c (18), and append the considered chunk ID to this link's request queue (23). We update the chunk's aggregate retrieval success probability A_c (24) in order to determine whether the chunk should be assigned to more other links. Finally, the next chunk is picked for assigning it to a request queue (26).

The algorithm is performed up to R-times, where $R = \frac{B_v}{ChunkSize} \cdot \tau$ denotes the maximum length of each queue Q_ℓ. The parameter τ denotes the scheduling epoch of a peer-to-peer node, i. e., the time after a node evaluates the overall system behavior and the behavior of the remote peers. The PULSE system uses an epoch of 2 s, and we use an epoch value of $\tau = 1s$ to account the more volatile environment we are operating in. The epoch τ is also used later on as the forecast period of the achievable throughput estimator in Section 5.3.2.

Chunks that have not been retrieved by any peer, so-called missed chunks, will be detected by the scheduler once they will not be available in the virtual chunk buffer by their delivery deadline. This may cause the scheduler to re-assign these chunks at the time of their actual delivery which in turn may cause the video to stall. However, in the meanwhile, the arrival

```
Input: $C$ (sorted list of missing chunks)
       $L$ (list of links)
Output: $Q$ (request queues)
for $r := 1 \ldots R$ do
    $c := 1$;
    $L' := L$;
    while $|L'| > 0$ do
        repeat
            while $A_c \geq threshold$ do
                $c := c + 1$;
            end
            $L'' := L'$;
            foreach $\ell \in L''$ do
                if $c \in Q_\ell$ then
                    $L'' := L'' \setminus \{\ell\}$;
                end
            end
        until $L'' \neq \{\}$ ;
        $p := 0$;
        foreach $\ell \in L''$ do
            if $P_\ell(d_c) > p$ then
                $p := P_\ell(d_c)$;
                $\ell' := \ell$;
            end
        end
        $Q_{\ell'} := Q_{\ell'} \cup \{C_c\}$;
        $A_c := A_c + p - A_c \cdot p$;
        $L' := L' \setminus \{\ell'\}$;
        $c := c + 1$;
    end
end
```

Algorithm 1: Chunk retrieval scheduler

of other chunks in the virtual chunk buffer, may make the need for these chunks obsolete. If Forward Error Correction (FEC) is used, some chunks may not be needed anymore. However, the decision whether a chunk is needed or not is made outside the scheduler in the regular peer-to-peer system. A chunk marked as handled by the FEC will not be re-assigned by the scheduler anymore but skipped.

5.2.3 Working Points of the Scheduler

This section discusses the various working points of the scheduler.

Let's assume $\tilde{\Theta} \ll B_{\mathrm{v}}$, i. e., the instantaneous combined download capacity of the system drops far below the video rate. In such situation, none of the access-links may be able to deliver any chunk within the delivery deadline or only a few chunks with a deadline far in the future may be delivered. However, this will basically cause the trading window to stall as no new chunk can be downloaded within time. The Algorithm 1 will basically fail to find any suitable link in line 18, as the likelihood to deliver a chunk within the deadline will always be zero: $P_\ell(d_c) = 0$. In this case, the scheduler assigns the chunks in descending order of their urgency to the access-links in round-robin.

Let's assume that all participating access-links provide a high throughout $\tilde{\Theta} \gg B_{\mathrm{v}}$, such that each access-link is able to retrieve the chunks on its own ($\Theta_\ell(t) \geq B_{\mathrm{v}}$). This condition can indicate that it is not required to run Cooperative Internet Access anymore, as every peer has sufficient throughput to retrieve the video, if it last for a longer period (where this period can be dependent on the scenario). For instance, considering the train scenario, shown in Figure 5.1, this can happen if the train stops in a train station where all mobiles experience a good network coverage and where high-speed UMTS access with HSDPA is available. The chunk scheduling result and assigned request queues to each access-link of the Algorithm 1 depends on the number of access-links c and the number of chunks $|C|$ due in a scheduling round. The following assumes that any of the links can deliver the chunks within the delivery deadline:

$|C| > |L|$ There are more chunks than links, causing that at all links get at least one chunk assigned and depending on the ratio of $\frac{|C|}{|L|}$ more chunks. For instance, a ratio of 2 indicates that each link gets 2 chunks assigned.

$|C| = |L|$ There are as many chunks as links, causing that each link gets one chunk assigned.

$|C| < |L|$ There are fewer chunks than links, causing some links have no chunk assigned. These links will fall idle, as they have no chunks to be retrieved. For instance, a ratio of $\frac{|C|}{|L|} = 0.5$ indicates that only half of the links get one chunk assigned.

Let's now assume that we have the situation that almost all access-links are slow, but that one access-link is very fast $\Theta_{\ell=x}(t) \geq B_{\mathrm{v}}$. This will result in $\sum_{\ell=0}^{L-1} \Theta_\ell(t) \approx \Theta_{\ell=x}$, causing that all links may get assigned a chunk, but in fact all chunks are assigned to the fastest link and finally also retrieved via this link. Links that are to slow, will probably not be assigned a chunk and fall into probing. However, this situation has the drawback that all chunks are mainly assigned and retrieved via the fastest link. If this particular link suddenly decreases the achievable throughput massively (e. g., below the video rate) the chunks will not be retrieved and they need to be urgently rescheduled to other access-links.

Considerations for chunks: It may happen that a chunk is missed out during one runtime of the Algorithm 1, as the algorithm cannot find a sufficient amount of links to handled a particular chunk. Given that the chunk has not already been retrieved in the meanwhile, the algorithm will re-consider the chunk in the next runtime and search again a fitting link, probably adding a new link and thus adding also to the chunk's aggregate retrieval success probability A_c.

5.3 System Design Considerations

The prior section described the principles of chunk scheduling in resource constrained environments while the is section details the implementation of the whole Cooperative Internet Access system for peer-to-peer video streaming.

5.3.1 Representation of Access-Links

Each participating peer must at least provide one access-link to the local system but it can provide more than one access-link if available at this peer. Each access-link is passively monitored to record the achievable throughput, as input to the throughput estimator described in the following Section 5.3.2.

An access-link can take 3 link states:

Up The link is providing achievable throughput, either by downloading or uploading a chunk for the local system or by other traffic originated at the particular peer.

Down The link is down and not providing achievable throughput to the local system.

Probing The link is **Up** but the measurements have not yet collected sufficient data to judge about the access-link's behavior for the throughput estimator.

Idle The link is **Up**, but actually not downloading or uploading anything, the throughput is zero. However, the throughput estimator is still able to estimate the throughput. A link may fall back to **Probing** if it is not used by any traffic for a certain time.

The above link states are used throughout the remainder of this section.

5.3.2 Estimating Throughput

The download predictor module in each peer uses the input of the passive measurements of the achievable throughput of each access-link. These measurements are used to predict the near-future behavior of the access-link. For all requests in the queue it reports samples of the probability distribution function for the probability of a successful chunk retrieval till a given deadline to the central controller. The controller uses this knowledge to assign new requests to the peers.

An earlier empirical study [106] proposed to use a Global Positioning System (GPS) based approach relying on past throughput recordings stored in a back-end database, to estimate throughput. We propose a no-reference throughput estimation scheme without depending on GPS and past recordings, but relying solely on current measurements. This takes away the requirement for each peer to supply a GPS module and also the need for somebody to operate and maintain a backend database.

We use a Simple Moving Average (SMA) as basis for our estimate and extend it as described in Formula 5.3 to calculate the future achievable throughput $bw(t_i + \tau)$ based on the current measured values and the average $\tilde{bw}_{i,j}$. The SMA as such is not sufficient to estimate throughput. The SMA takes n past values of a sample interval t_i, t_j where $t_i > t_j$ is the time period spanning the complete moving average window.

$$bw(t_i + \tau) = \tilde{bw}_{i,j} + \frac{bw(t_i) - bw(t_j)}{t_i - t_j} \cdot \tau \tag{5.3}$$

We allow each peer to measure with a certain degree of fluctuation in time, i.e., $t_i - t_j$, but restrict the estimation to a fixed step τ in the future. This relaxes the measurement timing at each peer, as it is not necessary to measure exactly every τ.

The throughput estimator results for 2 operators based on 4 measurement runs each, is shown in Figure 5.4. The figure shows the probability P_{bw} that the estimation $bw(t_i + \tau)$ is matching the actual bandwidth, expressed as a function of the estimation precision δ_e. The estimation precision δ_e denotes how much (in per cent) the estimated throughput can vary from the actual measured bandwidth. P_{bw} is also shown for a step rate $n = 10$, showing that a step rate $n \gg 2$ is not beneficial.

The throughput estimation can achieve an precision of 45 % of the actual occurring throughput, given a δ_e of 15 % which is sufficient enough for the chunk scheduling. Larger values of δ allow higher hit rates but at the cost that the throughput estimation is diverging heavily from the actual throughput.

5.3.3 Chunk Size Considerations

The chunk size and thus the likelyhood that a chunk can be transmitted within the throughput estimation period τ is of major importance for the system's operation. Chunk sizes resulting in longer than τ seconds transmission in average will not be covered by the throughput estimation. We used the measurement data to determine the probability of chunk delivery within τ as a parameter of the chunk size, as shown in Figure 5.5.

Very small chunk sizes ($< 10 kbyte$) will allow the system to always deliver chunks within τ, but will increase the size of the buffer maps. Reasonable chunk sizes are the range of 10 to 20 kbyte, as at least 70 % of all chunks are delivered within τ. However, there may be peer-to-peer

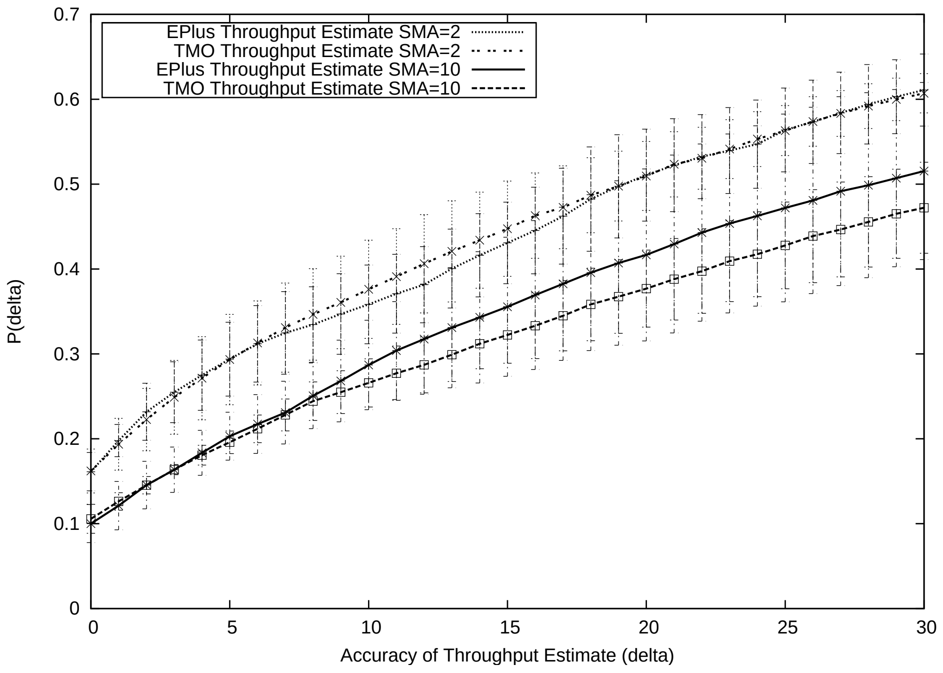


Figure 5.4: Probability of Throughput Estimation

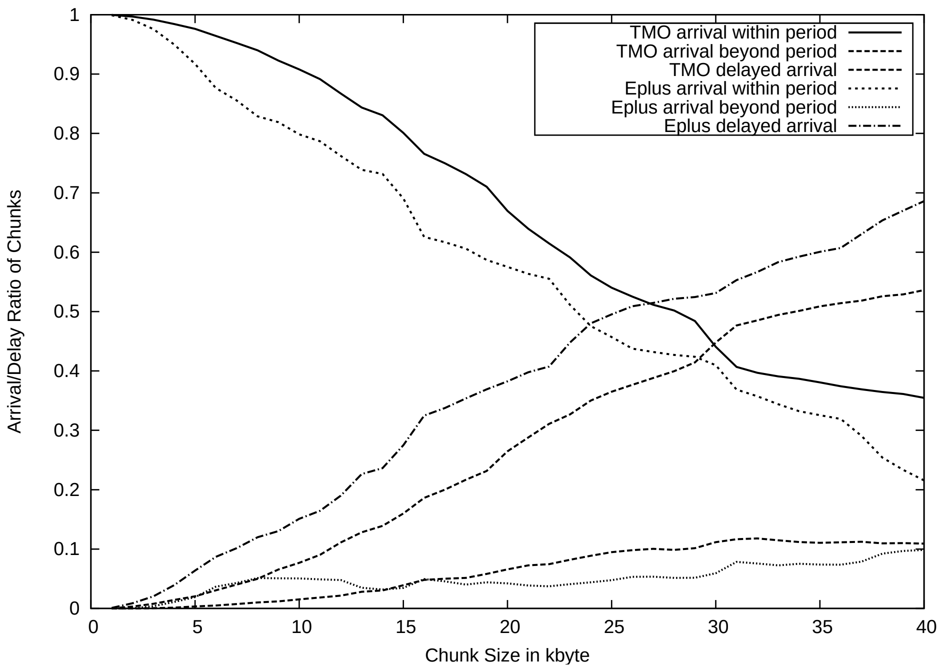


Figure 5.5: Probability of Chunk Delivery within τ

video streaming systems that operate with rather large chunks sizes and also varying chunk size depending on the used video rate (see PPSP measurement study [77] that indicates chunk sizes greater than 14 kbytes). Figure 5.5 suggests that chunks that exceed 40 kbytes of size will make it very unlikely to be delivered within 1 second to a peer, i e., larger chunk sizes will not allow the system to retrieve the chunk in a reasonable time for video streaming. However, assuming much larger chunk sizes, we peer-to-peer video streaming systems could re-use the concept of

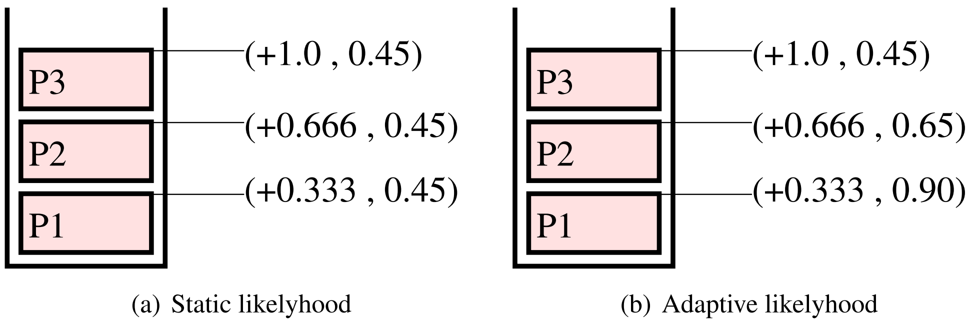


Figure 5.6: Situation Reports with estimation of 300 kbit/s and chunksize of 12.5 kbyte; tuple (+0.3333, 0.45) denotes delivery deadline in +0.333 s with likelyhood of 45 %

sub-pieces in Bittorrent [107]. A sub-piece is fraction of a chunk that can be identified. A sub-piece could be again in the range of 16 kbyte, cf. [107].

5.3.4 Estimator to Controller Communication

Section 5.3.2 describes how the system estimates the future achievable throughput. This estimation process is handled locally by each particular peer and each peer announces only the result to the controller. It is not sufficient for a peer to announce the current and estimated achievable throughput to the controller process, as the controller process has no further indication on how good the estimated actually is.

The peer uses the entries of the request queue to announce the likelyhood to deliver chunks within time boundaries. For each access-link it can announce one or multiple estimations for the achievable throughput. For each queue entry, the relative delivery time and a likelyhood that the chunk will be delivered until by then, is stated, as shown in Figure 5.6.

The peer uses the measured achievable throughput and calculates the throughput estimation for the next τ s, as described in Section 5.3.2. The peer uses the likelihood of the throughput estimation according to Figure 5.4 to determine the estimated time of arrival (ETA) of a chunk and the probability of this forecast. Let's assume that the throughput estimation $B(x)$ with $x \subset t$ results in an estimated retrieval time for a chunk of $t_{retrieve}(x) = 0.290s$. The probability of the estimated throughput is for a $\delta_e = \pm 0\%$ equal to 10 %, according to Figure 5.4. However, for $\delta_e = \pm 15\%$ the probability is up to 45 %, which is used in the example to set the estimated arrival time of the first chunk to $ETA = t_{retrieve}(x) \cdot (1 + \delta_e)$ which turns for our example to $ETA = 0.290 \cdot (1 + 0.15) = 0.333s$.

This value is used to determine the estimated time of arrival for the chunks in Figure 5.6(a). The first chunk can be delivered +0.333 s from now, the next chunk in +0.666 s from now on and the 3rd chunk in +1.0 s from now on. The real reporting uses absolute times, but we used relative times to ease the explanation.

The reporting scheme in Figure 5.6(a) assumes that the throughput estimation's likelihood is equal of the whole time period of the forecast. However, there may be future forecast schemes which allow an adaptive likelihood over the forecast period, as shown in Figure 5.6(b), where estimations closer in time are rated with a larger likelihood as estimations more far ahead.

5.3.5 Cooperative Internet Access Controller Process

The key element for the peer-to-peer video streaming case is the controller process. This section describes the data structures and control flows of the controller-process.

5.3.5.1 Data Structures

The first data structure managed by the controller process is the list of participating peers (*peer_list*) in the local system. This peer_list is used to manage the overall system in terms of peers that arrive or gracefully leave, and also to detect dead peers that left the system without notice. It contains the IP contact information (i. e., the IP address version, the IP address, the used transport protocol, and the port number of that transport protocol).

A second list is maintained to keep track of the access-links provided by the peers (*ac_link_list*). Each peer must actually contribute a single access-link but it can also contribute multiple access-links, if the peer itself is multi-homed. The ac_link_list keeps track of the current status of the link, as reported later on in the Situation Reports, the general link status (Down, Up, Probing, Idle), and the currently assigned Request Queue.

The controller has to keep track of the system's virtual chunk buffer. The virtual chunk buffer (*v_chunk_buffer*)is the set union of all peer's chunk buffers in the local system. However, the controller process does not need to keep the chunks, but has to keep track of the available chunks in the local system. The controller process uses the virtual chunk buffer to judge what chunks are available at the remote peers, what chunks are missing in the local system, which chunks have to be considered for downloading, the downloading deadlines of each chunk, and the management of the trading window, i. e., when to move forward with the window. It also keeps track per chunk which nodes have already received a particular chunk (see also Section 5.3.5.3).

The chunk request list (*rq_list*) stores the current outstanding chunk requests delegated to the participating peers. For each chunk request the list states the download deadline (value is copied from v_chunk_buffer), the peers that should download the chunk together with the position in the peer's request queue, the particular download success probability for each involved access-link and the combined likelyhood A_c for this chunk. The particular download success probability for each involved access-link is used to recalculate the combined likelyhood A_c, if an access-link disappears from the system and also if a link is added.

5.3.5.2 Control-Process Control Flow

This section outlines the control flow of the controll-process which includes also the execution of the scheduling algorithm described in Section 5.2.2. It is event driven and reacts to events caused by the peers and also caused locally at the controller-process.

The control flow of the controller process is shown in Figure 5.7. The process is suspended after the initialization phase and waits for any event to happen. Not all events are handled immediately, but queued for a limited time to accommodate several events in one run instead of running multiple times for each single event. This is necessary, as the peers work asynchronously, e. g., one peer may received a new request queue to work on while another peer is reporting the arrival of a new chunk while another peer is reporting a severe change on its access-link. The events are queued unless stated otherwise in the below description.

We define a hold timer T_W for which events may be queued at maximum before they are processed. The time T_W depends on the chunk generation frequency: $T_W = \frac{1}{f_c}$. We use the chunk generation frequency as event hold time, as this frequency gives a fixed handle to deal with the system dynamics. The other parameters, such as, arrival rate of chunks, change of conditions on the access-links, can exhibit a varying level of dynamics. For instance, at some times chunks may arrive very fast (large achievable throughput) while at other times chunks may arrive slow or not all(low or now achievable throughput). The scheduler needs to perform its tasks, independently of these circumstances at a certain pace.

The controller process waits for any event at **"wait for event"** in Figure 5.7 to occur after the initialization of the process. The events and their handling are:

New Peer An arriving peer adds at least 1 new access-link to the system. Any new access-linkis marked as *Probing* and the scheduler assigns a number of urgent chunks to the link's request queue. The success probability A_C of each assigned chunk is not changed, as a probing link does not provide any such information. The links in probing uses the assigned chunks to enter the peer-to-peer system and to start the measurement on the access-link which in turn enables the particular peer to estimate the future link performance. The scheduler reacts immediately to this type of event, so that any new peer can be used for the system as soon as possible, i. e., the controller does not queue this event.

Dead Peer A leaving peer causes the scheduler to reconsider the chunks that had been assigned to the request queue of the dead peer. The scheduler has to update the rq_list in that it segregates the success probability of the access-link of the dead peer for the chunks that have been part of the access-link's request queue. The scheduler reacts immediately to this type of event and runs Algorithm 1 to reschedule the chunks to any other access-link. Dead peers either announce their leave from the system with a message (graceful leave) or in the case of disappearing peers, a dead peer watchdog timer notifies the controller. The dead peer watch dog timer watches if local peers react in a timely manner to request messages sent by the controller process. The dead peer detection and timer handling is omitted in Figure 5.7 and detailed in Section 5.3.7.

Buffer Map Update A buffer map update (or multiple of them) will give information to the scheduler which chunks are now available at the remote peers. This information is used to determine which chunks can now be retrieved. The scheduler will run Algorithm 1 if the update indicates that there a new chunks needed by the local peers, after the expiration of T_W. The new chunks are added to the sorted list C of missing chunks. It may be the case that the update received by one peer is not valuable to the local system, as other remote peers already have indicated the presence of such chunks and the local system is

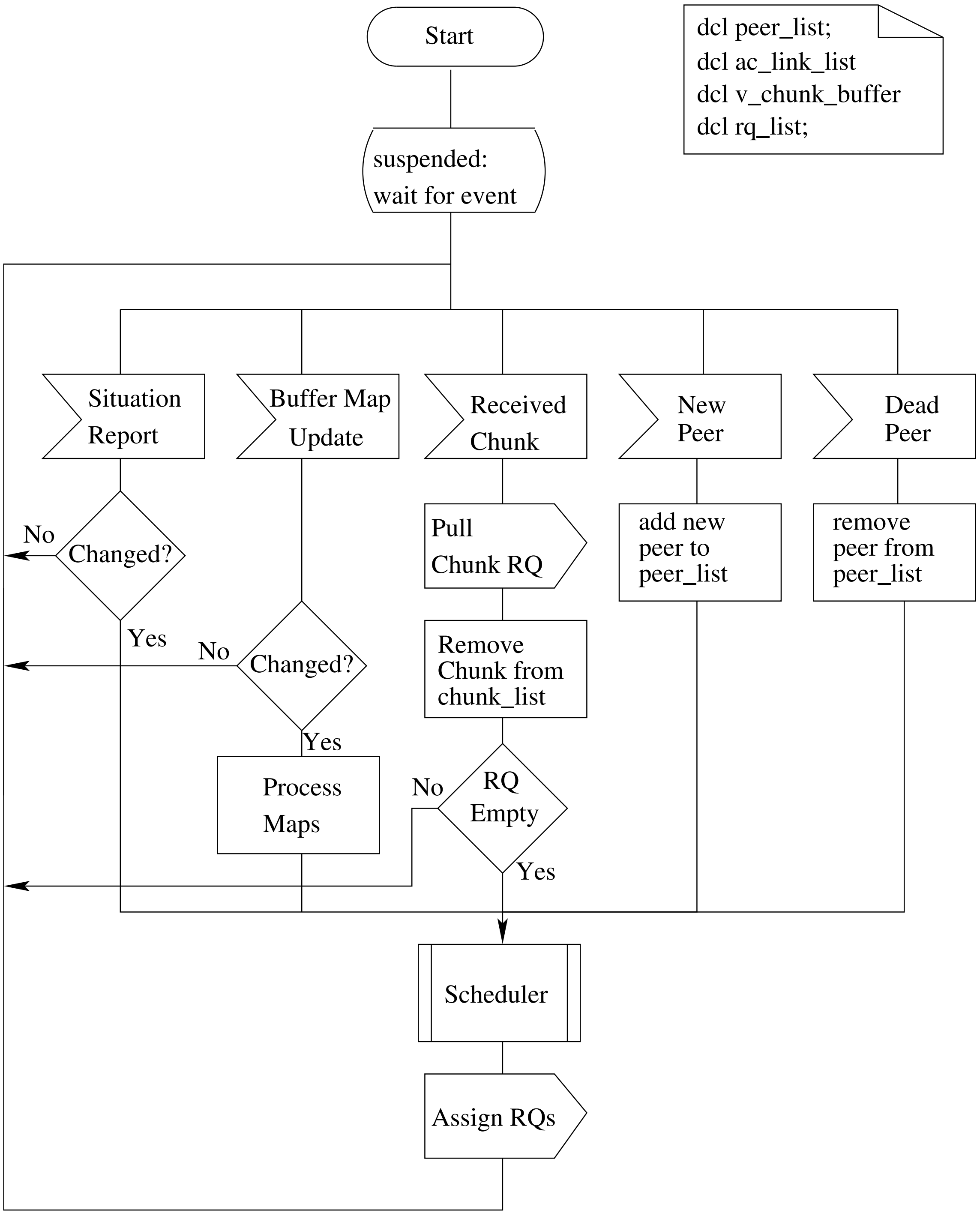


Figure 5.7: Control flow of the controller-process

already retrieving the chunks. However, this decision step is already taken in the overall control flow, as shown in Figure 5.7.

Explicit Situation Report Each local peer periodically sends a Situation Report to the controller to update the controller about the status of the request queue and to show also its liveliness. Such a report must also be sent if there is a severe change in the link's situation, If there is no change, or only a slight change, the link statistics are updated but the scheduler is not evoked – *Changed?* in the figure. In the case that the link's achievable throughput is much faster than originally envisioned, there will be no explicit Situation Report, as the Received Chunk message will carry that information anyhow. Only if the link's achievable throughput is slower than estimated a Situation Report is issued. In the case the achievable throughput decreases, as Situation Report is only issued if any of the chunks in the link's request queue will miss its delivery deadline. The scheduler has to update the rq_list in that it segregates the success probability of the access-link of the peer for the chunks that have been part of the access-link's request queue. Afterwards, the scheduler runs Algorithm 1 to reschedule the chunks to any other access-link, after the expiration of T_W.

Implicit Situation Report An Implicit Situation Report is sent piggy backed with the Received Chunk message and updates the link statistics. This report updates controller about the current status of the request queue.

Received Chunk The controller pulls a received chunk from all other peers' request queue with the message *Pull Chunk RQ* sent to all peers which have this particular chunk in their request queue, so that this particular chunk is not pulled again. The received chunk is marked as available in the v_chunk_list and it is also checked if the trading window can move on by one or multiple chunks.

A received chunk will only cause the scheduler to be evoked, if any of request queues of one of the the access-links runs empty, shown as *RQ Empty* in the figure. The access-links with an empty request queue are added to list of links L and the Algorithm 1 is called to load the links' request queue In the case that there is no need to schedule chunks, the particular access-link are marked as Idle.

The assignment of chunks after the scheduler is done with the *Assign RQ* message sent to the peers. A single message is sent to all peer and carries the request queues for particular peers which are about to receive a new queue assignment. The particular peers can be all peers or a subset of them.

5.3.5.3 Trading Window Synchronization

The controller process, as well as the the local peers, have to keep their trading window synchronized, as they need to have the same state about the beginning and the end of the trading window and also about which chunks are already available at one of the local peers. This synchronization is necessary, as peers may loose synchronization with the local system's state and it will happen that a peer joins the system later on. In any case, this peer has synchronize its trading window with the system's state, including the list of already retrieved chunks.

The controller process announces together with the *Assign RQs* message the current state of the virtual chunk buffer, which contains the information of the trading window position and also which chunks are already available and at what peer. A peer checks the *Assign RQs* message for its request queue and also for the virtual chunk buffer. In the case it detects that the virtual chunk buffer lists a chunk that is still missing in its chunk buffer, the peer contacts one of the peers with this particular chunk directly and downloads it.

5.3.6 Cooperative Internet Access Peer

This section discusses the changes required at a mesh-pull based peer-to-peer video streaming peer to enable the usage of Cooperative Internet Access. We use the PULSE [76] system as the reference for this section, as it is well documented and accessible by the general public. However, the guidance of this section is valid also for other, but similar peer-to-peer video streaming systems. See Section 2.6 for a brief introduction to PULSE and other peer-to-peer video streaming systems.

One design goal of Cooperative Internet Access is to keep the actual peer-to-peer video streaming system **unchanged** on a global scale. Cooperative Internet Access requires changes within a peer-to-peer node's logic (replacing the download logic of the chunk scheduler – see Section 5.3.6.3) and also in the structure (i. e., adding the sharing-link and the measurement module – see Section 5.3.6.1). However, the protocol used in the global peer-to-peer system remains unchanged, as all changes are of local nature to the Cooperative Internet Access nodes. The changes are also only applied if a node decides to use Cooperative Internet Access, otherwise it behaves exactly like any other peer of the particular peer-to-peer system.

5.3.6.1 Adapted Peer Architecture

The peers architecture in Figure 5.8 re-uses the node architecture in Figure 4.10 and is the piece of the software to be deployed on the mobile nodes. The lower left part (*chunk buffer*, *chunk retrieval or exchange*, *local playout*) is basically a mesh-pull P2P video streaming system. The *local chunk redistribution* is responsible for the redistribution of video chunks within the group, and is part of the Cooperative Internet Access system.

The content retrieval from the Internet may use the same protocol, or a different one, P2P or client/server based. We assume that one of the peers is elected to be the *central controller* and that it executes as described in Section 5.3.5. It is responsible for telling the other peers in the group which chunks to download via their access-links, respectively. The peers store these assignments in a request queue.

In addition to that, each peer monitors the effective throughput on its access-link. Based on these measurements and the recent history it tries to predict till when it will be able to complete the tasks in its request queue (this assumes that all chunks are of equal size or that the sizes are known a priori). For each pending request in the queue it reports several tuples *(point in time, likelyhood that chunk download will be finished by then)* to the controller, as outlined in Section 5.3.5.2.

5.3.6.2 Peer Data Structures

For the operations, we need an access-link interface list (*acli_list*), as there could be more than one access-link on a single node. This acli_list stores the type of interface (e. g., UMTS with the name of the network operator), the statistics of the link obtained by the the throughput measurements and also the information about the estimated future achievable throughput.

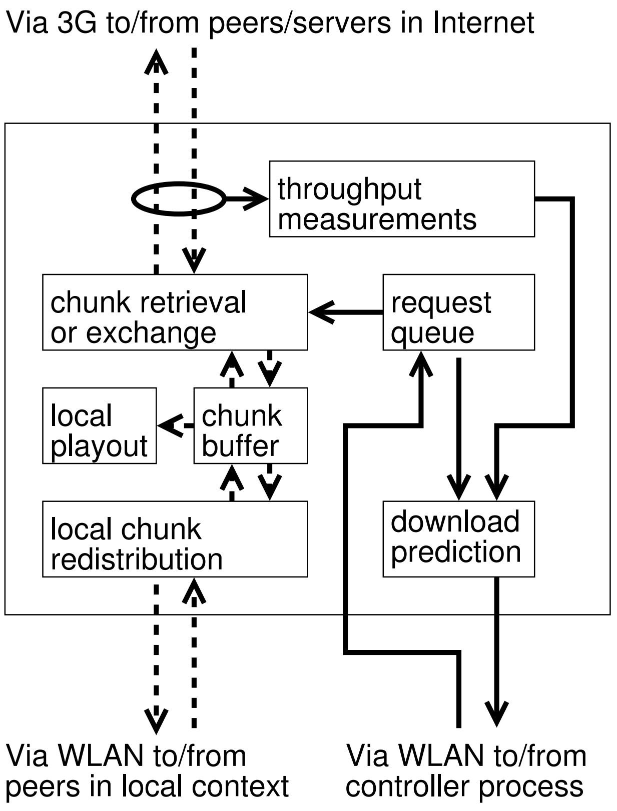

Via 3G to/from peers/servers in Internet
throughput measurements
chunk retrieval or exchange
request queue
local playout
chunk buffer
local chunk redistribution
download prediction
Via WLAN to/from peers in local context
Via WLAN to/from controller process

Figure 5.8: Node Architecture

For each acli_list, a chunk request queue (*rq_queue*) is maintained. This queue keeps the chunk retrieval assignments told by the controller process.

Each peer has to maintain a list of the participating peers (*peer_list*) in the local system. This peer_list is used and implemented the same way as described in Section 4.4.3. This list keeps record of the participating peers and also which of the local peers is acting as the controller process.

The Cooperative Internet Access peer re-uses the data structures of the peer-to-peer system to a large extend. For instance, the chunk buffer of the peer, the handling and storage of the buffer maps, the neighbor (passive) peer list and the active peer list. Chunks are copied to the local chunk buffer, if they are received form the local peers via the local chunk redistribution and vice versa, if other local peers request a chunk from a peer, the chunk is copied from the local chunk buffer to the local chunk redistribution. The buffer maps are read and checked for changes by the Cooperative Internet Access part of the peer and changes are send to the controller process.

The design goal is to keep the change impact on the peer as low as possible, so that the Cooperative Internet Access approach can be integrated in a wide range of peer-to-peer systems.

5.3.6.3 Adapted Peer Logic

A Cooperative Internet Access peer requires changes in the logic of a peer to be enabled to participate in the Cooperative Internet Access system. However, most parts of the logic stay the same, such as, the Peer Manager and its functions, the Buffer Manager, the System Management, and the Network Interface.

Cooperative Internet Access requires that the chunk download is coordinated by the controller process and not controlled by the peer itself anymore. The chunk upload can be still handled by the peer, as the peer has to honor the chunk upload requests from remote peers. This requires a change in download chunk scheduler of the peer scheduler logic. The peer does not determine on its own which chunk it is going to request from its remote peers, but the controller process assigns a chunk request queue to this peer. The peer will retrieve the chunks in the order of the request queue from its remote peers. However, it is peer's decision to choose about the remote peer where it sends the download request to. This allows the peer to react to any local change which is not affecting the local Cooperative Internet Access system. For instance, the local peer requests the download of a chunk from a remote peer, but the remote peer just vanished. The local peer can just pick one of its other remote peers for the chunk download, as long as they also have the chunk that is to be requested (this information is available via the buffer maps).

There might be the case, where the controller process assigns no chunks to a particular peer, resulting in an empty request queue. This allows the peer to switch back to its regular scheduling (if possible) and exchange chunks with its remote peers. This will keep the peer in the loop of the peer-to-peer system, as it may be required to download and upload in the peer-to-peer system design in order to learn about other remote peers and their state. The peer must switch back to process the request queue if it receives an *Assign RQ* message with a request queue assignment.

PULSE uses an epoch of 2 seconds to evaluate the usefulness of the remote peers, which includes running the chunk scheduling algorithm. This is kept for the upload part, as the download part is under the control of the controller process which is evaluating the download situation more frequently (see Section 5.3.5.2).

Peer-to-peer streaming systems download and upload to a maximum number of remote peers only. The in-degree and out-degree of a peer [108], denotes from how many peers data is downloaded and to how many peers data is uploaded respectively, at the same time. Peers usually have ongoing data exchanges with several peers. This is beneficial for peers that have sufficient achievable throughput, but may turn bad for peers with insufficient achievable throughput. Typically, peer-to-peer streaming systems rely on TCP which tends to equally split the overall achievable throughput amongst the TCP connections (given that the access-link of the peer is the bottleneck). This is fine, as along as the access-link is not resource constrained, but of limited use when the access-link is resource contraint. In such a situation, Cooperative Internet Access will enforce that the peer downloads only a single chunk from a peer and also uploads only a single chunk to a peer. These considerations are also valid for a properly implemented UDP-based system, if such a system is implemented in a TCP-friendly way [RFC 5348].

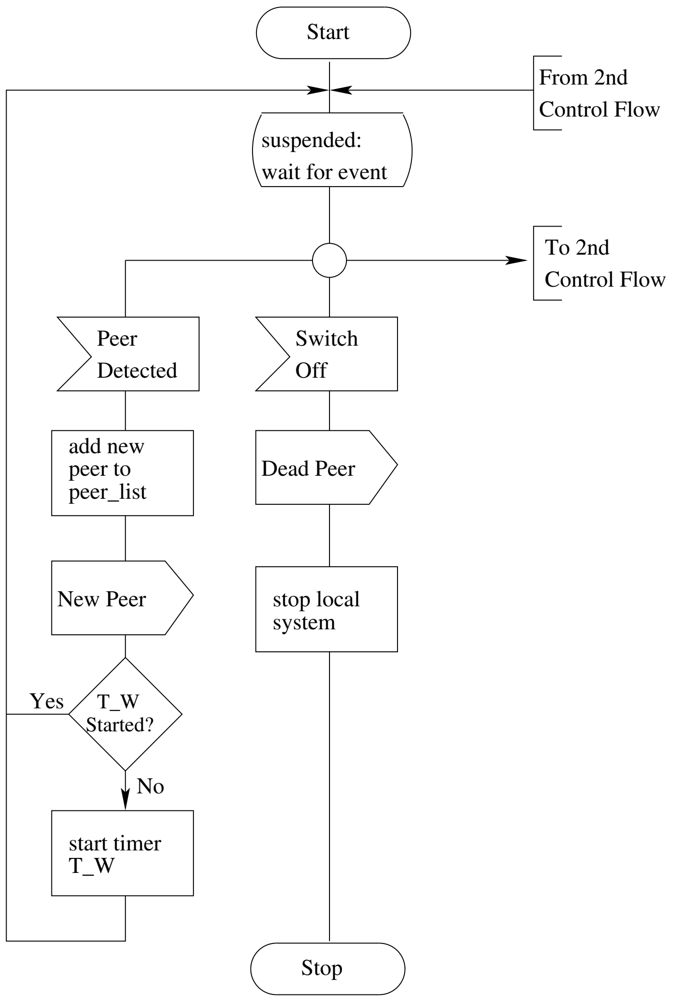


Figure 5.9: Cooperative Internet Access Peer Control Flow – Peer Management

5.3.6.4 Peer Control Flow

The Cooperative Internet Access peers are split in the original peer-to-peer streaming system part and in the actual Cooperative Internet Access part. The peer-to-peer streaming system part runs as soon as the user starts the peer application on its computer device. The Cooperative Internet Access part will stay idle and do not interfere with peer-to-peer streaming until other Cooperative Internet Access peers are detected on the sharing-link. Once other Cooperative Internet Access peers are detected (given that the users enables Cooperative Internet Access), Cooperative Internet Access will take over the logic, as described in Section 5.3.6.3. The control flow chart Figure 5.9 shows the beginning of the control for a Cooperative Internet Access peer, but only the pure Cooperative Internet Access peer management part. The actual control flow for the cooperative chunk retrieval effort is shown in this chart Figure 5.10.

The Cooperative Internet Access peer control flow is event driven, i. e., either a local event or or a remote event (received via a message) will cause an action by the peer. In the initialization phase, where no other Cooperative Internet Access peer is around, the peer waits for a message of any arriving Cooperative Internet Access peer (*Peer Detected*) in Figure 5.9. On receiving a *Peer Detected* message, the receiving peer will reply with a *New Peer* message to notify the other peers about its own presence. The New Peer message contains information about the current content the peer is interested in (e. g., the TV or video channel) and information about its access-link or access-links. The peer will start its local hold timer (T_W in the figure, T_W in the text), if it has not already been started. In the case the peer will be switched-off by the user, i. e., the peer can gracefully leave the system, it notifies the other peers and the controller process by sending a *Dead Peer* message and stops the local Cooperative Internet Access system afterwards.

Once a peer has detected other peers, and also the presence of a controller process, it will be ready to handle the events and messages shown in the control flow chart in Figure 5.10. The peer's control flow chart mirrors the controller process flow chart in terms of message to be sent or received. The various events or messages are:

Assign RQs This message is sent by the controller process and carries the new request queue per access-link on that peer. The peer checks the request queue information and adds or replaces its own rq_list or entries within. The successful reception and handling of the *Assign RQs* request is acknowledged with an *ACK*. The absence of an *ACK* is counted at the controller process as an indication that the peer is unresponsive.

Pull Chunk RQ The controller pulls a particular chunk or chunks from the peer's request queue. The peer removes the chunk from its request queue and stops the downloading of the chunk, if the chunks is being downloaded at that time.

Received Chunk This message is sent by any peer that has successfully downloaded a chunk. A peer receiving such a message can load the chunk to its local chunk buffer from the peer which retrieved the chunk.

Loaded Chunk This is local event occurring if this peer has successfully loaded a chunk via its access-link. This event has to be sent immediately to all other peers and the controller process, allowing every peer to load the chunk and to tell the controller process that this chunk has arrived in the system. The peer prepares a *Received Chunk* message, together with an implicit Situation Report (see Section 5.3.5.2). The peer will optionally add a Buffer Map Update, if there is any updated information about the buffer maps.

T_W, T_W expired The hold timer T_W expired, as there has been no other event for the time. The peer prepares a *Situation Report* message including the current status of the request queue. The peer will optionally add a Buffer Map Update, if there is any updated information about the buffer maps.

Local Situation Changed The achievable throughput situation has changed and the peer notifies the controller process about this with an explicit *Situation Report* message (see Section 5.3.5.2). The peer will optionally add a Buffer Map Update, if there is any updated information about the buffer maps.

5.3.7 Dead Peer Detection

The Cooperative Internet Access peers, as well as the controller-process which is executed on one of the peers, must detect dead peers, as those peers are consider in the scheduler but cannot

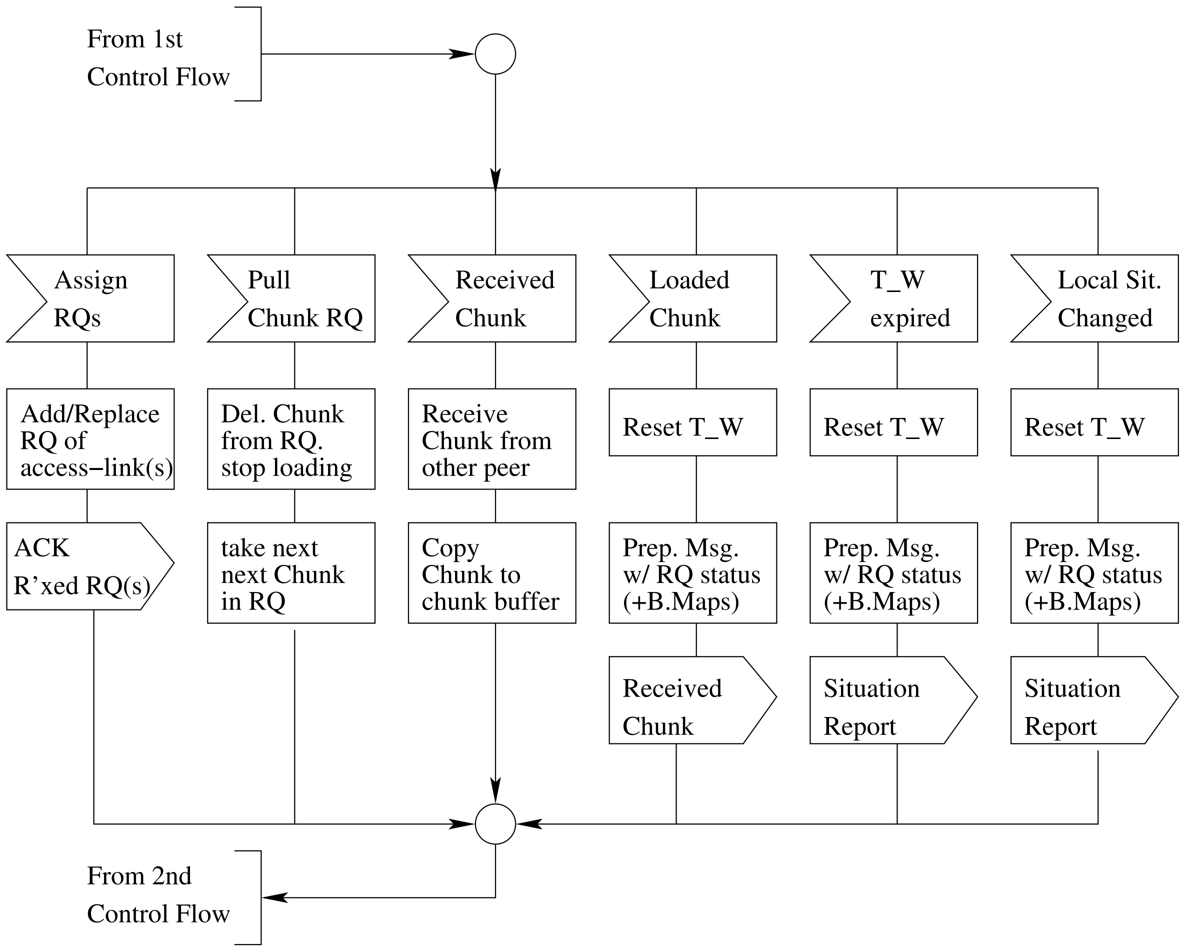


Figure 5.10: Cooperative Internet Access Peer Control Flow – Chunk Management

contribute to the system anymore. A dead peer is a peer which left the system without notifying the other peers via the *Dead Peer* message.

Each peer must show signs of its liveliness by reacting to messages sent to it or by periodically reporting its status. A peer acknowledges the reception of a new request queue with an *ACK* message to all peers (including the controller). The controller must keep track of the *ACK* messages of all peers and react to a missing *ACK* message within the hold timer period. This will cause the detection of a missing *ACK* message after a maximum time of T_W. A peer reports periodically its status even though there is no chunk assignment or no change to report. Such a peer sends a *Situation Report* to all peers.

A particular peer is considered as dead if the dead peer watchdog timer expires. This timer is kept for each peer and is reset upon the reception of any message from that peer. This timer expires, if there is no message received from this peer after the expiration of the hold timer T_W. Such a dead peer is removed causes the access-link of the peer and the peer itself to be removed from the access-link list and the peer list, respectively.

6 Web Browsing in Constrained Environments

6.1 An Introduction to Web Browsing

HTTP-based [RFC 2616] traffic carrying HTML-content [109], as known as web traffic, is the most popular traffic type in the Internet today (2010). A recent study of the Internet's traffic composition outlined that 52.00 % of the measured traffic in 2009 is Web traffic ([99], slide 15) compared to 41.68 % in 2007. Web traffic is carrying HTML, where the content inside the HTML or referenced by a HTML document itself can vary a lot between, for instance, text, picture, videos, or interactive web pages that offer the functionality of a word processor program.

The further reading of this section requires a basic understanding of how HTTP and HTML work in practice.

6.1.1 HTTP-based Web Access

HTTP [RFC 2616] is a request/response protocol that is used to retrieve elements from web servers or to upload elements to a web server. Web browsing is a client/server based Internet application between a web browser and web servers. The web browser is the user interface, where the user can enter a link to the desired web page to see and where the web page is eventually displayed. The actual content, i. e., the web page are hosted on a HTTP-server or on multiple, different web server.

The user has to enter a Uniform Resource Locator (URL) in the browser or to click on a link. The web browser uses the URL to locate the web server and to retrieve the web page. A typical HTTP-request for a web page is (as captured on the wire), for example:

```
GET / HTTP/1.1
Host: scholar.google.de
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.19)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
[...]
```

where the response is the *index.html* document, which contains further references to more information elements, such as graphics, text, video, etc. A typical setting is that the *index.html* contains the text of the web page, but that all remaining information elements on that page now have to be retrieved element by element by the browser. Here is a typical response:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
X-Content-Type-Options: nosniff
Date: Tue, 10 Aug 2010 15:10:00 GMT
Server: scholar
Transfer-Encoding: chunked
----------------: ----
Cache-Control: private, x-gzip-ok=""
[...]
```

The HTTP response shown above denotes the *Content-Type* carried in the message payload. The HTTP payload is called message body or entity body. The response can carry a number of additional header fields not mentioned here other than the *Cache-Control* header which is used by HTTP proxies, as discussed in the following section.

6.1.2 HTTP Proxies

One element specified in the HTTP standard [RFC 2616] are HTTP proxies. These proxies are intermediaries between the web browser and the HTTP server and can interfere with the HTTP-traffic. A HTTP-proxy can serve several purposes [110]:

- ”to keep machines behind it anonymous;
- to speed up access to resources (using caching);
- to apply access policy to network services or content, e.g. to block undesired sites;
- to log or audit usage, i.e. to provide company employee Internet usage reporting;
- to bypass security or parental controls;
- to scan transmitted content for malware before delivery;
- to scan outbound content, e.g., for data leak protection;
- to circumvent regional restrictions.”

The so-called HTTP cache proxy (cf. Section 8.1.3 of [RFC 2616]) intercepts HTTP traffic and creates local copies of the information elements transferred via HTTP if the proxy is allowed to do so (this depends on the cache headers in HTTP). The proxy intercepts any further HTTP traffic and retrieves only the elements from the Internet that are not yet stored in the proxy or that are not allowed to be stored in the proxy. This can reduce the traffic on an access-link from the Internet to the hosts behind that proxy, if the web pages being retrieved via this proxy have shared elements. Shared elements may be, for instance, pictures such as site logos or video trailers embedded on a web start page which are access by multiple browsers or users. For the remainder of this section we refer to HTTP cache proxies as HTTP proxies.

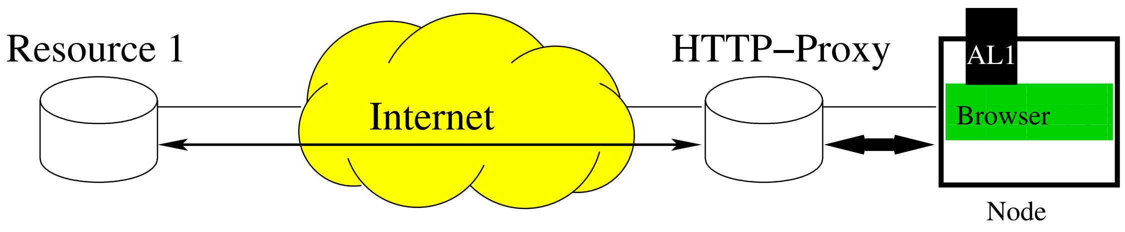


Figure 6.1: HTTP proxy for a single node

Figure 6.1 shows a single node that connects to a HTTP proxy and retrieves a web page (shown as resource 1). The hosts requests a web page that will result in a large data transfer, depicted as big arrow between the node and the proxy. The proxy does not need to retrieve all information elements form the resource (given that the proxy could already cache some elements in an earlier data exchange) but can use the cached elements and needs only to retrieve a subset from the resource, depicted as the smaller arrow between the resource and the proxy. Today's web browser are also caching elements in their local browser cache. These cached elements are only reusable by the same browser while network cache-proxies can cache elements across different browsers and across different hosts.

The HTTP proxy should not simply cache everything that is transmitted via it, but must follow the cache control information included in the HTTP messages, as defined in Section 14.9 of [RFC 2616]. For instance, the *Cache-Control* header defines if the information elements transferred in the entity body is *private* and may not be cached or if it is *public* and may be cached. Information elements which are allowed to be cached have also more attributes assigned to them for the cache's operation. For instance, the *max-age* attribute indicates until when the element is valid for caching. The element must be again retrieved after the expiration of the stated time in *max-age*.

6.1.3 Structure of a Web Page

A web page is assembled of a HTML file (with content) and more references (links) to other elements that must be retrieved to properly display the page to the user. The browser is actually taking the html file and loads the other missing elements from their respective locations, i. e., the other elements must be loaded from the server that served the html file, but they can be loaded from anywhere else.

The content transferred via HTTP and HTML is changing frequently. For instance, some years ago, most of the web pages were rather static, until the advent of the so-called Web 2.0 technologies [111], such as Asynchronous JavaScript and XML (AJAX) [112] which lead to more dynamic web pages. The differences between static and dynamic web page are:

- static pages are not changing the content after they have been loaded in the browser;
- static pages are roughly the same for every user;
- dynamic pages are generated dynamically for each users, for instance, depending on the used browser;

- dynamic pages can still change even after the were loaded in the browser.

Dynamic web pages are hard to be cached for cache proxies, as either the specific element cannot be cached or (worse) there is no overlap in elements delivered by the web server to the browser. This prevents a cache proxy to cache elements for reusing them later. However, even today (2010) with the majority of web pages commonly used being dynamic web pages, web site designers use different methods to differentiate between dynamic content and static content [113].

Our observation is that web sites either use a specific "directory" on their web server or a second host to server static content. The specific "directory" is a fixed subtree of the URL on the same server that is used to store all static elements, such as background figures. The request below is referring to the path */static/sys/...* for this purpose:

```
GET /static/sys/v9/bg/bg_suchmaske.jpg HTTP/1.1
Host: www.spiegel.de
[...]
```

The reply to this:

```
HTTP/1.0 200 OK
Date: Tue, 10 Aug 2010 01:19:15 GMT
Server: Apache/2.0.63 (Unix) DAV/2
Last-Modified: Mon, 10 Aug 2009 10:32:40 GMT
ETag: "501f9-2c0-470c71af66200"
Accept-Ranges: bytes
Content-Length: 704
Cache-Control: max-age=259200
Expires: Fri, 13 Aug 2010 01:19:15 GMT
Content-Type: image/jpeg
[...]
```

A web site may also use a second host to serve static content, which is identifiable by a different host part in the URL. The second host is usually embedded in the HTTP message body, i. e., the HTML text. For instance, at browser that is retrieving a web page, it initially retrieves the start page:

```
GET / HTTP/1.1
Host: www.facebook.com
[...]
```

In the subsequent steps is additionally fetches the static content from the second server (*static.ak.fbcdn.net* in this case). The second server is typically hosted on a Content Delivery Network (CDN) that is able to server the content to a large audience without loading the actual originating content server.

```
GET /rsrc.php/zEX21/hash/75j4m1ms.png HTTP/1.1
Host: static.ak.fbcdn.net
[...]
```

The reply to this:

```
HTTP/1.1 200 OK
Content-Type: text/css; charset=utf-8
Last-Modified: Sat, 01 Jan 2000 00:00:00 GMT
P3P: CP="DSP LAW"
Pragma:
Content-Encoding: gzip
X-Cnection: close
Content-Length: 5483
Vary: Accept-Encoding
Cache-Control: public, max-age=30848512
Expires: Tue, 02 Aug 2011 16:55:47 GMT
[...]
```

6.2 Challenges for Web Browsing

Web-based browsing has less stringend network requirements compared to the peer-to-peer video streaming discussed in Section 5, as the resources accessed via WWW are in most cases[1] not time-critical, but nonetheless, the volume of data is considerable for resource constrained environments. The pace maker for web-based applications seems also to be what is possible today with fixed-line network access technologies. For instance, [114] claims that the average size of a web page, that is the web page and all elements that are loaded for that page, has quintupled from 2003 to 2009. However, the achievable throughput of fixed-line high speed access-lines, such as, FTTH, has constantly increased, but with a faster pace as compared to other technologies, for example, UMTS.

The trend of growing web page sizes is impacting the other end of the spectrum of network access lines – the users with low-bandwidth network access, such as, for instance, dial-up modem users or mobile wireless users in rural areas without fast mobile wireless. These users are also in a *resource constrained* network environment and will suffer from the gap between their slow network access and the growing web pages: they will wait for a very long time to see the page or give up and cannot participate anymore in that.

Whether a node (and the respective user) is in a resource constrained environment is relative to what the users of the web applications are expecting of the network: a very patient user may be willing to wait for a long time for a we page to be delivered while an impatient user will simply give up waiting for the web page soon. There is a vast number of studies on how the time to deliver web page influences the behavior of a user, as for instance in [115, 116]. However, in turns out that the the upper bound for a web site to be displayed to the user (this includes network retrieval time and rendering time of the browser) is below 1 minute. Most of the users will stop from either accessing a web page, if this particular page is loaded too slowly, or completely refrain from using the Internet, if all web pages are loaded too slowly.

We call a user accessing a web page to be in a resource constrained environment if the user cannot access the desired web page within 1.5 times the regular acceptable ”waiting” period for the retrieval and display of a web page.

[1]The author cannot rule out this possibility, given the versatile usage of HTTP and HTML.

6.2.1 Preconditions

Web traffic has its own specific traffic pattern between the web browser and the server. The traffic is very bursty followed by longer idle periods: the user is requesting a web page and it is loaded by the browser as fast as the network is able to deliver the page and page's elements [117]. However, in recent years the traffic patterns have changed as the applications working on top of web traffic have changed.

Two recent observations are blogs, i. e., where user can actively publish their own content on web servers [118]. In this case, the traffic pattern is changed in the way that not only elements are downloaded but also more frequently uploaded. On the other hand, there are more and are web pages based on Asynchronous JavaScript and XML (AJAX) [112] that fetch further content even after the web page has been load. The observations in terms of changed traffic patterns are that AJAX applications transfer more bytes in each session compared to non-AJAX applications; they have a larger number of requests per session; and the inter-request-times are significantly shorter [119].

Nonetheless, the one of the main characteristics of web traffic remains: the burstiness but with changed inter-request-times.

6.2.2 How much is cacheable?

As of today (2010), it is impossible to find literature about how much traffic HTTP proxies are going to save in general and also in particular situations. There are some vagues numbers, but there are no reliable sources on that topic. The main reason for this seems to be that the savings by HTTP proxies depend on the type of users using the proxy (e. g., enterprise vs. home users), the type of content accessed (e. g., news pages or social networking sites), and how web pages are designed or operated (e. g., web pages can allow or disallow to cache element).

We have two sample HTTP proxy excerpts to show how much traffic can be cached by such a proxy and how dependent it is on the visited web sites – and thus dependent on the users. Both statistics in Table 6.2.2 and Table 6.2.2 are courtesy by two different companies who provided the statistics while not being mentioned. The proxies are located on the up/downlink of the companies and operate as a cache. Both samples are reflect the statistic for the period July 1-31, 2010.

Table 6.2.2 shows the top 10 entries access in company A, while only a few entries are actually being cached. The last column indicates the total amount of bytes cached as compared to the column "Total Bytes". There is almost no caching for the most popular site (google.de), but the 2nd and 3rd most popular sites are ached to some extend. These sites have typically a number of information elements which are cacheable and also usable by multiple users, e. .g, they have mainly common banner graphics, etc.

Table 6.2.2 shows the top sites accessed. The last column "hit ratio" reflects how much of the traffic in bytes has actually been cached. For instance, de.archive.ubuntu.com and www.spiegel.de have both a significant cache hit ratio of 34.16 % and 51.58 %. Our sites, which generate a large

Site	Requests	Page Views	Browse Time in days	Total Bytes in MBytes	Bytes Received in MBytes	Bytes Sent in MBytes	Cache Bytes in MBytes
www.google.de	1813191	269.725	29.0	8290	6950	1340	147.07
www.facebook.com	1543093	236.547	25.2	13100	11850	1240	4480
www.bild.de	3868298	138.576	11.9	43380	39760	3620	33040
webradio.antenne.de	163239	128.379	7.6	292.67	194.06	98.60	50.61
cluster.evig.de	102657	98.342	5.4	145.52	83.93	61.59	3.75
85.17.147.131	7765	77.374	3.8	63.62	18.41	45.21	0.062
www.google.at	413575	73.704	7.2	2020	1700	328.18	39.37
www.dir.bg	75595	65.937	9.4	107.74	36.50	71.24	3.16
www.meinvz.net	1060616	63.283	8.0	7740	6870	888.88	3730
service.gmx.net	682701	60.362	3.0	2960	2160	813.42	1190

Table 6.1: Sample proxy statistics for company A, sample time July 1-31, 2010

destination	No. of request	% total requests	hit rate in %	seconds per request	Bytes	overall part in %	hit ratio bytes in %
de.archive.ubuntu.com	65677	0.77	84.93	0.13	12816M	5.57	34.16
liveupdate.symantecliveupdate.com	128346	1.51	18.82	0.10	6726011K	2.86	18.28
swr.ic.llnwd.net	88	0.00	0.00	8388.06	6565074K	2.79	0.00
download.windowsupdate.com	5011	0.06	16.82	0.84	6195866K	2.63	1.25
195.10.10.207	29	0.00	0.00	12505.11	5508975K	2.34	0.00
ftp-stud.fht-esslingen.de	572	0.01	43.71	2.10	5127182K	2.18	0.01
ftp.hp.com	276	0.00	0.36	69.97	4033935K	1.71	0.72
armdl.adobe.com	3788	0.04	1.11	6.09	2890612K	1.23	0.53
download.oracle.com	588	0.01	36.73	2.34	2835923K	1.20	0.21
us.archive.ubuntu.com	2784	0.03	5.46	0.47	2581540K	1.10	0.01
ftp.uni-erlangen.de	2412	0.03	0.17	0.46	1996251K	0.85	0.26
safebrowsing-cache.google.com	135601	1.60	34.44	0.04	1891303K	0.80	43.73
v17.lscache7.c.youtube.com	41	0.00	0.00	30.77	1841443K	0.78	0.00
ftp.de.debian.org	2393	0.03	31.30	0.30	1695222K	0.72	10.07
www.spiegel.de	139922	1.65	72.12	0.03	1671029K	0.71	51.85
svn.ict-societies.eu	407	0.00	0.00	1.69	1509855K	0.64	0.00
download.microsoft.com	88	0.00	9.09	33.08	1326076K	0.56	5.07
www.myotherdrive.com	253	0.00	0.00	16.10	1295668K	0.55	0.00
is2.myvideo.de	375	0.00	47.20	1.46	1295302K	0.55	32.19
kh.google.com	83592	0.98	0.05	0.07	1252498K	0.53	0.00
archlinux.limun.org	1028	0.01	0.39	0.29	1246514K	0.53	0.00
au.download.windowsupdate.com	2268	0.03	4.63	1.10	1147744K	0.49	5.08
www1011.megaupload.com	1	0.00	0.00	229.54	1024000K	0.43	0.00
www1123.megaupload.com	1	0.00	0.00	233.77	1024000K	0.43	0.00
www1041.megaupload.com	1	0.00	0.00	253.29	1024000K	0.43	0.00
other: all-level-domains	7921132	93.23	24.77	1.85	154974M	67.38	10.17
Sum	8496674	100.00	25.80	1.87	230003M	100.00	10.36

Table 6.2: Sample proxy statistics for company B, sample time July 1-31, 2010

amount of traffic, such as youtube.com, are not cached at all. This can be due to non-cacheable content, or the proxy configuration is set to ignore that type of content.

However, both tables show that the access web sites can vary a lot between different user groups and thus also result in a different amount of traffic that can be cached.

6.2.3 Challenges in Resource Constrained Environments

Access to the web pages and the resulting process of retrieving the information elements is difficult to predict. It is hardly foreseeable, when a user wants to browse a web page, i. e., sending a request to the browser. Second, the complete size of the web page to retrieve is not that obvious, as it consists of numerous information elements, that can each have a varying size. The size is not known a prior in HTML, as only the element name and the link to it are given. The HTTP protocol [RFC 2616] offers a way to retrieve the content-length of an element with the HEAD method which is discussed in Section 6.3.3. This is fundamentally different from the peer-to-peer video streaming application in Section 5, where the size of the information elements (chunks) is known in advance and also the timing of the chunks.

6.3 Cooperative Web Access

We are coming back to the idea of Cooperative Internet Access of letting multiple nodes share their access-link to mitigate a resource contraint situation for web browsing. Let's assume that we have multiple users that are in the same local context, e. g., a remote village without high-speed Internet access, but most of the users have a mobile phone or an ISDN phone line. This situation is common in many rural areas, in almost any country – independent of whether it is a developed or developing country. A user that is browsing the web will need to be very patient, as the size of web pages has grown of the years and is still growing (see discussion in Section 6.1), but the maximum bitrate, and thus the achievable throughput, of the access lines have not.

One way of improving the situation is to deploy more network capacity in these areas, but this in not under the control of the users. A better way is to cooperatively access the web by sharing the access-links of multiple users. An important precondition to this in the scope of web access is that it is very unlikely to have multiple users accessing exactly the same content, but that there is an overlap between the content. For instance, many people use facebook [120] to keep in touch with their friends and relatives, as of today (2010). Only a few people will access the same web pages on facebook's web site but these web pages often have common elements, such as banner pictures or background pictures. These common elements can be locally stored and do not need to be retrieved over and over again by each but shared amongst them.

Another important element for cooperative web access is mentioned in Section 6.2.1: web traffic exhibits a very specific traffic pattern, of retrieving the web page at a relative short period of time, using the access-link for that time, but also with relative long times of no or low activity on a link. The access-link has some spare capacity left which can be used to retrieve information elements on behalf of other users.

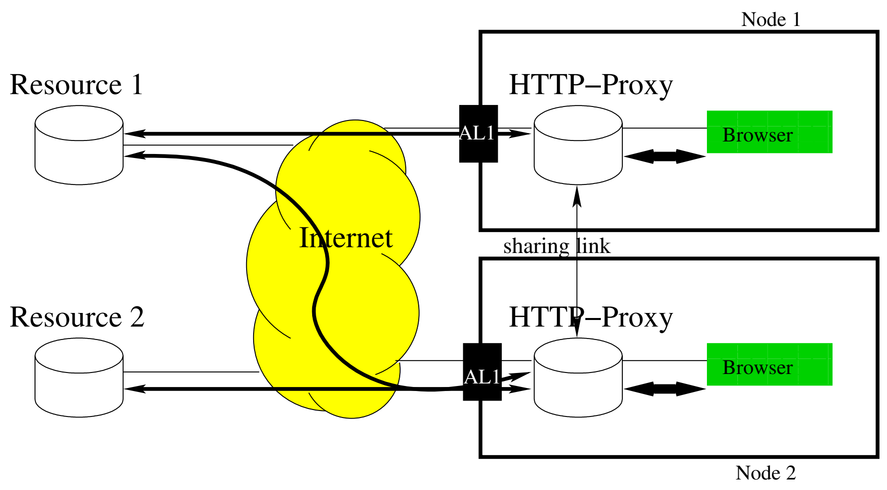


Figure 6.2: per node HTTP-Proxy for a single node with cross proxy communication

The cooperative web access approach combines these two elements, i. e., common elements on web pages and spare capacity, to decrease the retrieval time of web pages for the users.

6.3.1 Cooperative Web Access enabled HTTP

Some modifications to the nodes are required to enable them for cooperative web access. The access to the web page has to be monitored by an entity and this entity has to judge if elements are to be retrieved via the node's access-link, re-used from a cache (either in the browser or from a HTTP proxy), or retrieved via the access-link of another node. The nodes need to be able to see if other nodes in the vicinity have elements cached which otherwise the nodes would have to retrieve on their own. Second, the nodes need to be able to request other nodes to retrieve elements on their behalf. The retrieval *on-behalf of others* is the difference to the conventional approaches, as discussed in Section 2.4.2.

All of this can be embedded in a HTTP cache proxy which is an intermediary for the HTTP protocol and also caches elements transferred via HTTP, as described in Section 6.1.2. The HTTP cache proxy, or short HTTP proxy, has to be part of the node and can be either implemented as a separate entity on that node or it can be embedded in the web browser. The web browser on this node access web pages via the HTTP proxy, as depicted in Figure 6.2. The two nodes in the figure connect their HTTP-proxies via the sharing-link. Whenever the proxy is now receiving a request from its local browser to retrieve an information element, it looks up the information element in its local cache. The local proxy requests the information element from the proxy of node 2, if the element is not found in the local cache. Only if there is no copy in any of the proxies, the element is retrieved via an access-link directly from the resource. If an element is found in a cache, this is called *cache hit*, as compared to if an element is not found, it is called *cache miss*.

There is a number of such inter-cache protocols that implement the coordination of a group of caches, such as, Internet Cache Protocol (ICP) [RFC 2186] , Hyper Text Caching Protocol (HTCP) [RFC 2756], and Cache Array Routing Protocol (CARP) [121], BuddyWeb [45] or Squirrel [46]. HTTP proxies can only query other proxies whether an information element is already cached or not, but they cannot request other proxies to retrieve information element on behalf of it. Once a proxy receives a request from its local browser, it looks up the various information elements of a web page in its local cache. Cache hits are directly served, while cache misses have to be retrieved via the own access-link. The *cooperative web access* approach extends the principle of inter-cache communication with the cooperative Internet access, so that the node local proxy can retrieve elements from other proxies' caches, or via an access-link of another cache (i. e., on behalf of the requesting proxy). This allows to use the available and achievable throughput of the other access-links of the nodes in the same local context.

6.3.2 Situation Manager

A first step before contacting other cooperative web access nodes is to judge if the node is in a resource constrained environment. The node itself may be a mobile node that is connected to various access-links at different point of times, resulting in situations where the achievable throughput is sufficient for accessing the desired web pages and also situation where the achievable throughput is not.

The situation manager takes care of this and can either use passive measurements embedded in the browse to judge if the desired content is retrieved fast enough, or offer a switch to the user, so that the user can on its own decide that the retrieval process is too slow. The issue here is that the waiting time between requesting a web page and the final delivery, i. e., the retrieval time, is mainly a subjective factor: It may depend on the person accessing content, its expectation in terms of retrieval time, if the person is patient or impatient, and probably even more factors, e. g., if the person is in a hurry or not or if the person is used to access the Internet via a modem line only.

The situation manager could maintain a user specific heuristic to judge whether the retrieval time is too long or nor and switch to the cooperative web access mode and back, as needed.

The place for the situation manager is within the browser. The browser is the only place to determine if a web page is loaded to slowly, either by the rendering engine that detects that data is still missing after a time or the user that can hit a button indicating that is too slow for him or her.

A cooperative web access enabled node re-uses the membership management functions of the general Cooperative Internet Access framework (Section 6), to find other cooperative web access nodes in its physical vicinity.

The following section assume that the node is always in the cooperative web access mode.

6.3.3 Dissecting the Web Page

A web page is assembled of a HTML file (with content) and more references (links) to other elements that must be retrieved to properly display the page to the user, as outlined in Section 6.3. The browser actually takes the HTML file and loads the other missing elements from their respective locations, i. e., not all elements must be loaded from the server that served the HTML file. The browser requests these elements from the server, or in our case, refers to the proxy to retrieve the elements. However, traditional cache-proxies can only check of the requested elements are either already cached locally (cache hit), or on neighboring caches, or that the element is not cached at all (cache miss). In the latter case, the proxy has to retrieve the element from the origin server.

The proxy must also retrieve the missing elements via the access-link from the servers located in the Internet, for any of these elements. This is troublesome, as the proxy behaves similar to the browser and is experiencing the same issues as the browser, as outlined in Section 6.2.3. A smart proxy would dissect the web page and aim to prioritize which element is retrieved in what order.

To do this, the cooperative web access proxy has to the understand the various elements that can be loaded from a web page. This is a challenging job, as there are so many different element types, e. g., pictures in multiple formats, videos in multiple formats, 3D graphics, etc. – it is actually a plethora of element types which is hardly to manage by a proxy. Nonetheless, a proxy can roughly dissect a web page in classes of elements (cf. also the Internet media types signaled via the content-type header [RFC 2046] in HTTP or Multipurpose Internet Mail Extensions (MIME) [RFC 2045] in general), for instance:

plain text A plain text element.

text/html A html document that contains text and links to other elements.

mp4 A MPEG-4 file.

jpeg A JPEG file.

The cooperative web access proxy has to store a heuristic about each file type in general and about each file type per accessed web site, e. g., for youtube.com. Each heuristic stores the average value of the each content-type, learned retrospectively after retrieving an element of a particular content-type. The general, site-independent, heuristic is used for unknown web sites which have never been visited before. The site-dependent heuristic is built-up for each single site and used whenever this site is visited.

This heuristic information is used later in the scheduler of Section 6.3.4 to judge about the to be expected size of the information element before they are retrieved via one access-link.

This heuristic is required, as the information elements of a web page do not carry any information about the actual data size of any particular element, i. e., it is unknown to the retrieving entity (i. e., the browser or the proxy) how much data must be loaded for an element, at the time request time. The HTTP response carries a content-length header that gives the data size of an element. Another option supported by the HTTP-protocol is the HEAD method. The HEAD method is similar to the GET method, in terms of status information returned, but it does not retrieve any element. Using the HEAD method will cause an additional overhead. The proxy has to send a HEAD message for each element, which may be an additional burden for the resource constrain access line and also lengthen the whole process.

The additional burden would require in the worst case at least two Round Trip Times (RTT) per element to check with the HEAD method:

- 1 RTT to establish the TCP connection;
- 1 RTT to send the HEAD method to receive the result to it.

This neglects any processing time for the HEAD method at the server.

6.3.4 Scheduling Element Downloads

Let E be the list of all information elements required for a web page. The scheduling is performed for all missing elements E', i. e., elements which are neither available in the requesting node's cooperative web access proxy nor in the other local node's proxy. However, it is necessary to have a basic set of policies to let the user decide which data must not be requested via other nodes. The main reasons for this are privacy aspects (i. e., the other nodes can read the transmitted content) and security aspects (e. g., encrypted connections cannot be handled by other nodes and content may be changed and sniffed). The list of missing elements E' is sorted in descending order, i. e., information elements with a content-type which is assumed to be larger than any other content-type is set of the top of the list. The heuristic described in Section 6.3.3 is used to have an educated guess about the element's size.

We assume that the scheduler knows about the other node's throughput estimation by the time the scheduler operaters. The input to the scheduler is the list of information elements E' to be retrieved, the list of available access-links L with their throughput estimation. The list L is also sorted in descending order, i. e., fast links are on top of the list. The elements of the sorted list E' are to the links out of L. This will assign elements which are assumed to be large in size to the fast links, while elements which are assumed to be smaller are assigned to slower links.

The scheduler will assign as many elements as access-links available and will be called once an access-links is available again.

6.4 System Design Considerations

6.4.1 Bazaar Cache Protocol

This section describes the protocol used between the local nodes to determine if an information element is locally available or if it has to be retrieved, by either the requesting node itself or by any other local node. The protocol is called *Bazaar Cache Protocol* as it reflects a bit the situation at a bazaar: a request is ”shouted” (multicasted) to the other involved nodes to see what they can offer.

The system uses link-local multicast [RFC 5771], as defined in the Cooperative Internet Access framework in Section 4.4.2 to find other nodes and to query the other node's cache for information elements. These messages are sent via an unreliable UDP transport. A cooperative web

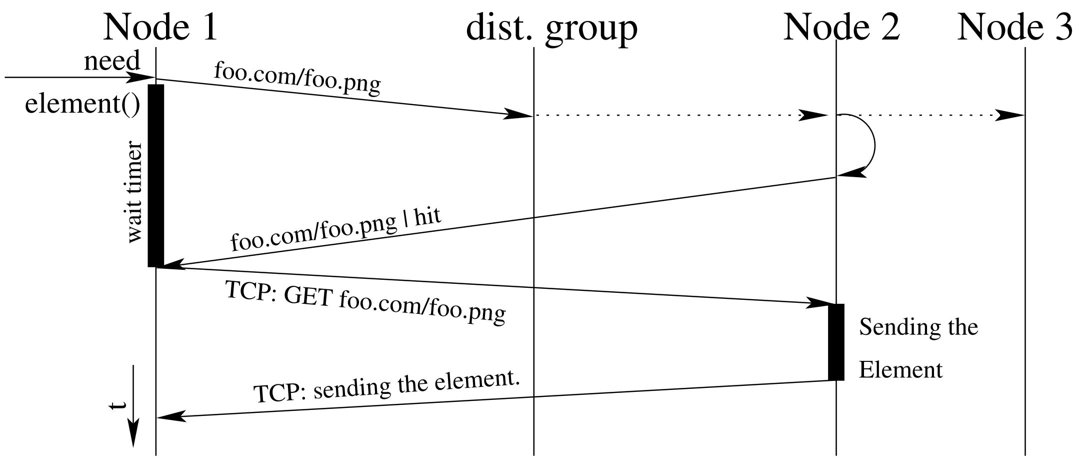


Figure 6.3: Schematic flow chart for cache hit

access node joins a particular link-local multicast address that is dedicated to cooperative web access. A node sends all of its request to this cooperative web access link-local multicast address and all participating nodes are receiving this request (unless there is a failure case, where the request gets basically lost). The nodes use link-local multicast to request other nodes about the availability of information elements in their cache and also their willingness to retrieve missing elements on behalf of the requester.

Using the link-local multicast mechanism for the protocol has the advantage that the nodes do not need to keep track about other nodes and also they do not need a communication association with them, for instance, a TCP connection to every host at all times. The disadvantages are that the message transport is unreliable, as all requests are sent to a multicast address. However, we assume that a local area network used for the sharing-link is reliable enough to deliver the messages. We describe in Section 6.4.2, how the system handles lost messages.

A node that needs to determine if any node has a particular element or a set of particular elements in its local cache sends a request message the multicast address (distribution group). This request message contains per requested element a (URL) (e. g., http://foo.com/foo.png). Each host that either has a cache hit or a cache miss but is willing to retrieve the element, replies directly to the requesting node. The element to be retrieved is very likely only of interest to the requesting node at the time of the request and therefore we use a reliable transport connection to the node it picks in order to retrieve the element.

Figure 6.3 shows the schematic flow chart for a cache hit. Node 1 requests a single element by sending the request message to the distribution group. Node 3 does not have the requested element and it may not have spare capacity left, and is thus not responding. Node 2 has the requested element cached locally and replies with a message directly to Node 1 indicating the cache hit. Node 1 now connects via TCP to Node 2 and retrieves the element.

Figure 6.4 shows the schematic flow chart for a cache miss with a single node being willing to retrieve the missing element on behalf of the requester. The request message is the same as for the cache hit case. Node 3 does not have the requested element and it may not have spare capacity left, and is thus not responding. Node 2 also does not have the requested element but is

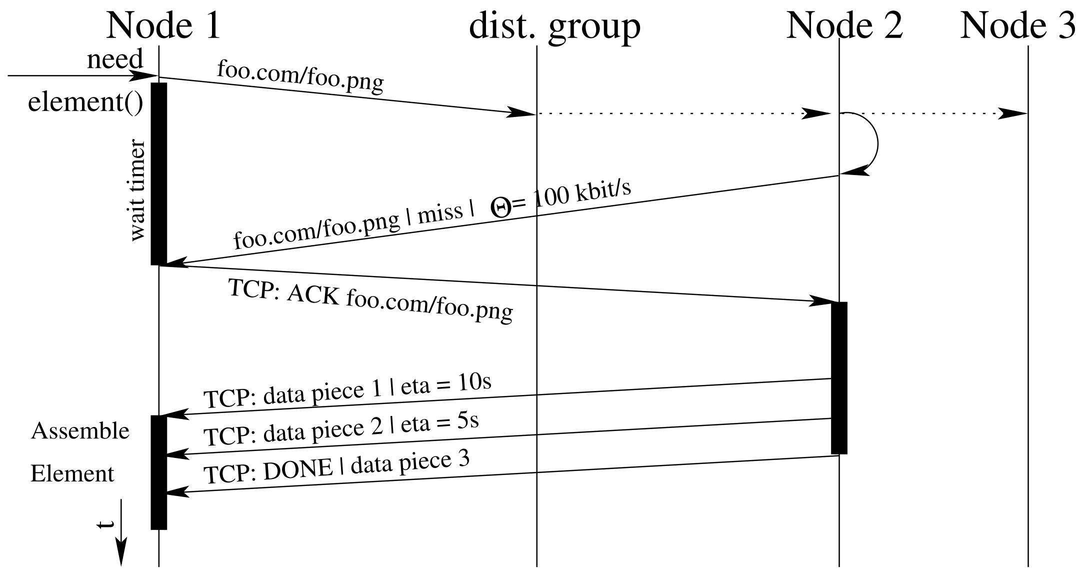


Figure 6.4: Schematic flow chart for cache miss

actually willing to retrieve the element and can serve the element with an estimated achievable throughput of $\Theta = 100 kbit/s$. Node 1 connects via TCP to Node 2 and requests the retrieval of the element (ACK message to Node 2). Node 2 becomes the handling node for this element and retrieves it via its access-link, and forwards each received segment of the element to Node 1. This cut-through allows Node 1 and Node 2 to detect if each other is still alive, plus Node 1 constantly receives an Estimated Time of Arrival (ETA) to judge how long it will still take until the whole element is retrieved. This cut-through has to be designed in a way that both transmission paths, e. g., from Internet to the handling node (Node 2) and from handling node to receiving node (Node 1), are non-blocking with respect to each other. Otherwise, a blocking transmission on either of these paths will block the transfer on the still working one.

Figure 6.3 and Figure 6.4 show that the responses are always addressed to the requester and not to all nodes, in contrast to the request message. This is one possible implementation of the request/response protocol. Another implementation could let the responses address the all nodes, by sending the response message to the distribution group's multicast address. This enables all nodes to learn about the other node's cached entries. Each node could cache these responses and try to use the cached responses to look-up already cached content faster. On the other hand, the caching of responses might lead to inconsistencies in situations where nodes are leaving or arriving frequently, as the cache would be outdated; further, the cache itself will need to main entries for data it probably is never in need of and the need to maintain the data, e. g., keeping track of how long these entries are valid and if the origin node is still keeping these entries in its local cache.

6.4.2 Controller Design

Each participating cooperative web access node suplies its own controller for the scheduling of the information elements. Each controller is bound to a node and has to handled the local requests for elements but also reply to requests from other local nodes. However, the controller does not manage the whole local system, as compared to the peer-to-peer case in Chapter 5 where a single controller-process handles the retrieval process of the local peers.

Existing proxy implementations have a mature set of cache organization algorithms and therefore the cooperative web access controller is meant as an extension to existing cache implementations, such as, for instance, the Squid Cache [122]. The controller itself does not keep any state across requests for element and it is instantiated for every new request arriving at the local proxy. The controller operates according to the flow chart in Figure 6.5.

For every new request for an element, the controller checks initially if this element is available in the local proxy. If it is available, the local proxy is handling this request (*serve element locally* in Figure 6.5) and the controller is done with this request. In the case that a particular element is neither in the cache, nor it is allowed to be retrieved by a another node, e. g., for privacy reasons, it is retrieved via the node's access-link (*Remote OK?* is decided to *NO*).

Elements that are not locally cached and that are allowed to be retrieved via other nodes are requested from other nodes (*Req. nodes for element*), as described in Section 6.4.1. The controller maintains a reply messages waiting timer (*timer1*0. *Timer1* is used to collect reply messages from the nodes, as there can be multiple replies to a request.

The controller will retrieve the requested element from a particular node, in the case of a cache hit (*timer1 & cache HIT* in Figure 6.5); following the flow chart in Figure 6.3. For a cache miss with some other nodes willing to retrieve the element (*timer1 & cache MISS*), the controller will select one node (if there are multiple nodes) and request the element from that node, as shown in Figure 6.4.

In the case there is no response at all (*timer1 expired*), the element will be added again to the element list E' and rescheduled in the next scheduler run.

The dimensioning of *timer1* is dependent on two factors: the round-trip time on the sharing-link and the search time for an element in the cache of the HTTP proxy. The round-trip time of, e. g., a LAN is typically in the range of 1 ms and of a WLAN in the range of below 10 ms. It is hard to predict how fast an element can be search on a hard drive, but we assume that this is in the range 100 ms. Therefore the *timer1* should be initially set to 150 ms.

6.5 Discussion

This section is discussing the conceptual design of cooperative web access, but lacks some more details that will require experimental results before going ahead with a more fine-grained specification of the system and the algorithms. For instance, the file type heuristics, described in Section 6.3.3, requires measurement of real web traffic of multiple persons that access a variety of web pages.

Another point of interest is how long the idle periods on a resource constrain link are. These idle periods may be very different, depending on the usage profile of the user. For instance, a user reading news pages may have long idle periods, while a user accessing an web-based word processor may have no or only very short idle times.

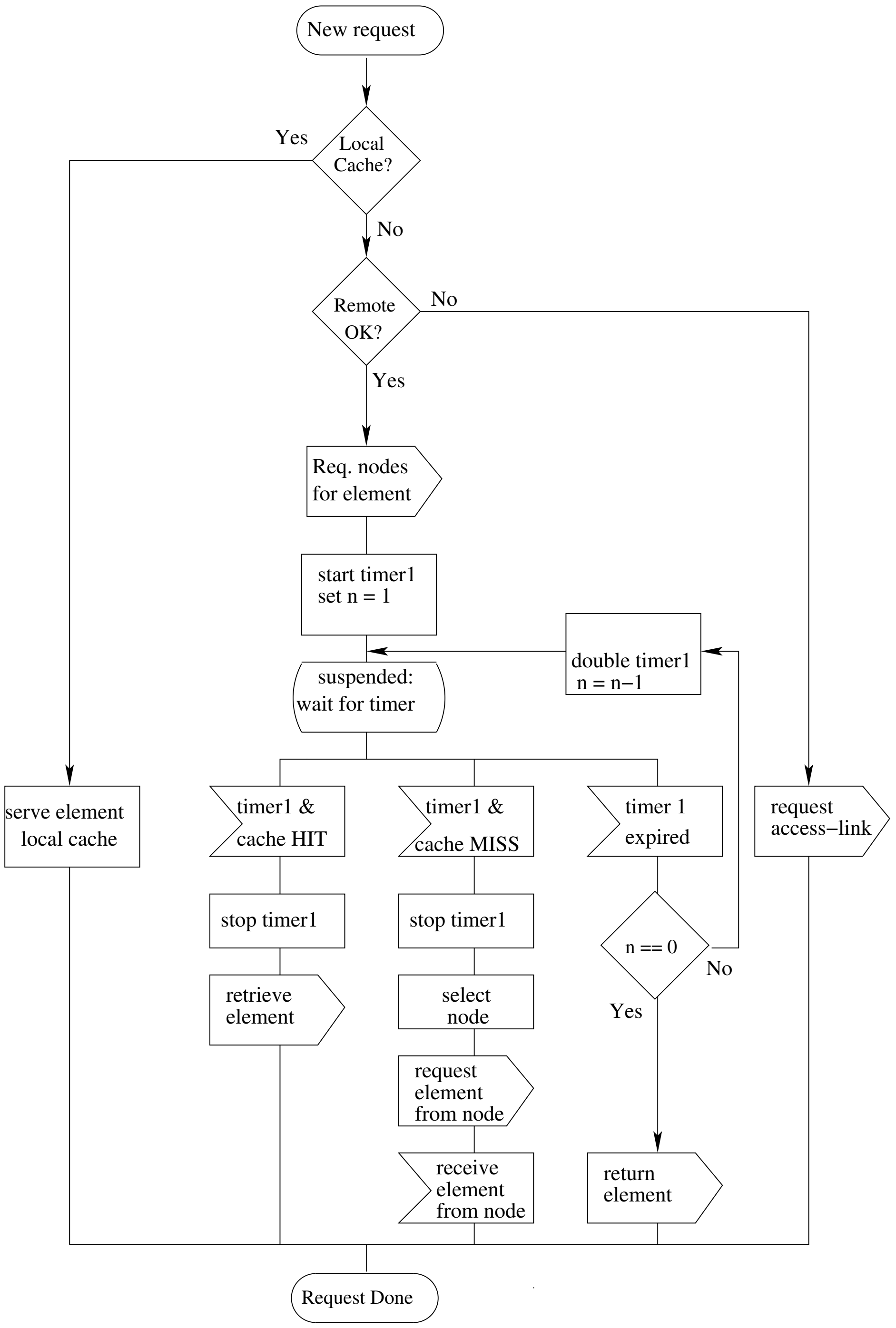


Figure 6.5: Flow chart for cooperative web access requester

The design and implementation of the cooperative web access system requires a larger measurement campaign to grasp the property of web traffic on resource constrained link and also to determine today's web traffic mix of real users. Only these measurements will clarify the possible performance gain of the cooperative web access system. However, this is left for further studies.

7 System Simulator and Performance Evaluation

7.1 Introduction

7.1.1 Why Simulation

A new or changed system can be tested if it works as expected in terms of semantically correctness of the system specification and correctness of implementation, and what the performance of the system is under what condition. There is a number of possibilities how a system can be tested, such as analytical studies of system aspects, simulative studies of a whole system, emulation of the system in a testbed, or deployment in a real network environment. We first elaborate about the system to be evaluated and discuss the pros and cons of the evaluation methods.

It is possible to set up a Cooperative Internet Access system with several computers (the nodes) and the respective access-links with a sufficient amount of money and also human resources. Each access-link requires a subscription to a network operator which occurs costs for the subscription and it also requires the appropriate network access equipment to be available, such as, a mobile phone or a fixed-line modem. The access-links have to be terminated at a computer which are connected via the sharing-link. Furthermore, depending on the scenario, the system has to be placed in a location which presents a resource constrained environment, e. g., in a train.

However, these settings may be hard to fulfill and they may be also limiting the evaluation scenarios to what is possible with the above setting. For the author it was impossible to obtain more than the 2 measured data subscriptions for the mobile wireless network subscription (cf. Chapter 3 for the measurements), simply for cost reasons. However, even with more subscriptions, it remains challenging to run large scale tests, as the whole test equipment needs to be moved to the test locations and operated for the whole runtime of the tests. The tests will always last for the testing period and cannot be shortened, which in turn reduces the chance of performing more or other tests (a time limitation). It is also challenging to operate more than 2 computers in a train for space reasons.

A considerable limiting factor is the inability to change system parameters and to re-run a test under the same condition. For instance, to run and to compare the performance of multiple schedulers under the same condition is impossible if the system is tested in a real environment, as it is not possible to run multiple schedulers at the same time. The outcome of one scheduling process, i. e., which element is retrieved via which access-link, will influence the other scheduling process, as they have to re-use the same access-links or at least the same access network to which the access-links connect. It is also impossible to re-run tests under the same condition but with different system parameters, as for instance, the variation of the estimation precision of the throughput estimator.

These limitations rule out the use of a deployment for the performance evaluation in the evaluation phase, as it poses too many limitations to the evaluation process. Nonetheless, deploying and testing the system under real conditions would be a necessary step before making the system operational, but this is beyond the scope of this thesis.

An environment with more control is the emulation of the system in a testbed. The Cooperative Internet Access system would be implemented and deployed on real computers with emulated access-links. This requires the full implementation of the Cooperative Internet Access system and a specific use case of it (i. e., peer-to-peer video streaming or cooperative web access) and integration in an existing base application, such as, a peer-to-peer video streaming system or in a web proxy. However, this can impose limitations to the evaluation as there are adaptions required to the base application, as for instance, adaption of the network part of the application to be ready to deal with resource constrain environments, such as, dealing with choked TCP connections, due to arbitrary transmission issues not recovered by TCP [123]. Furthermore, the emulation can be performed only in real time and not faster than realtime, posing a time constraint on the evaluation process.

An analytical performance evaluation would consider the number of access-links, the probabilities of achievable throughput, uptimes, and downtimes, in conjunction with the parameters of the chunk diffusion, i. e., the chunk size and the chunk generation frequency. The analytical evaluation would give a first impression out the general possibility of the approach, but it would not reveal details such as, for instance, complexity of the scheduler, numbers of messages to be exchanged on the sharing-link, etc. The analytical evaluation would just not consider the system aspect of Cooperative Internet Access, but solely the chances of transmitting information elements via a set of access-links.

A simulative semantically and performance evaluation avoids the cost issues of the real deployments, as a simulated Cooperative Internet Access system does neither need multiple network operator subscriptions nor the installation and deployment of multiple nodes or access-links. These can be simulated and also the number of access-links and their behavior can be changed or adapted to test the system. This environment allows also the adaptation of system parameters, which cannot be changed in real systems, e. g., the behavior of the access-links). It also allows to change most of the system parameters without reprogramming the real system (might be hard) and also to test the different options, such as, different schedulers or different throughput estimator precision. The simulation increases also the efficiency of the evaluation in terms of required time: each simulation runs needs only a time fraction of the real run time of such a system.

However, a simulative evaluation has the drawback of abstraction of a number of elements. For instance, the access-links are represented by a statistical model which *approximates* the real link behavior and also the assumption of the peer's behavior. We discusses the deficiencies of the simulation model in Section 7.2.6 and also the base assumptions in Section 7.2.

7.1.2 What is Simulated

We simulate the operations of a mesh-based peer-to-peer video streaming system with a varying number of local peers involved and with varying video bitrate. The behavior of the access-links is based on the measurements and the obtained link behavior statistics in Section 3.4.1. Peer-to-peer video streaming in resource constrained environment is a challenging task for retrieving the chunks and retrieving the right chunk at the required time, as the video transmission imposes strict time limits. We evaluate the peer-to-peer video streaming in such an environment in the following sections.

On the other hand, simulating the outcome of the web access of Chapter 6 is considerable harder, as the content, i. e., the composition of the web pages, e. g., whether it is made of plain text, AJAX, or video, will heavily the influence of the result. There is web based content that simply cannot be cached, if the content is completely dynamically generated on a per user base, i. e., there are now shared information elements between users. On the other hand, there may be web based content that is completely static, and has only information elements that can be shared. Furthermore, the web content is changing very frequently probably leaving the possibly gained results useless within a few months. This makes simulation of such a setting extremely hard and also the outcome is only meaningful for the selected setting, but not beyond. The cooperative web access approach should be evaluated in a testbed or in a real deployment, but this is out of scope of this thesis.

7.1.3 Used Simulator Environment

There are numerous simulators in the field of networking, ranging from *generalized network* simulators which simulate IP networks from ground-up to *specialized network* or *application* simulators tailored to the exploration of a particular aspect. For instance, OMNET++ [124] is a *generalized network* simulator while P2PTVSim [125] is a specialized peer-to-peer video streaming simulator. Orthogonal to this is how the simulators operate: there are time-discrete simulators and event based simulators. In a time-discrete simulator, state changes in the system can only occur at fixed, discrete time intervals. In event based simulators state changes occur whenever an event occurs (cf. [126]).

For our purpose we have the following items to be fulfilled by the simulator environment. The simulator must allow the observation of:

- impact of scheduling of information elements;
- impact of transmission on each access-link to fragments of the information elements;
- impact of Cooperative Internet Access system to information element handling in the application on top of Cooperative Internet Access (i. e., peer-to-peer video streaming).

The impact can be at best observed with an event based simulator, as the implementation of the Cooperative Internet Access system itself would also rely on events to control and steer the system (cf. also Section 5.3.5.2 for the various control events in the system). A time discrete simulator would deliver not any insight in the event based behavior of the system and we are not interested in more continuous effects in the system, such as the behavior of the transmission protocols, for instance, as caused by TCP.

Our event based simulator is implemented using the IKR Simlib [127], which an event based simulation environment in C++. We chose the IKR Simlib as it is a lightweight and modular environment which provides many bases classes, e. g., event scheduler, distribution functions, or evaluation classes. The Simlib is well structured and documented, so that the exact behavior of the base system is fully specified and understood. These are the reasons for the author to explicitly not rely on more complex but well established simulation environments, such as OMNET++, where it is can be difficulty to fully understand the behavior of the whole system.

7.2 Simulation System Model

The Cooperative Internet Access framework is based on the assumption that multiple nodes are cooperatively accessing content in the network and that each node brings in its access-link as joint resource, but a single node could have multiple access-links. In fact, for the purpose of the simulation we collapse the group of i nodes into a single node with a number of i access-links, i. e., a single i-times multihomed node. This simplification allows simulations towards observing effects between behavior of access-links and chunk-to-link assignments, but of course neglects certain effects of the sharing-link. The neglected effects are, for instance, packet loss, the transmission delay, and processing times in the network stack of the nodes. However, the sharing-link is not considered as an issues as the link is made of a Local Area Network link which usually provides trouble free networking.

The simulator works on a chunk-level, i. e., it does not consider effects caused by the link-layers, IP-level, or TCP-level, other than the measured effects out of Section 3.4.1. For the simulations, we assume that the delivery of chunks outside the local system runs smoothly, i. e.:

1. the local peers have sufficient remote peers that are able to respond to the chunk requests sent by the local peers
2. the remote peers always have the chunks already in their local chunk buffer if a local Cooperative Internet Access peer requests a particular chunk.
3. all chunks are delivered in-time by the peers and get only delayed or dropped at the access-links. A chunk can get dropped even though TCP is assumed for the transmission, as TCP connections can also be aborted due to network errors.
4. the remote peers do not evaluate the "usefulness" of the local peers: typically each peer evaluates how good its counterparts have served its own chunk requests and removes badly performing peers in favor of better performing peers (e. g., the outcome of the tit-for-tat, see also Section 2.6.6).

Additionally, we do not simulate arriving and leaving local peers, i. e., local peer churn. These assumptions are necessary to evaluate the impact of our proposal on top of existing peer-to-peer

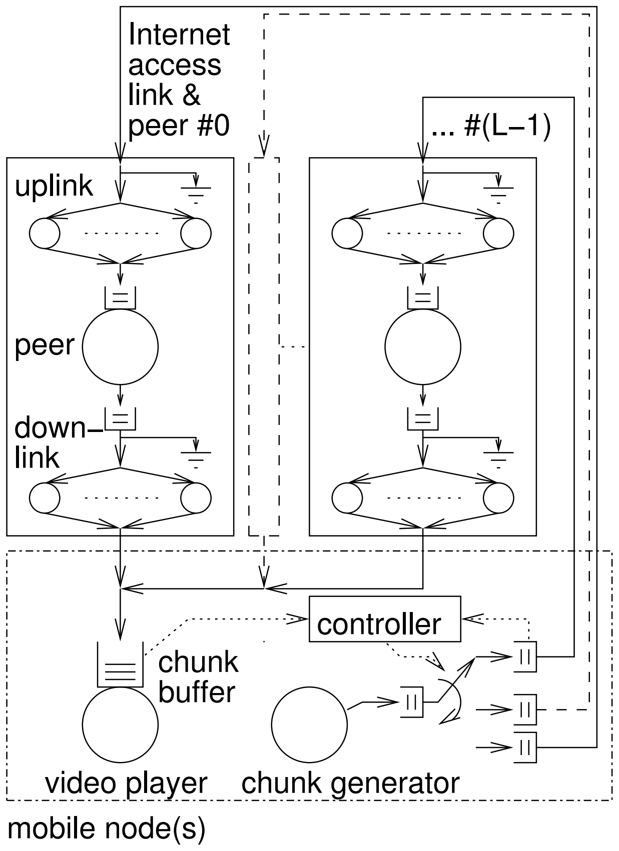


Figure 7.1: Simulation system model

systems and not to interleave with chunk diffusion issues outside the system. A leaving peer would cause an additional signaling overhead and would require the scheduler to re-consider the chunks which had been assigned to the leaving peer. However, for the scheduler there is no difference between a leaving peer and a peer where the access-link link is suddenly failing to deliver any chunk in terms of scheduling the chunks. But the leaving impact would impact the overall achievable throughput.

The system is modelled as depicted in Figure 7.1 in such a way that there is a chunk generator, a single controller, L access-links ($0 ... L-1$), the local chunk buffer, and a video player.

The next sections detail the implementation of the various elements, such as the modelling of the access-links is described in Section 7.2.2, the sharing-link model is described in Section 7.2.3, the video player is described together with the chunk buffer in Section 7.2.4 and the simulated schedulers are described in Section 7.2.5.

It should be noted, that the only the scheduler is changed during the simulations scenarios, i. e., all other elements stay the same.

7.2.1 Chunk Generator

The chunk generator is shown as part of the simulated node in Figure 7.1 as it is integrated in this way in the simulator. In a real peer-to-peer system the chunk generator resides in the source of the chunks, where the video is processed and packed in the chunks. The chunk generator gives also the information about which chunks are available to the scheduler, as compared to a peer-to-peer system implementation where a particular peer has to learn the available chunks from its neighbor peers via the buffer maps (cf. Section 2.6.3).

The chunk generator in the simulator creates chunks with a fixed size *ChunkSize* at a given interval depending on the selected video transmission bitrate B_v, as described in Equation 5.1 in Section 5.1.1. However, the generated chunks do not carry any video and are only used to simulate the transmission of the chunks.

7.2.2 Modelling the Access-Links

The Cooperative Internet Access simulator requires a link model to simulate the behavior of the access-links which have, together with scheduler, the major impact on the system's behavior. Each access-link is modelled by an uplink (from the node to the Internet) and a downlink part (from the Internet to the node) with a peer in between, as shown in Figure 7.1. A chunk that is placed by the controller in the request queue of an access-link is requested from a remote peer by sending to via the uplink part to that peer. The peer responds with sending the requested chunk to the node via the downlink part. A chunk request the response (sending the chunk) can basically get lost on either parts of an access-link which is denoted by the *ground* symbol (known from electronics) in the figure.

7.2.2.1 Uplink Model

The actual model of the uplink part is an ideal link following the assumptions in Section 7.2, i. e., each chunk request is transmitted to the remote peer and also successfully served by that peer. A chunk request in the simulator causes that the chunk is placed in request queue of the access-link and is directly handled by the downlink part, if the chunks is due for transmission, otherwise it is queued at the access-link.

7.2.2.2 Downlink Model

The measurement results presented in Section 3.4.1 and discussed in Section 3.5 are the basis for our UMTS downlink model used in the simulator. We use the measured data instead of synthetic data to be as close as possible to real networks with our simulation. Section 3.4.1 shows that the measured data leads to the conclusion that the measured UMTS links exhibit a negative exponential behavior for throughput and the up/downtime distribution on the downlink. The mean values λ_2 out of Table 3.4 are used for the achievable bandwidth λ_{bw} mean value, for the link up-time λ_{up} mean value, and for the link downtime λ_{down} mean value of the particular negative exponential distribution functions. Table 7.1 shows the λ values per operator used in

Operator	λ_{bw} in $\frac{s}{kbit}$	λ_{up} in s^{-1}	λ_{down} in s^{-1}
Operator 1	0.00898	0.14410	0.0326549
Operator 2	0.018644	0.234367	0.0296241
(Operator 3)	0.01	0.1667	0.5
(Operator 4)	0.01	0.1818	0.3571

Table 7.1: Configuration Parameters of Link Model

the simulator's link model. We also extrapolated additional values for two further operators, to model to total number of 4 independent operators in Germany (see also Section 3.5 for the number of operators), based on the results of the 2 measured operators. We always assume all access-links to be disjoint even if they belong to the same operator, i. e., links of the same operator share the same statistical properties other than the probability of failing at the same time.

For each operator's set of distributions, we use a negative exponential distribution function in the simulator: equation 7.1 denotes the maximum achievable throughput if a link is up while equation 7.2 denotes the duration of the uptime and equation 7.3 the downtime. The times t_i denote the time points in Figure 7.2.

$$\hat{\Theta}_i = \lambda_{bw} e^{-\lambda_{bw}} \tag{7.1}$$

$$t_{up} = t_{i3} - t_{i0} = \lambda_{up} e^{-\lambda_{up}} \tag{7.2}$$

$$t_{down} = t_{i4} - t_{i3} = \lambda_{down} e^{-\lambda_{down}} \tag{7.3}$$

Figure 7.2 show the timing behavior of the downlink model used in the simulator. A link in the model can either be up (with throughput) or down (without throughput) and belongs to one operator (i. e., Operator 1, Operator 2, Operator 3, or Operator 4). At simulation start-up, all links start in state down. At the point of time when a link changes from down to up, it will stay up in the time period $[t_{i0} : t_{i3}[$, with a ramp-up phase $[t_{i0} : t_{i1}[$ where the achievable throughput increase from $\Theta(t_{i0}) = \frac{\hat{\Theta}_i}{4}$ to $\Theta(t_{i1}) = \hat{\Theta}_i$. The increase in the period $[t_{i0} : t_{i1}[$ is stepped with $\frac{\hat{\Theta}_i}{3}$ every $\frac{t_{i1}-t_{i0}}{3}$ where $t_{i3} - t_{i2} = t_{i1} - t_{i0} = \frac{1}{10} t_{up}$ and vice versa for the decrease in the time period $[t_{i2} : t_{i3}[$. The maximum achievable throughput $\hat{\Theta}_i$ is achieved in the time period $]t_{i1} : t_{i2}]$, where $t_{i2} - t_{i1} = \frac{8}{10} t_{up}$.

After the time that the link is up it changes to down for the time period of t_{down}. After t_{i4} is elapsed, the link repeats the up status as described above. This change between up and down continues periodically throughout the simulations. The characteristics of the achievable throughput in Figure 7.2 is a model of the behavior of the achievable throughput that to fit the real behavior of a TCP connection on UMTS link: the achievable throughput increases if the quality of the radio channel increases (e. g., a mobile terminal is approaching closer to the cell tower) and fades out if the quality of the radio channel decreases. The achievable throughput

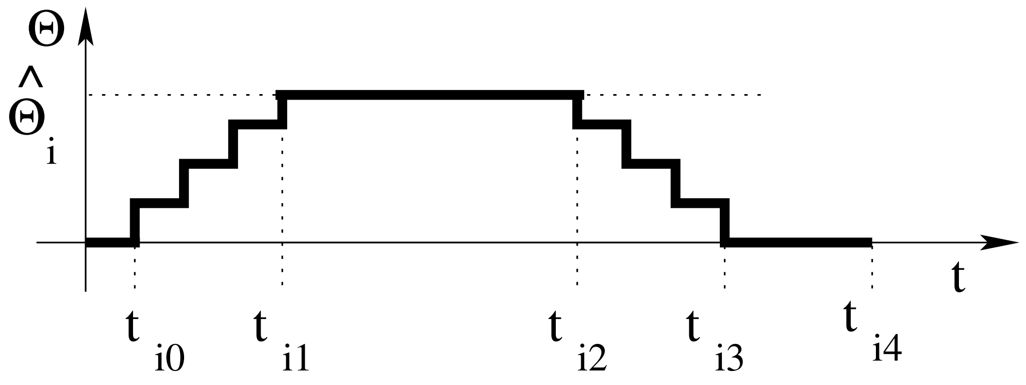


Figure 7.2: Behavior and Timing of Link Model

will not follow this pattern in all cases. At some occasions the achievable throughput may be cut-off sharply, if the radio link fails completely, or the throughput may vary in certain ranges without going fully to 0. These circumstances are not considered in the link model.

The simulator uses the 4 different operators parameters of Table 7.1 to create access-links for the simulation runs. The operators are assigned round-robin according to the number of links, i. e., each new link will be taken from a different operator. For instance, a total of 5 links will results in 3 links assigned from operator 2, operator 3, operator 4, and 2 links assigned from operator 1.

7.2.2.3 Transmitting Chunks

A chunk that should be transmitted is passed to the particular access-link and fragmented in parts of 1000 bytes. For each fragment, the access-link calculates the transmission duration based on the current achievable throughput $\Theta(t_i)$ and fragment size. The fragment is hold in the link until transmission duration is elapsed and afterwards it is delivered to the mobile node. The particular fragment is immediately passed to the node, if the link is currently down, i. e., either the link has been down before the new chunk arrived or the link went down during the transmission. The link is busy with the transmission of a particular chunk until all fragments have been delivered to the node or the link is down. It can only accept a new chunk for transmission, if there is no chunk in transmission right now.

The access-link implements also the measurement module for each particular link. In each fragment, the current measured achievable throughput is noted. This information is used by the mobile node to estimate the link behavior and to detect whether the link is up or down. Furthermore, the access-link includes full information about the future behavior of the achievable throughput. This information is later used by the *theoretical* scheduler (see Section 7.2.5).

7.2.3 Modelling the Sharing-Link

The sharing-link is considered to be an ideal link, e. g., without packet loss, and without any performance limits in terms of number of packets per second or throughput. We make these assumptions to evaluate if the given system model, e. g., number of links, behavior of access-links, scheduler, and the resulting need to exchange chunks and control messages via the sharing-link can actually be transmitted via existing network technologies, such as, WLAN.

The sharing-link is used to in two logical directions: from the controller to the local peers and from the local peers to the controller. The controller assigns chunk request queues to the local peers, where the local peers reply with status reports of their measured achievable throughput, their current status of the assigned request queues, and with chunks that were successfully retrieved. The simulator does not transmit these messages, but notes the frequency of occurrence for messages from the controller and to the controller.

7.2.4 Video Player and Chunk Buffer

Chunks that arrive at the mobile node are stored in the local chunk buffer. This buffer is used by the controller to judge which chunks are already locally available at the node, which chunks are requested and which are still needed. The chunk buffer is shared with the video player.

The video player is actually not displaying any video, but is taking care of the sliding window operations and the decision which chunks can be omitted (under the assumption that Forward Error Correction is used). The simulator uses the sliding windows mechanism, as described in Section 5.2.1. The trading window has width of 64 chunks and a resulting sliding window width of 32 chunks. These values are taken from PULSE [76].

The video player is allowed to jump ahead one chunk, if there is no progress in the sliding window for 5 s, as the system was not able to retrieve the needed chunks.

7.2.5 Simulated Schedulers

The core of the simulator is the controller process with the chunk-to-link assignment scheduler. The simulator implements three schedulers, the deadline based scheduler with throughput estimation, as described in Section 5.2.2, a *theoretical* and a *round-robin* scheduler.

The *theoretical* scheduler lets the controller know the future achievable throughput for each link. This knowledge allows the controller to obtain the best possible set of links to transfer all chunks in-time. While this approach cannot be implemented in a real system, it yields the theoretical upper bound of efficiency. This scheduler fits the chunks to the access-links so they can be retrieved, unless there is no access-link that can deliver any chunk within its uptime.

The *round-robin* scheduler simply picks the next link that is not busy with another chunk and is considered being up. This scheduler does not consider the current achievable throughput of the access-links.

The deadline based scheduler uses the throughput estimation scheme described in Section 5.3.2 and the scheduling algorithm 1 defined in Section 5.2.2 The throughput estimator uses the measured achievable throughput reports of the access-links, but allows to change the precision of the estimates. The deadline scheduler uses a fall back strategy, if it cannot find any suitable access-link for any chunk. It simply assigns the chunks on a round robin base, in order to keep the system loading chunks and to keep the measurements running.

7.2.6 Deficiencies of the Simulation Model

Simulators are always based on assumptions and on a restricted environment that is a reproduction of the real environment, and a simulator introduces meanderings between the reality and the simulator. These meanderings are unavoidable when testing and evaluating a system in a simulator, but user of the simulator and ”consumers” of the simulator results should bear this in mind. This also hold true for the Cooperative Internet Access simulator and this section discusses the deficiencies of the peer-to-peer simulator:

TCP issues We’re using TCP and there are some issues with TCP in mobile environments: hick-ups, as TCP does not recognize that the link is back; TCP performance enhancement proxies (PEPs) that buffer packets and deliver them in a burst, even though the simulator considers these as lost. Not considered are TCP issues, such as in [128].

Variable sized chunks real video streaming systems require that the video is delivered in a constant pace in the time domain. However, this results in variable sized chunks, as the video codec will generate at some points full video frames, while at other times only difference information to the full video frame. This fact will result in chunks that are not of equal size.

Uplink issues The uplink part does not models the upload of chunks and chunk requests from the local peer to the remote peers, and it is thus not considering resulting issues caused by the uplink.

Sharing-link issues issues causes by the sharing-link are not considered. However, in reality, the sharing-link will also cause issues, especially if a wireless link is used.

7.3 Evaluation Metrics

We evaluated the simulation in terms of chunk retrieval quality P and the *minimum initial buffering time for continuous playout*. The chunk retrieval quality P indicates whether the system was able to retrieve the required chunks or if the system stalled if there were no chunks available. P is calculated in this way, that the chunk playout at the source ($C_{playout}(t)$) is taken as reference and all results of the schedulers ($C(t)$) are normalized to the source:

$$P_{video} = \frac{\int_0^{t_{end}} C(t)dt}{\int_0^{t_{end}} C_{playout}(t)dt} \tag{7.4}$$

A value of P equal or close to 1 indicates that almost all chunks were successfully received and that the video can be rendered. A value of $P \ll 1$ indicates that most of the chunks were not received by the system and that the video cannot be rendered.

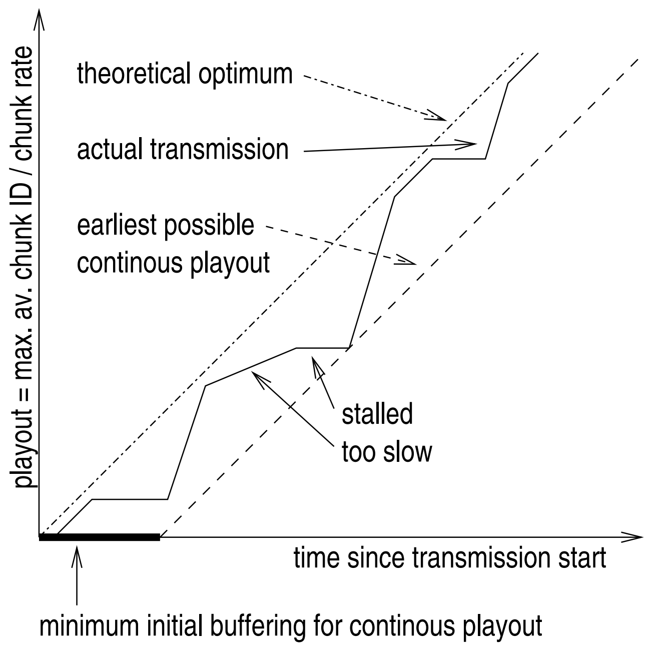

playout = max. av. chunk ID / chunk rate
theoretical optimum
actual transmission
earliest possible continous playout
stalled
too slow
time since transmission start
minimum initial buffering for continous playout

Figure 7.3: Minimum initial buffering time for continuous playout

P as such is not sufficient to judge whether the chunks have always arrived on-time, i.e., a scenario could let the whole system stall for hundreds of seconds and then obtain the chunks very bursty, as illustrated in Figure 7.3. Therefore, we define a second measure to judge the effectiveness, i.e., the *minimum initial buffering time for continuous playout*. The dash-dotted line in Figure 7.3 indicates the availability of video chunks at the source vs. the time since the start of the transmission. In an ideal system without transmission delays, these chunks could be played out at the destination nodes at the same time. However, depending on the availability of the access-links and depending on the scheduling strategy, chunks arrive with varying delays at the considered destination node (solid line). The dashed line indicates the earliest possible playout of the stream. That is, the chunk buffer at the receiving node will be empty at least once, but there will be no buffer underruns and consequently the video will not stall. The bold section on the x-axis denotes the corresponding initial buffering time. It should be noted that this time can only be determined retrospectively as it requires knowledge about the whole transmission process.

7.4 Simulation Results

This section presents and discusses the simulation results based on the simulator implementation outlined in Section 7.2. The simulations used video bitrates B_v in the range between 300 kbit/s and 600 kbit/s, as most of today (2010) peer-to-peer video streaming systems use bitrates in that range (cf. [5]). The chunk size is set to 12.5 kbyte. Each simulation run simulates a time of 1800 s and and the results are the averaged results out of 20 runs.

We use the evaluation metrics out of Section 7.3 to judge the different schedulers and their output. The results are presented separated for the access-link and sharing-link.

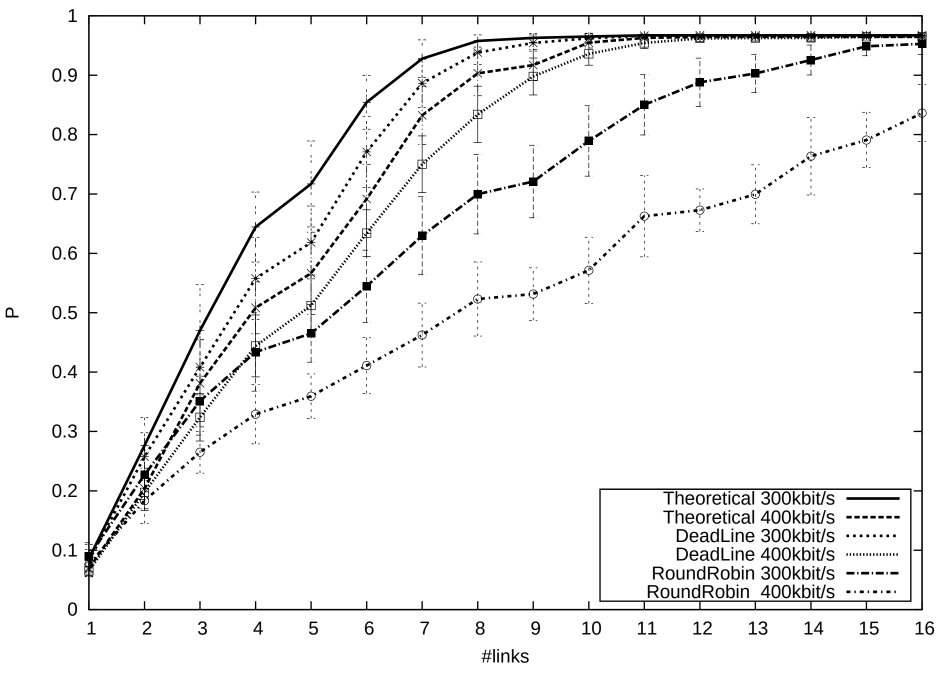


Figure 7.4: Chunk retrieval quality for $B_v = 300kbit/s$ and $B_v = 400kbit/s$ – with precision set to 45 %

7.4.1 Access-Link Related Performance

This section evaluates the performance of the access-links when using different schedulers and varying parameters, such as the bitrate of the video and the likelihood that the estimated throughput of the throughput estimator is correct. The throughput estimator in Section 5.3.2 can operate with an precision of 45 %, i. e., the estimated throughput for the next 1 second is in 45 % the cases correct, given that the scheduler obeys, that the estimation result has an error of $\pm 15\%$. The simulator allows to change this likelihood to see how this impacts the results.

The number of access-links varies between 1 link and 16 links. A single link shows the to expected behavior for a node not equipped with Cooperative Internet Access, i. e. the behavior of an off-the-shelf peer. 16 links is a good upper bound in terms of number of nodes, as this results in 4 links taken of 4 different operators on peers that are in physical proximity. Adding more peers would result in adding more UMTS links. However, adding more and more peers will not increase the system capacity, as these links are connected to the same operators, probably using the same cell or the same backhaul. UMTS cells have an upper capacity bound in terms of available bitrate, but also in terms of number of concurrent data users, and adding more access-links will result in a higher resource sharing of the available throughput but not an increase anymore.

7.4.1.1 Impact of Video Bitrate

We first present the simulation results for different video bitrates, while keeping the precision of the throughput estimator at 45 %. Figure 7.4 and Figure 7.5 show the chunk retrieval probability P for varying bitrates with the theoretical, the deadline based, and the round-robin schedules, as function of the number of available access-links.

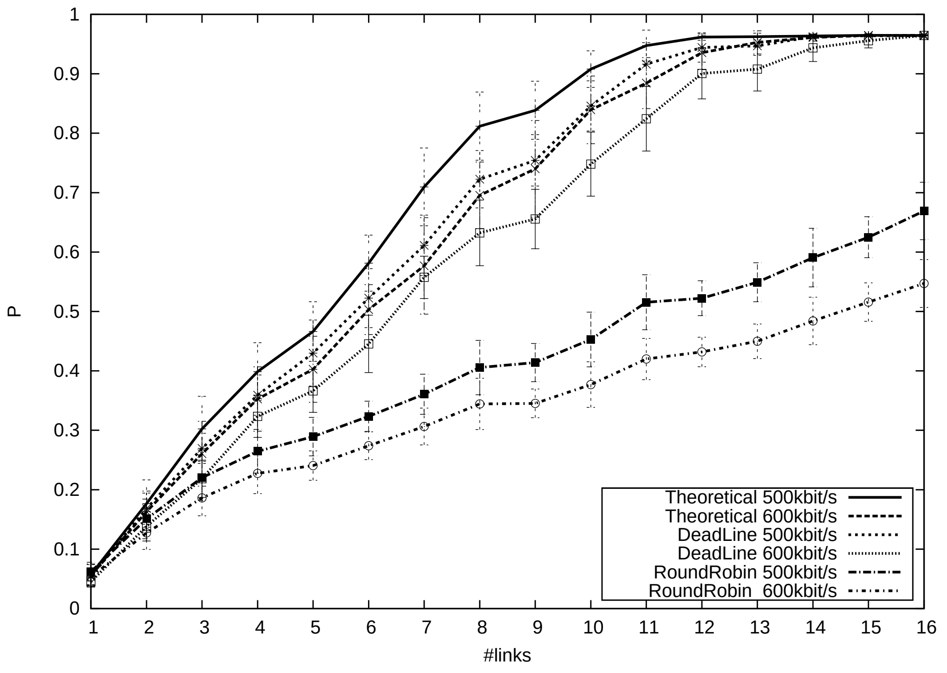


Figure 7.5: Chunk retrieval quality for $B_v = 500kbit/s$ and $B_v = 600kbit/s$ – with precision set to 45 %

The theoretical scheduler yields the upper bound that the system can reach, if the exact behavior of each link is known in advance. The theoretical scheduler, but also the deadline based, actually never reach a value of $P = 1$, which is due to the used buffer in the system. The minimal initial buffer time also never reaches a value of 0. The simulator uses a buffer phase in the beginning to overcome transient effects, but this chunks in the buffer are not considered in the simulation results, causing the effects described in the evaluations.

All schedulers start in the figure at the same origin, in the case of only a single access-link, i. e., which is the case of today when a regular peer is trying to retrieve the video via one UMTS link. A value of $P < 0.1$ shows that the peer is unable to retrieve less than 10 % of all required chunks within the complete simulation time of 1800 seconds. In general, the chunk retrieval quality increases with an increasing number of access-links. However, only the deadline based scheduler is close to the theoretical scheduler, while the round-robin is clearly outperformed.

Figure 7.6 and Figure 7.7 show the minimal initial buffer time for the 3 schedulers. We assume a reconnection threshold of 200 seconds of the peer-to-peer streaming system. This reconnection threshold indicates the time between the current playout point the current system video playout point at the source (the nodelag) where a peer would stop fetching chunks at its current window position, but not in the simulator, as we want to show the full range of results. The peer will restart the chunk retrieval process in order to obtain a smaller node lag. The minimal amount of access-links required to stay below this reconnection threshold is 7 links for a video bitrate of 300 kbit/s (Figure 7.6). For a video bitrate of 600 kbit/s this point is reached with 12 access-links.

These results suggest that the video bitrate should stay below 500 kbit/s to obtain good results with a number of access-links that is feasible to gather. In a real scenario, the number of access-

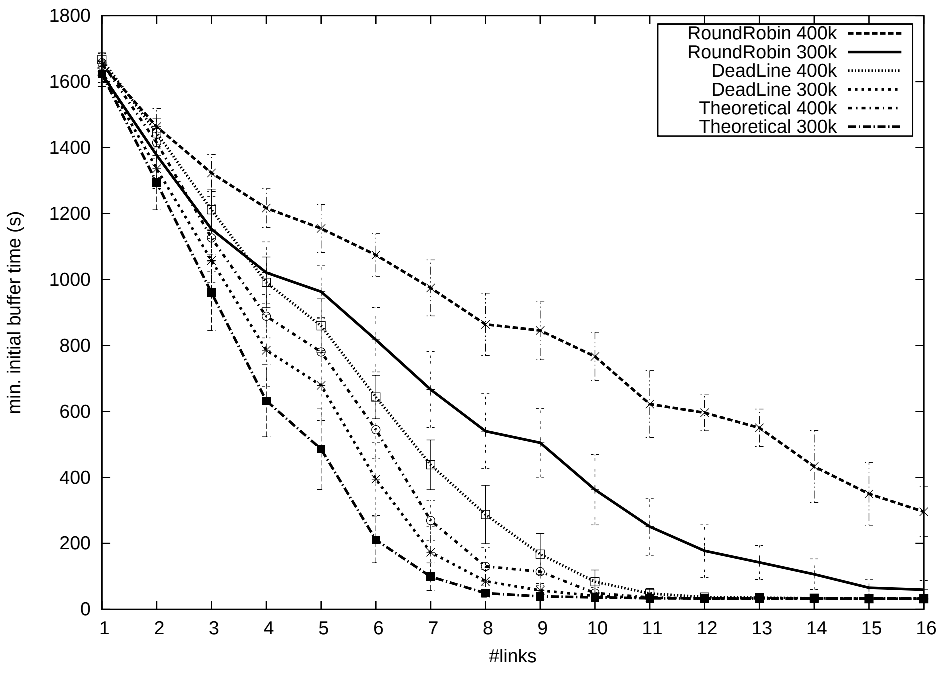


Figure 7.6: Min. initial buffer time for $B_v = 300kbit/s$ and $B_v = 400kbit/s$ – with precision set to 45 %

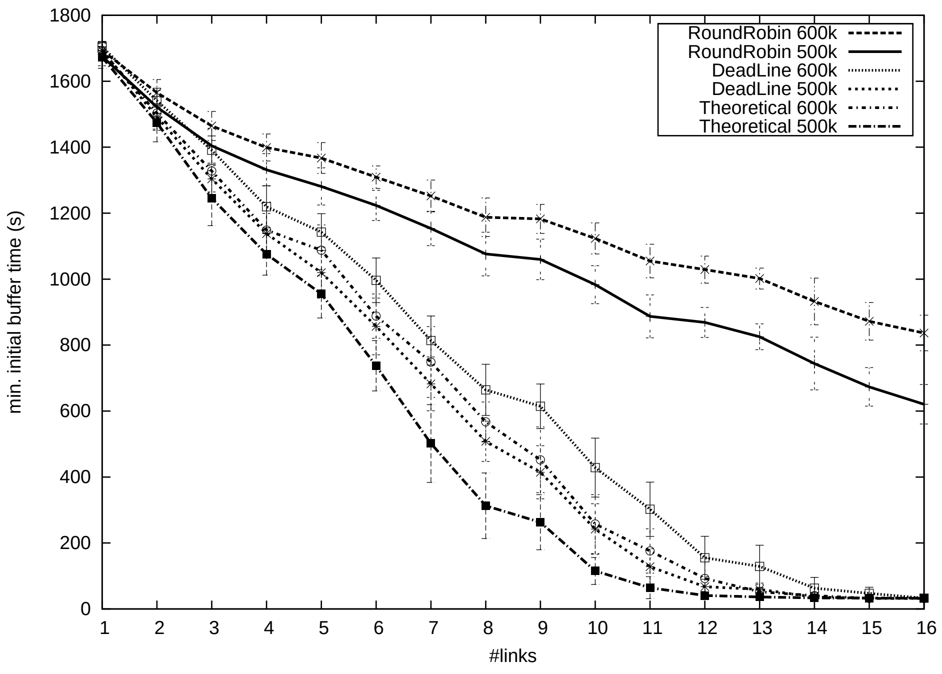


Figure 7.7: Min. initial buffer time for $B_v = 500kbit/s$ and $B_v = 600kbit/s$ – with precision set to 45 %

links corresponds to the number of people that is in physical proximity and that is willing to see the same content. It is possible to see the video content with $B_v = 300kbit/s$ with $P \approx 0.9$ and a minimal initial buffer time of 200 s, if there are 7 access-links available. The required number of access-links increases to a required minimum of 9 for a video bitrate of $B_v = 400kbit/s$.

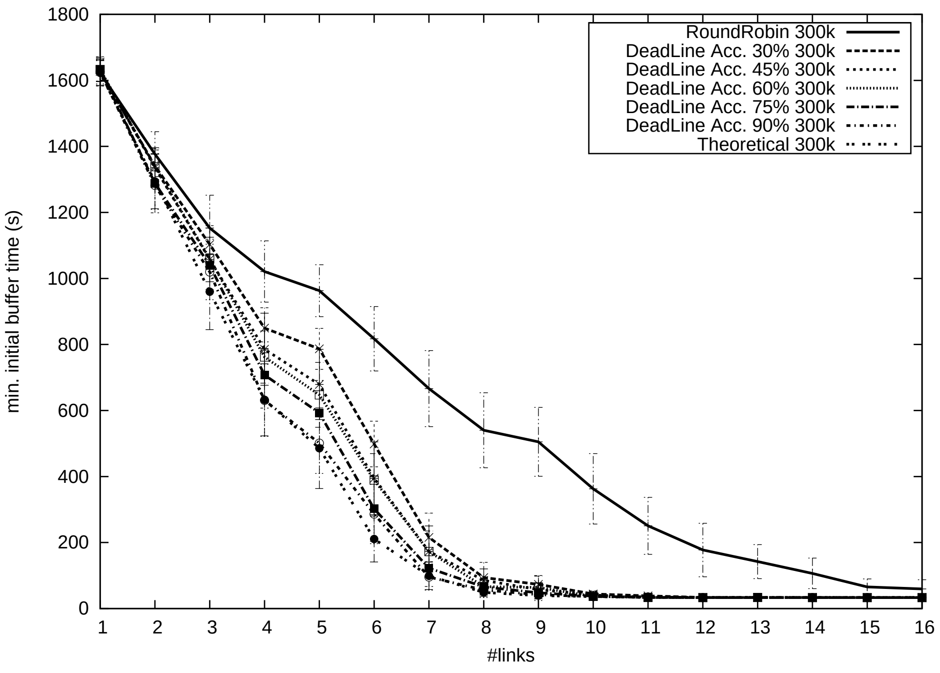


Figure 7.8: Min. initial buffer time for $B_v = 300kbit/s$

It should be noted that a minimal initial buffer time larger than 200 s seems to be impractical, as the peer-to-peer video streaming would just lag too much behind all other peers. The risk is high that the mobile peers will not find any other peer that is still interested in the chunks at that point of time.

However, it may be difficulty to find 7 or more people in one location which are watching the same content, other than in long-distance trains. For instance, an ICE high-speed train has a total capacity of 460 seats and half of them being in average occupied (the average utilization is 47%, see [91]). It may be likely to find 7 to 10 peer interested in the same content.

7.4.1.2 Impact of Estimator precision

One key element of the deadline based scheduler is the throughput estimator described in Section 5.3.2 and the usage of the estimation probability in the Algorithm 1. This section investigates the impact of the estimation probability to the chunk retrieval quality and to the minimal initial buffer time. We have simulated a range of accuracies for likelihood of the throughput estimation, starting from being correct in 30 % of the cases to being correct in 90 % of the cases.

The throughput estimator in Section 5.3.2 yields an precision of 45 % (or below) and in fact the simulation shows acceptable results with 30 % or 45 % precision for all video bitrates. For $B_v = 300kbit/s$ in Figure 7.8 a number of 7 access-links is good enough to stay below 200 s minimal initial buffer time. An even better precision will not increase the gain in time. The chunk retrieval quality in Figure 7.9 is $P \approx 0.9$ for 7 links. For $B_v = 400kbit/s$ 9 links are required to again meet the goal of below 200 s minimal buffer time, where it is notable that all

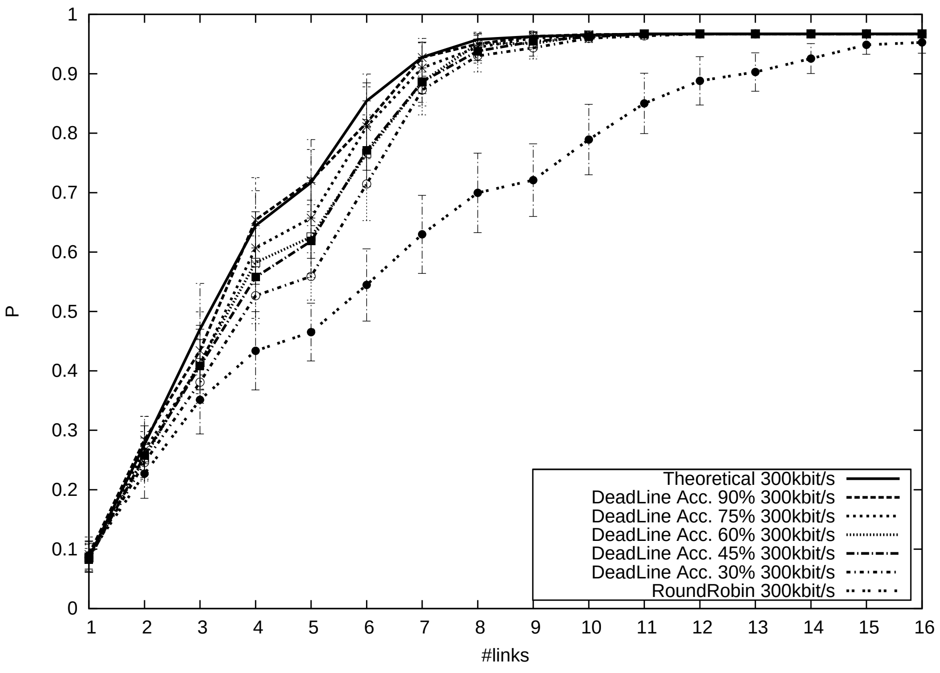


Figure 7.9: Chunk retrieval quality for $B_v = 300kbit/s$

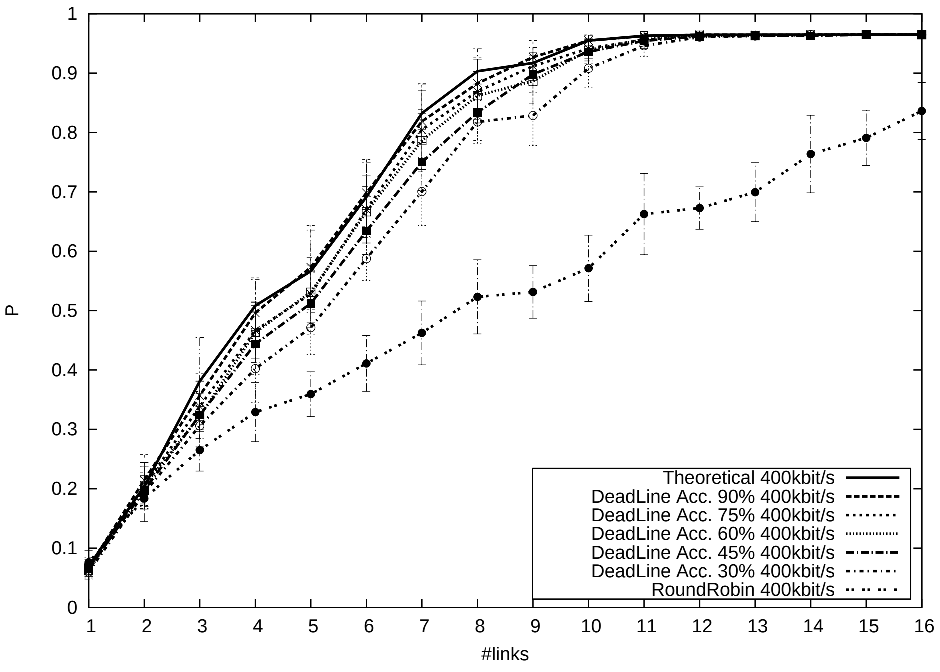


Figure 7.10: Chunk retrieval quality for $B_v = 400kbit/s$

the graphs for all precision settings collapse for more links (Figure 7.11). This break even points is at 11 links for $B_v = 500kbit/s$ (Figure 7.13 with $P \approx 0.9$ Figure 7.12). For $B_v = 600kbit/s$ it is required to gather 12 links to obtain a similar result (cf. (Figure 7.15 and Figure 7.14).

It is worth noting that the round-robin scheduler can reach reasonable results in terms of the chunk retrieval quality, i. e., for 16 links, between $P \approx 0.95$ for $B_v = 300kbit/s$ to $P \approx 0.55$

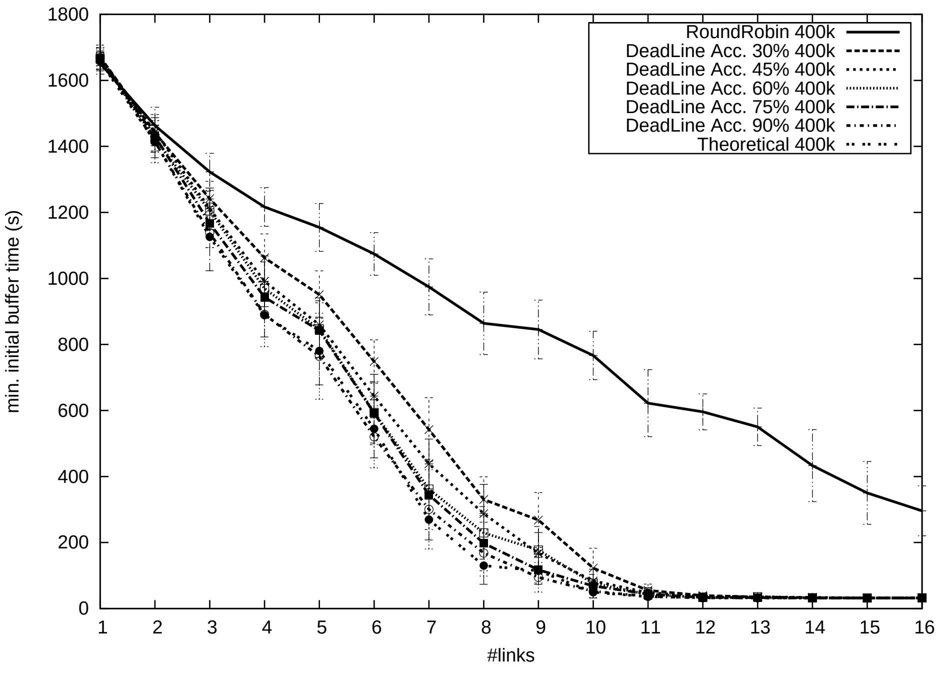


Figure 7.11: Min. initial buffer time for $B_v = 400kbit/s$

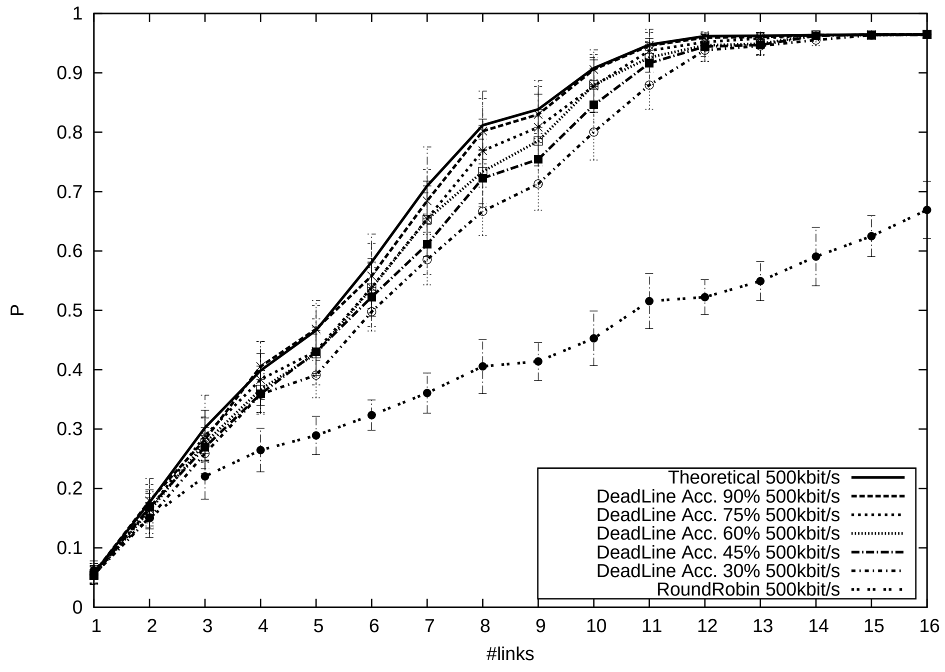


Figure 7.12: Chunk retrieval quality $B_v = 500kbit/s$

for $B_v = 600kbit/s$. However, the weakness of the round-robin scheduler is apparent when considering the minimal initial buffer time: it is still fair for $B_v = 300kbit/s$ with almost as good as the theoretical and deadline based scheduler, but for any greater video rate the round-robing scheduler cannot catch up, e. g., with approximately 300 s for $B_v = 400kbit/s$ and 16 links.

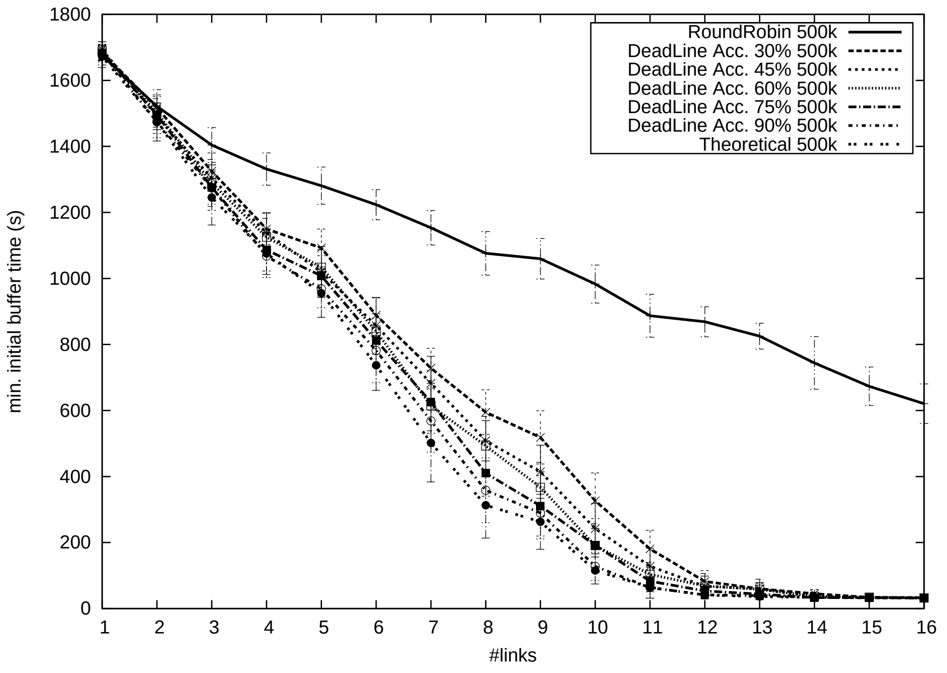


Figure 7.13: Min. initial buffer time $B_v = 500kbit/s$

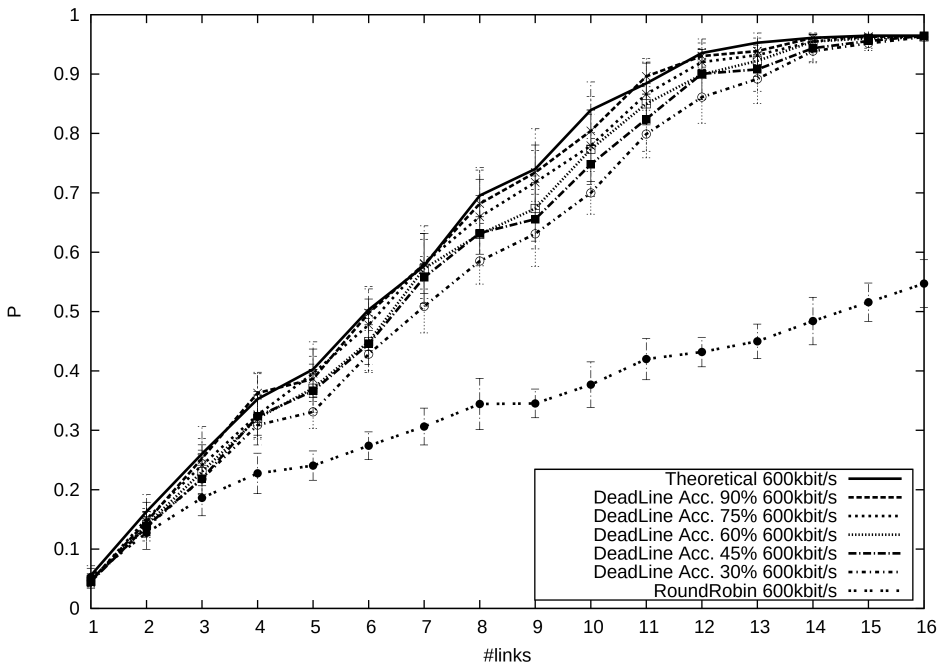


Figure 7.14: Chunk retrieval quality $B_v = 600kbit/s$

The simulation results already show the standard deviation for each data point: The variances of each curve often intersect, even between precision values that are far appart. For instance, in Figure 7.15 the curves for the accuracies 45 %, 60 %, and 75 % overlap for most number of links. This hints to the fact, that estimator precision is one factor, but the scheduler algorithm already copes well with the difficult situation of assigning chunks to links. The scheduler typically assigns the chunk to several access-links, as none of the links has a high likelihood to

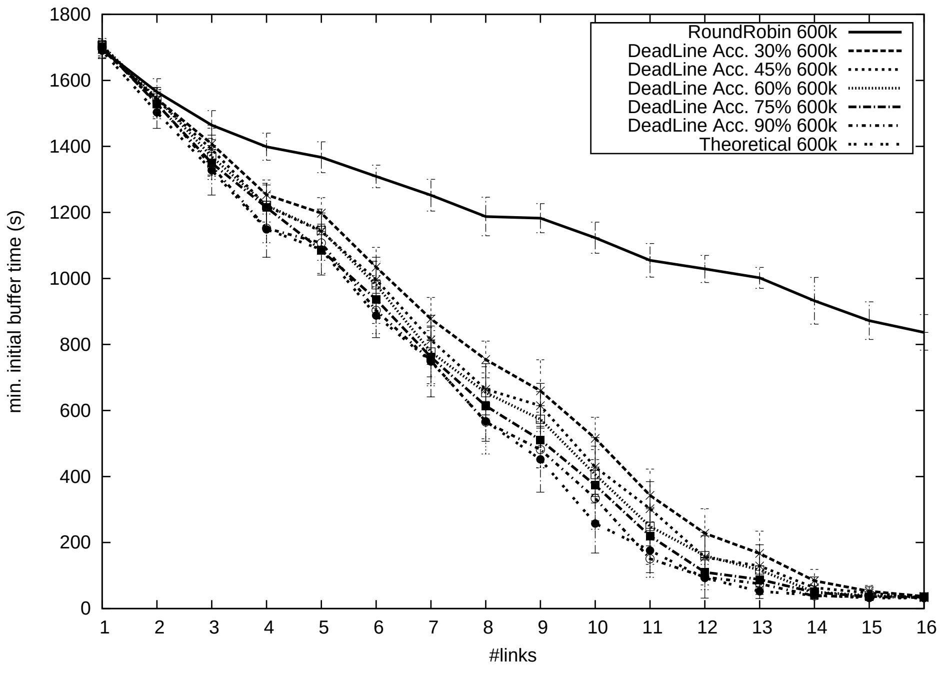


Figure 7.15: Min. initial buffer time $B_v = 600 kbit/s$

deliver any chunk within its deadline. However, this assumes that the access-links are disjoint in their behavior (see also Figure 4.9), as in the simulator.

However, the estimator precision and also the volatileness of the environment impact the scheduler. A lower estimation precision requires the algorithm to perform multiple assignment rounds for each chunk, as it has to assign a single chunk multiple times to various access-links. On the other hand, the scheduler has to re-assign more chunks, if the environment is really volatile, i. e., the achievable throughput is changing on very short time scales.

All figures show an asymptotic behavior for an increased number of access-links, as more links increase the likelihood of transmitting all required chunks in the right time to the local system.

7.4.2 Sharing-Link Related Performance

This section evaluates the impact of the Cooperative Internet Access system on sharing-link. It is assumed that the sharing-link is a shared medium that supports multicast. A candidate for the sharing-link is, for instance, WLAN. The support for multicast is necessary, as the information element redistribution described in Section 4.4.5 is based on that the retrieved chunks are multicasted to all participating local nodes.

For the simulation evaluation, we differentiate between the originator of a message within the Cooperative Internet Access system: messages are either send by the controller process or by the local peers. The controller captures the information for every particular peer, i. e., the request queue for each peer and the current status of the virtual chunk buffer, in a single message that is multicasted to all participating peers (see Section 5.3.5.2) for more details on these messages.

The current implementation of the controller-process in the simulator exhibits this behavior:

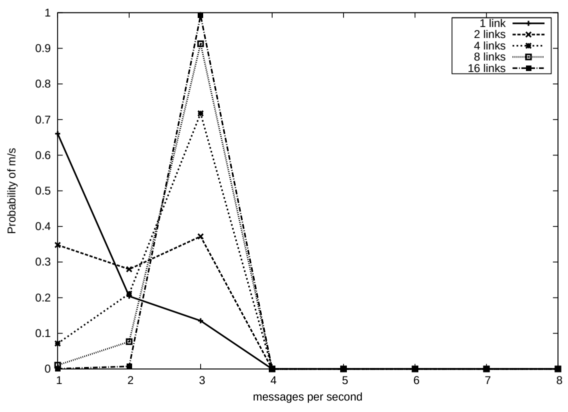
1. the peak number of messages is proportional with the chunk generation frequency f_C, but it is not proportional with the number of local peers. One message sent by the controller is multicasted to all peers, thus decoupling the rate from the number of peers.
2. the controller-process is evoked at the time of a chunk creation, resulting in scheduling often only a single, new chunk.
3. the better the throughput estimation, i. e., a larger precision, and the more access-links in the system, the more often the controller has to act. We are taking the 90 % precision as example: each chunk is assigned only once to a request queue, as the precision exceeds the required threshold of $A_C = 70\%$ For a small set of access-links, i. e., below 4, each access-link has typically a request queue of several chunks, keeping the link busy with retrieving chunks. However, for a larger access-link set, above 4, each request queue has only a few chunks or only a single chunk assigned. This requires the controller to assign new request queues very frequently.

We show exemplary results for $B_v = 300kbit/s$ as the lowest examined video bitrate, and $B_v = 600kbit/s$ as the highest examined bitrate, with a throughput estimation precision of 45 % and 90 % to discuss the impact on the sharing-link. The results show the outcome of a particular simulation run that is representative for Cooperative Internet Access. We do not present average values as it is more important to discuss the actual number of messages per second for a given run.

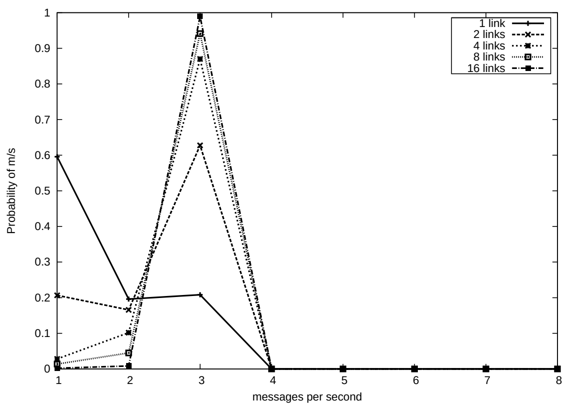
The implementation of the sharing-link is limited in the sense that events causing a message are tightly coupled with the creating of new chunks, i. e., the controller process is evoked each time a new chunk is generate. A chunk is generated at the frequency f_C (see equation 5.1 in Section 5.1.1). A peer in real peer-to-peer video streaming system cannot know when a new chunk is generated at the source, but instead, it must rely on the chunk status information that it receives from its neighboring peers via the chunk bit maps. However, the modelling of the sharing-link gives an upper bound of how many messages are exchanged, if the controller would immediately know the creation of each chunk and the chunk would be immediately available at the remote peers.

7.4.2.1 Messages sent from the Controller

The controller sends messages to all peers to assign or replace request queues or to report about the current status of the virtual chunk buffer. Figure 7.16 show the probability of messages sent per second by the controller process to the peers for $B_v = 300kbit/s$ with 2 different estimation accuracies. The peak is at 3 messages per second for more than 1 1ink in use. The peak corresponds to the chunk generation frequency $f_C = 3s^{-1}$. There are only marginal differences between the precision of 45 % and 90 %, as in this situation the number of links required to obtain all chunks and keep a small nodelag, is relative low. The situation is not that challenging, as compared to higher video bitrates The number of chunks to be retrieved per second is relative low (e. g., 3 chunks per second) and therefore the request queues are also rather short assigned

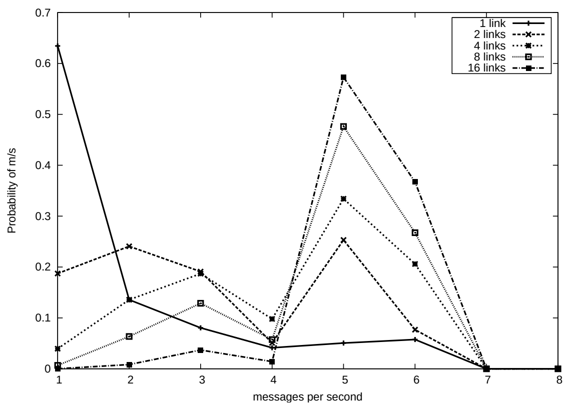


(a) precision 45 %

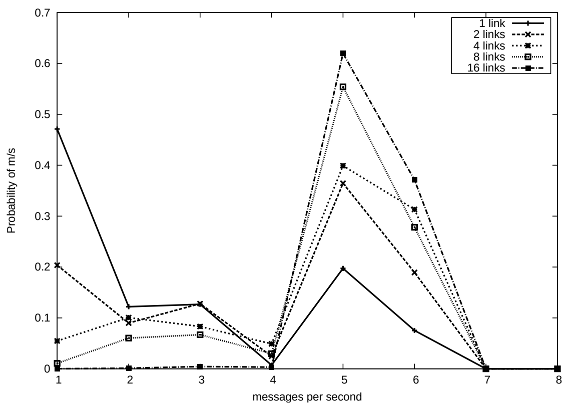


(b) precision 90 %

Figure 7.16: messages from controller for $B_v = 300kbit/s$



(a) precision 45 %



(b) precision 90 %

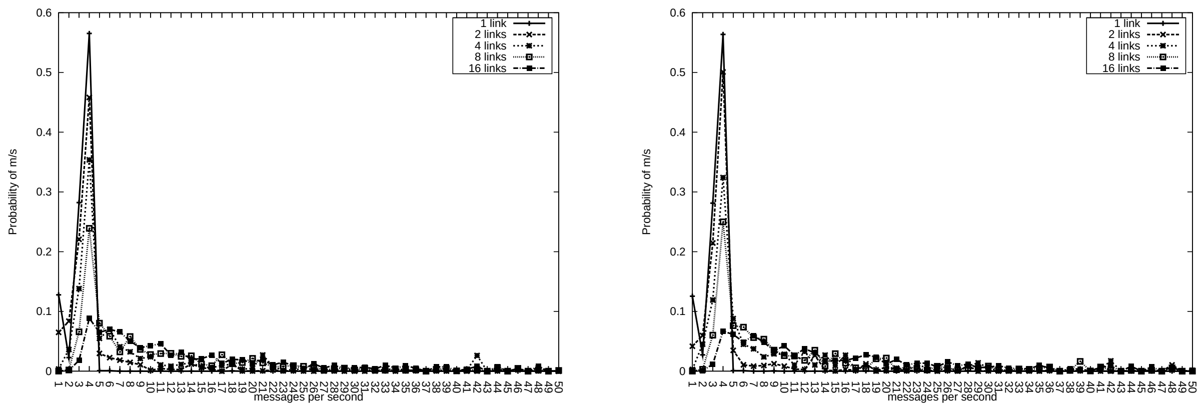
Figure 7.17: messages from controller for $B_v = 600kbit/s$

to them. This results in the need to assign quite frequently new request queues. For a high number of links (starting from 4 links here) the instantaneous combined download capacity $B(t)$ is so high that the local peers can keep-up with the chunk distribution and therefore the control is evoked at the chunk generation frequency, i. e., 3 times a second resulting in sending a new request queue to some peers 3 times a second.

The situation is different for higher video bitrates, e. g., for $B_v = 600kbit/s$, as shown in Figure 7.17. The peak number of messages per second is still proportional to the chunk generation frequency, but does not most of the messages are sent below that rate (again for more than 1 link).

7.4.2.2 *Messages sent to the Controller*

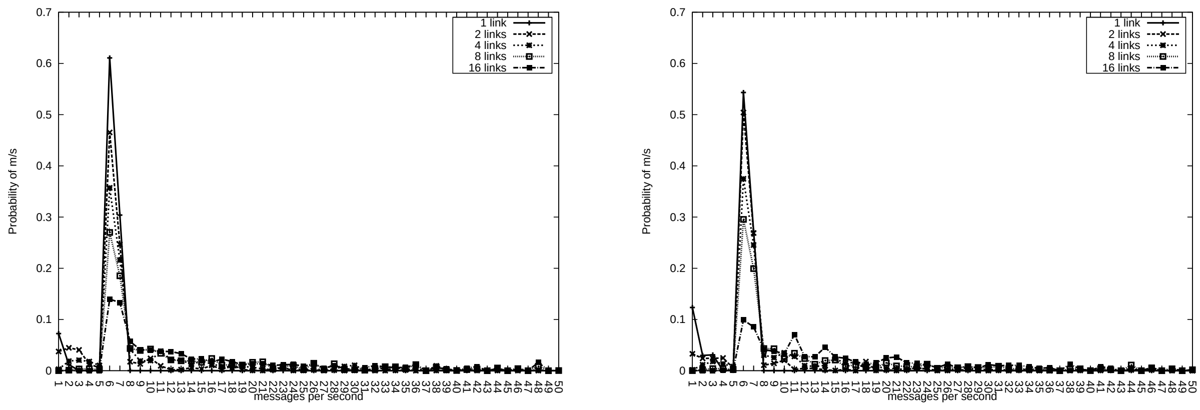
The number of messages sent from the peers to the controller is much higher compared to the number of messages sent from the controller to the peers. A controller message carries "aggregated" information for all peers, while the peers cannot aggregate messages across peers.



(a) precision 45 %

(b) precision 90 %

Figure 7.18: Messages to controller $B_v = 300 kbit/s$



(a) precision 45 %

(b) precision 90 %

Figure 7.19: Messages to controller $B_v = 600 kbit/s$

Each peer has a number of information to tell to the controller, e. g., successful download of a chunk, severe change of the link's situation, etc.

Figure 7.18 shows the probability of the messages per second sent by all peers to the controller, depending on the number of involved access-links, for $B_v = 300 kbit/s$ and precision values of 45 % and 90 %. Most of the messages are sent with a rate of below 20 messages per second,, but there is also a small fraction that is sent at rates above 20 messages per second. The burst in messages sent to the controller happens, if the achievable throughput situation at the peers changes, so that many peers have to report changes at the same time. The peak messages sent per second is also proportional to the chunk generation frequency, e. g.,, 3 chunks per second for 300 kbit/s, but the peak decreases with the number of access-links involved.

The probability of the messages per second sent by all peers to the controller for $B_v = 600 kbit/s$ exhibits the same pattern, but with the peak shifted to the chunk generation rate of 6 per second, as depicted in Figure 7.19.

The high number of messages per second to the controller is mainly due to the fact the behavior of the link's change quite frequently which may result in messages sent. The simulator is also not piggy packing messages sent from the peers to the controller, as compared to the piggy-packing of status reports described in Section 5.3.5.2. The peak is decreased for a larger set of access-links involved, as there are more links involved, which cause the need for more communication with the controller process, but at different times. This causes a spreading of events across more links.

7.4.2.3 Quantifying the Impact

The messages sent per second is not fully quantifying the resources needed on the sharing-link. The peers have to be able to handled the messages per second in terms of receiving the messages, filterting the relevant messages, and processing the filtered messages. However, for the local peers, the number of messages to be received is relative low, as they have only to listen to messages sent by the controller and messages indicating the successful retrieval of chunks. Whereas the controller must be able to receive all messages sent by all peers.

The messages sent by the controller contain information about the virtual chunk buffer and the request queue for each local peers. the virtual chunk buffer carries information for all chunks out of the trading window and for already retrieved chunks it also carries the information which peers have this chunk. This requires:

- current start position of the trading window (*sizeof(ChunkID)*)
- current end position of the trading window (*sizeof(ChunkID)*)
- list of chunks in trading window T_W as buffer map BM, where the single bits indicate the presences of a chunk. ($|BM| = \lceil \frac{|T_W|}{8} \rceil$)
- list of local peers in possession of chunk, listed per chunk. ($|Peers| \cdot (|PeerID| + |BM|)$)

Let's assume that ChunkIDs are of 4 bytes length (sizeof(ChunkID) = 4), the trading window is of 64 chunks length ($|T_W| = 64$, the number of peers is 16 (which is the maximum number of peers used in the simulation), and the PeerID is an IPv4 address (4 bytes length) and a TCP port number (2 bytes length). The buffer map has a size of 8 bytes. This results in $|Peers| \cdot (|PeerID| + |BM|) = 16 \cdot (6+8) = 224$ bytes for the peer's information about the buffer maps. Together with the trading window information, this results in 240 bytes.

Furthermore, the message sent by the controller carries the request queue assignments for all peers or a subset of peers. Assuming the worst case that all peers need to receive request queue assignments, no chunk has been retrieved yet, the peer's access-links are up and running, and that the throughput estimation precision all access-links is 30 % while all links are equal fast in terms of throughput. This requires that each chunk is assigned 3 times (with a threshold of 70 %) to different links. This requires to assign 12 chunk requests to each access-link according to $\frac{|T_W| \cdot 3}{|Peers|}$

This in turn will require a total of $|Peers| \cdot$ ChunkRequests $\cdot |ChunkID|$. This results in $16\ \text{peers} \cdot 12\ \text{Chunk Requests} \cdot 4\ \text{bytes} = 768\ \text{bytes}$ The total packet size is $240\ \text{bytes} + 768\ \text{bytes} = 1008\ \text{bytes}$, which fits well, including the IPv4 header (without options) of 20 bytes and the UDP header of 8 bytes, in a single UDP datagram over Ethernet with an MTU of 1500 bytes. The total used data is $240\ \text{bytes} + 768\ \text{bytes} + 20\ \text{bytes} + 8\ \text{bytes} = 1036\ \text{bytes}$.

The messages sent by the peers are either relative small messages but frequent, if they contain only status updates (a full request queue plus additional data – in the range of 100 bytes) or the messages carry a complete chunk (several kilobytes). The arrival rate of the chunks in the local system determines the cause the throughput on the sharing-link: Only a few chunks must be transmitted via the sharing-link, if only a few chunks arrive. However, a bursty arrival of chunks may also cause burst on the sharing-link.

Let's assume that all of our 16 links (see above) receive a chunk at the same point of time and they are distributing their particular chunk at the same time via the sharing-link. Given a chunk size of 12.5 kbyte and 16 links, and WLAN. with 54 Mbit/s, assuming that all chunks are transmitted back-to-back, we can saturate the link for 0.0296 s.

The sharing-link needs to provide high-speed network access so that it is able to carry the load of the information element redistribution and the signaling messages. The capacity of the sharing-link is not proportional to the number of participating nodes, but to the required throughput of the particular application.

8 Conclusions and Outlook

The concept and system design of the Cooperative Internet Access approach in resource constrained environments has been presented in this thesis. The fundamental properties of this information centric approach are the coordinated access of multiple independent nodes to resources in a network by leveraging system multihoming, application level striping to aggregate access-link capacities, and delegation of data retrieval. The approach is flow agnostic, as it does not rely on certain data flows and employed transport protocols, and it does only require support by participating local nodes but not by any further party in the network. A key aspect of the approach is the scheduling of information elements based on an estimated link behavior in the near future.

Section 8.1 summarizes the contributions of this thesis and Section 8.2 lists challenges and future work.

8.1 Summary of Contributions

This is a summary of the contribution of this thesis:

- We identified the resource constrained environment and challenged the status quo of today: Nodes in such an environment either cannot access the Internet as the users simply give up or there is the need to adapt the accessed content to the particular environment. But content adaptation requires action from entities not involved in the environment. We move beyond this with the Cooperative Internet Access approach [129] where multiple nodes access one or multiple resources in a joint and coordinated effort to minimize the number of information elements to be retrieved by the participating nodes.

- Active measurements of a resource constrained scenario to quantify the achievable throughput and the network availability in an exemplary setting of a train journey on a frequently used train connection in Germany. The measurements are the basis for the statistical data used to gain insights in the achievable throughput, uptime and downtime behavior for two German mobile network operators, where these statistics are used in the simulative system evaluation (see Chapter 3).

- The specification of the Cooperative Internet Access framework in Section 4.2 including: a discussion about how the the framework copes with resource environments; definition of the conceptual design; and a discussion about the design options for an implementation of the system.

- The specification of the Cooperative Internet Access for peer-to-peer video streaming in resource constrained environments [130], with
  - discussion of several chunk assignment strategies and the impact of access-link failures on these strategies (Section 5.2.1)
  - the deadline-based chunk scheduler based on a First-Fit-Decreasing (FFD) bin packing algorithm which uses a probability to determine how likely a chunk will be delivered within the deadline (Section 5.2.2).
  - a lightweight access-link throughput estimator to determine the throughput of a particular access-link within the next 1 s in the future. The throughput estimator has been verified against the measurements out of Chapter 3.
  - considerations of the chunk size useful in the measurement resource constrained environment, based on the measurements out of Chapter 3.
- A system design for the implementation of the peer-to-peer video streaming in Section 5.3. Including how to integrate the approach with state of the art peer-to-peer video streaming systems, whereas these systems is assumed to use chunks and is pull-based. The definition of data structures and flow charts to implement the system and also for the evaluation of the system in terms of required coordination effort caused by the Cooperative Internet Access.
- The specification of the Cooperative Internet Access for web browsing in resource constrained environments, with
  - how to use the framework to enable faster web browsing in resource constrained environments by delegating the retrieval of web page elements to participating nodes (Section 6.3) and by cooperatively using the cache-proxies of neighboring nodes.
  - a scheduling method for the web browsing in Section 6.3.4 which relies on heuristics per element type (e. g., picture, video, or text) to judge about the element size and the other node's minimal throughput estimation.
- A system design for the implementation of the cooperative web access in Section 6.4 by leveraging existing HTTP proxies and adding bazaar cache protocol and a controller design.
- A simulative study of the peer-to-peer video streaming in resource constrained environments, based on the measured statistics in Chapter 3 for the modelling of the access-links.
  - We simulated the chunk retrieval via the access-links with three different schedulers, i. e., round-robin, theoretical, and deadline-based schedulers and demonstrated the feasibility of the approach. The deadline based scheduler requires at least 7 peers to display a 300 kbit/s video without any interruption while a 600 kbit/s video requires 12 peers to participate, showing how resource constrained mobile Internet access is. The simulator allows also to change the precision of the access-link throughput estimation to judge the impact of a future throughput estimator with improved precision.

  - We examined the impact on the sharing link and showed that messages from the controller process to the peers is limited by the chunk generation rate f_C. The messages sent from the peers to the controller are subject to much higher rates as these messages carry status information about the behavior of the access-links, e. g., change in throughput, and also control information sent proportional to the chunk generation rate.

8.2 Challenges and Future Work

The Cooperative Internet Access system described in this thesis addresses the major challenges for applications in resource constrained environments. We list here open issues which require future work as they are not solved as part of this thesis. These issues are not of fundamental nature but they need further investigations beyond the scope of the thesis.

8.2.1 Security Aspects

8.2.1.1 Security of the Information Elements

The delegation of information element retrieval to other nodes raises questions about the security of these elements. In the general case, these nodes, and the users owning these nodes, are unknown and untrusted to any other participating node. Delegating information elements to these nodes may thus pose a security risk for the elements, as these nodes, can either suppress the delivery of the elements, slow down the delivery (an issue for real-time applications such as peer-to-peer live streaming), read the content, and modify the content.

The general approaches to secure the transport of information elements between 2 nodes across an untrusted network, e. g., the Internet, such as end-to-end encryption, usage of certificates or private/public key pairs, is not applicable to the Cooperative Internet Access approach. The main point of this approach is to rely on sharing of common information elements between nodes, requiring that these nodes are of mutual interest and thus also readable by any participating party.

This rules out the handling of private data via the Cooperative Internet Access approach, as also mentioned in Section 6.3.4 under the privacy aspects. It is also not possible to handle encrypted or authenticated traffic which relies on network or transport security via the system, as the encryption and authentication is usually bound to an IP address and some additional parameters, such as the Security Parameter Index (SPI) of IPsec [RFC 4301] or a TCP transport port number if TLS [RFC 5246] is used.

The security level applicable to the information elements in our context is the protection of the elements against modification. For some applications, e. g., peer-to-peer file-sharing, protection of the whole content against modification is also possible. For instance, bittorrent uses chunk level hashes, as well as, a hash across the whole file [80]. However, this is not applicable to peer-to-peer live streaming, as not the full content is needed for the playout (cf. FEC recovery) and the complete content is not available a priori. These systems may still use chunk level hashes [76], given a way to quickly and safely distribute the hashes before the actual data chunks arrive at the peers. Currently, peer-to-peer video streaming systems are per se subject to pollution attacks [131].

The security of information elements of web pages is also still unsolved, i.;e., there is no deployed solution at this point of time which is able to verify single web site elements.

8.2.1.2 Selfish Local Nodes

The approach in this thesis assumes that every participating local node has actually an access-link, it is providing access to this link to the local system, and it does not limit the capacity of the access-link in way that it is not useful anymore (e. g., limiting the achievable throughput to 5 kbit/s for no reason).

A node which does not provide an access-link or just a limited one will still be part of the local system and benefit from the other nodes' joint work to retrieve the information elements, without contributing anything to the local system.

The local system will need a future extension to detect such situation in a reliable manner and to exclude such freeriders from the system. However, the extension will need to take care to not exclude willing nodes with a temporal outage of the achievable throughput; or to let a group a malicious nodes to team up and to exclude other honest nodes, i. e., a denial of service in the local system.

8.2.2 Other Aspects of Fairness

The Section 8.2.1.2 above describes issues related to fairness amongst the participating nodes in the local system, with a focus on selfish nodes exploiting other node's access-link resources.

An obvious return for well behaving nodes is the better performance of the application in a restricted environment, as shown in Chapter 7 of this thesis. This can also be an incentive for nodes to participate in the local system, but it does not account for the monetary cost of the access-link to be paid by each user of each node. A future fairness mechanism could not only consider the numbers of bytes handled by each access-link, but also the connection costs of each access-link. Such a scheme would base the fairness mechanism on the product of number of bytes handled and occurring monetary costs for the bytes. This would ensure that users with expensive access-links (e. .g, mobile wireless which is paid per use) will need to handle less bytes, as compared to a user with a flat-rate, but they still would carry the same cost of product of number of bytes handled and occurring monetary costs.

Another interesting point is if participating nodes can built-up credit when they devote their access-link to the local system at one time, but pay off with the credit at other occasions if they cannot devote their access-link to the system. This would also allow to let other users to buy credits with real money and to pay with these credits their participation in the system without the need to have an access-link.

8.2.3 Practicability of Approach

The Cooperative Internet Access requires at least 2 users (and their nodes) to be in physical proximity to participate to let the system work. The results out of the simulation in Section 7.4 show that for peer-to-peer live streaming a higher number of 7 to 12 users is required for a proper video display. Finding 7 or more users in the same train with interest to see the same content may or may not be possible, depending on the content (block busters vs. long tail) and the location. For instance, finding 7 users in a rural area is unlikely, but finding them in a commuter train can be possible.

For other applications, such as the cooperative web access, finding a few willing users might be not as hard, as even 2 nodes will already deliver a better service to both users, as compared to let everybody on its own. This can improve web access to users in rural areas with very limited access networks in terms of achievable throughput.

This issue of finding a sufficient number of users is an orthogonal issue to the technical challenges dealt with in this thesis, but non-negligible for a real deployment. Nonetheless, it is future work to determine how many users per application case, e. g., peer-to-peer video streaming or web browsing, and in what setting, e. g., static in rural area or moving in a train, are needed to let the system operate at an operating point useful to the users.

8.2.4 Impact of Media Codecs

The recent advances in video coding techniques, such as SVC [132] and MDC [133], fit perfectly to the approach as adaptive codecs will allow to adjust the video rate to the current throughput situation. It has to be seen how such adaptive codecs are integrated in the peer-to-peer live streaming systems and how this can be leveraged by the Cooperative Internet Access approach.

There is early work on using MDC for peer-to-peer video streaming [134], but to the best knowledge of the author no existing peer-to-peer system in deployment is using this.

8.2.5 Implementation Issues

This thesis elaborates on the general system design, two specific application using the Cooperative Internet Access approach, and a simulative evaluation based on measurements. But there is no implementation of the system yet which can be taken for field trials under live conditions. Such an implementation is future work and will require more considerations, but will also allow to test the system under very different conditions, e. g., in metro areas with higher mobile Internet penetration, but also in deployments where only ISDN Internet access is available.

Such an implementation will also allow to gain more insights on how the system reacts to

node churn which will impact the controller process, but also the system's instantaneous download capacity.

fairness in peer-to-peer systems It has to be investigated how the proposed peer-to-peer video streaming system interacts with algorithms in peer-to-peer protocols that ensure fairness among the nodes – so far the simulations assume that peers in the fixed part of the network are willing to give an arbitrary amount of chunks without an adequate service in return.

impact of the sharing link The behavior of the sharing-link will impact the overall system impact and it is to be explored how existing sharing-link technologies, such as WLAN, will impact the system.

number of users The system will have an upper limit of users up to where the systems makes sense. This has to be explored in a live deployment, as not only the purse system design plays a role, but also how the implementation and the sharing-link environment play together.

link behavior Section 4.2.5 elaborates on the importance to determine how the achievable throughput of 2 links correlates in order to determine the behavior in terms of joint, disjoint, or partially joint. An implementation of the system will need to implement such a mechanism based on signal analysis (cf. [96])

A Curriculum Vitae

Personal Data

Name:	Martin Stiemerling
Date of Birth:	24. May 1976
Place of Birth:	Cologne, Germany
Nationality:	German
Address:	Karl-Gehrig-Weg 15 69181 Leimen Germany

Research Interests

- peer-to-peer networking for video distribution and voice
- computer networks in challenged envrionments
- overlay networks
- traffic localization for peer-to-peer and content distribution networks
- Internet signalling

Scientific Education

May 2011 (expected)	Dr. rer.nat. in Computer Science
July 2007 to Februar 2011	PhD studies, University of Göttingen, Computer Science, Prof. Dr. Xiaoming Fu, Computer Networks Group
August 2000	Diplom Ingenieur (FH) Elektrotechnik (Diploma in electrical engineering)
March 2000 to August 2000	Diploma thesis "Video transmission in the Internet for educational purposes" joint work with Rolf Meßmer
Sept. 1996 to August 2000	Fachhochschule Köln, Nachrichtentechnik (University of Applied Science Collogne, communications engineering)

Work Experience

April 2007 (ongoing)	Senior Researcher, NEC Laboratories Europe, Heidelberg Research and development in the areas of peer-to-peer video distribution, traffic localization for peer-to-peer, peer-to-peer Voice over IP, network virtualization
October 2000 to March 2007	Researcher, NEC Laboratories Europe, Heidelberg Research and development in the areas of Mobile IPv6, Internet architecture, Internet signalling, Voice over IP, network management, overlay networks, peer-to-peer networks

Publications

Publications in the course of the PhD studies:

1. M. Stiemerling and Marcus Brunner, ”A Peer-to-Peer SIP System based on Service-Aware Transport Overlays”, Praxis der Informationsverarbeitung und Kommunikation (PIK), Special Issue on Voice over IP, Volume 30, No. 4, December 2007.

2. M. Stiemerling, X. Fu, and M. Brunner, ”A Network Virtualisation Concept Based on Ambient Networks SATO System”, 1st GI/ITG Fachgespraech Virtualisierung, Paderborn, Germany, February 2008.

3. X. Fu, M. Stiemerling, and H. Schulzrinne, ”Implications and Control of Middleboxes in the Internet”, IEEE Network, Special Issue on Implications and Control of Middleboxes in the Internet, September 2008.

4. J. Seedorf, F. Ruwolt, M. Stiemerling, and S. Niccolini, ”Evaluating P2PSIP under Attack: An Emulative Study”, IEEE Globecom 2008, New Orleans, LA, USA, December 2008.

5. M. Stiemerling, M. Brunner, S. Kiesel, and X. Fu, ”TORI: User Provided Future Networking Testbeds, IEEE International Workshop on the Network of the Future, in conjunction with IEEE ICC 2009, Dresden, Germany, June 2009.

6. J. Seedorf, S. Kiesel, and M. Stiemerling, ”Traffic Localization for P2P-Applications: The ALTO Approach”, Ninth International Conference on Peer-to-Peer Computing (IEEE P2P 2009), IEEE, Seattle, WA, USA,September 2009.

7. M. Stiemerling and S. Kiesel, ”A System for Peer-to-Peer Video Streaming in Resource Constrained Mobile Environments”, User Provided Networking (U-NET) workshop, in conjuncation with the ACM CoNext 2009 conference, Rome, Italy, December 2009.

8. M. Stiemerling and S. Kiesel, ”Cooperative P2P Video Streaming for Mobile Peers”, 19th IEEE International Conference on Computer Communications and Networks (ICCCN 2010), Track on Multimedia and Peer-to-Peer Networking (MP2P), Zurich, Switzerland, August 2010.

9. J. Seedorf, S. Niccolini, M. Stiemerling, E. Ferranti, and R. Winter, ”Quantifying Operational Cost-Savings through ALTO-Guidance for P2P Live Streaming”, Proceedings of Incentives, Overlays, and Economic Traffic Control Third International Workshop, ETM 2010, Amsterdam, The Netherlands, September 2010.

Publications before the start of the PhD studies:

1. M. Stiemerling, and J. Quittek, ”Middlebox Configuration Protocol Design”, IEEE IP Operations and Management (IPOM) 2002 Workshop, Dallas, USA, October 2002.
2. J. Silva Tobella, M. Stiemerling, and M. Brunner, ”Towards Self-Configuration of IPv6 Routers”, In Proc. of IEEE/IFIP Network Operations & Management Symposium (NOMS 2004), Seoul, Korea, April 2004.
3. M. Stiemerling, C. Cadar, A. Fessi, and M. Brunner, ”IP Multimedia Subsystem Interworking with Firewalls and NATs”, World Telecom Congress (WTC’04), Seoul, Korea, September 2004.
4. M. Blasi Folch, M. Stiemerling, and M. Brunner, ”SIP Policy Control using Modular Firewalls”, IEEE IP Operations and Management (IPOM), Beijing, China, October 2004.
5. C. Aoun, M. Stiemerling, E. Davies, and H. Tschofenig , ”Path-Directed Signaling Usage in the Internet”, IEEE IP Operations and Management (IPOM), Beijing, China, October 2004.
6. M. Stiemerling, J. Quittek, and L. Eggert, ”Middlebox Traversal of HIP Communication”, Workshop on HIP and Related Architectures, Washington, DC, USA, November 2004.
7. L. Eggert, J. Laganier, M. Liebsch and M. Stiemerling , ”HIP Resolution and Rendezvous Mechanisms”, Workshop on HIP and Related Architectures, Washington, DC, USA, November 2004.
8. M. Martin, M. Brunner, M. Stiemerling, and A. Fessi, ”Path-coupled signaling for NAT/Firewall traversal”, IEEE High Performance Switching and Routing (HPSR05), Hong Kong, May 2005.
9. M. Brunner, S. Schuetz, J. J. Silv Tobella, and M. Stiemerling, ”Plug and Play Configuration for Composable Networks”, In Proc. of 9th IFIP/IEEE Symposium on Integrated Management (IM 2005), Nice, France, May 2005.
10. S. Niccolini and M. Stiemerling, ”Toward ubiquitous IP Telephony deployments: state of the art, challenges and recommendations”, TNC 2005, Poznan, Polen, June 2005.
11. S. Niccolini, M. Stiemerling, D. Luzzi, and M. Brunner, ”Identification, Blocking, and Charging of Peer-to-Peer (P2P) Real-Time Applications”, World Telecommunication Congress 2006 (WTC06), Budapest, Hungary, April/May 2006.
12. M. Kampmann, F. Hartung, M. Stiemerling, B. Mathieu, and M. Song, ”Mobile Media Delivery using Overlay Networks”, EuMob 2006 Workshop, Alghero, Italy, September 2006.

13. B. Mathieu, M. Song, A. Galis, L. Cheng, K. Jean, R. Ocampo , M. Brunner, M. Stiemerling, and M. Cassini, ”Self-Management of Context-Aware Overlay Networks for Ambient Networks”, In Proc. of 10th IFIP/IEEE Symposium on Integrated Management (IM 2007), Munich, May 2007.

14. J. Quittek, S. Niccolini, S. Tartarelli, M. Stiemerling, M. Brunner, and T. Ewald, ”Detecting SPIT Calls by Checking Human Communication Patterns”, IEEE International Conference on Communications 2007 (ICC 2007), Glasgow, UK, June 2007.

Editorial Service

- Guest Editor, IEEE Network, Special Issue on Implications and Control of Middleboxes in the Internet, September/October 2008.

Conference Activities (2009 and 2010)

- TPC co-chair of Workshop on Network Virtualization, In conjunction with the 17th GI/ITG Conference on Kommunikation in Verteilten Systemen (KiVS 2011), Kiel, Germany, March 8-11, 2011.
- TPC member of IEEE International Conference on Computer Communications and Networks (ICCCN 2010) – Track on Internet Services, Architectures and Protocols (ISAP)
- TPC member of Globecom 2010 Next Generation Networks (NGN) Symposium
- TPC member of VISA 2010: The Second ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures
- TPC member of VISA 2009: The First ACM SIGCOMM Workshop on Virtualized Infastructure Systems and Architectures
- TPC co-chair of Workshop on Overlay and Network Virtualization, March 6, 2009, In conjunction with the 16th GI/ITG Conference on Kommunikation in Verteilten Systemen (KiVS 2009), Kassel, Germany, March 2-6, 2009.
- TPC member of IEEE International Conference on Computer Communications and Networks (ICCCN 2009) – Track on Network Architecture and Protocols (NAP)
- TPC member of 20th ITC Specialist Seminar on Network Virtualization

Bibliography

[1] Martin Sauter. *Grundkurs Mobile Kommunikationssysteme: Von UMTS, GSM und GPRS zu Wireless LAN und Bluetooth Piconetzen*. Vieweg Verlag, 3. auflage edition, 2008.

[2] European Commission. Shaping Europe's digitial revolution [online]. URI: http://ec.europa.eu/dgs/information_society/see_more/vb/index_en.htm [Retrieved on November 29, 2010].

[3] YouTube. youtube web site [online]. URI: http://wwww.youtube.com [Retrieved on November 29, 2010].

[4] M. Zink, K. Suh, Y. Gu, and J. Kurose. Watch global, cache local: YouTube network traffic at a campus network-measurements and implications. *Proceeding of the 15th SPIE/ACM Multimedia Computing and Networking (MMCN'08)*, 2008.

[5] E. Alessandria, M. Gallo, E. Leonardi, M. Mellia, and M. Meo. P2P-TV systems under adverse network conditions: a measurement study. In *Infocom 2009. The 28th Conference on Computer Communications. IEEE*. IEEE, April 2009.

[6] Timothy C. Kwok. Residential Broadband Internet Services and Applications Requirements. *IEEE Communications Magazine*, 1997.

[7] Wikipedia. DSL (Telekom) [online]. URI: http://de.wikipedia.org/wiki/DSL_%28Telekom%29 [Retrieved on November 29, 2010].

[8] Wikipedia. Global System for Mobile Communications [online]. URI: http://de.wikipedia.org/wiki/GSM [Retrieved on November 29, 2010].

[9] Wikipedia. General Packet Radio Service [online]. URI: http://de.wikipedia.org/wiki/GPRS [Retrieved on November 29, 2010].

[10] Wikipedia. Enhanced Data Rates for GSM Evolution [online]. URI: http://de.wikipedia.org/wiki/Enhanced_Data_Rates_for_GSM_Evolution [Retrieved on November 29, 2010].

[11] Wikipedia. UMTS Release 99 [online]. URI: http://de.wikipedia.org/wiki/Universal_Mobile_Telecommunications_System [Retrieved on November 29, 2010].

[12] Wikipedia. High Speed Downlink Packet Access [online]. URI: http://de.wikipedia.org/wiki/HSDPA [Retrieved on November 29, 2010].

[13] Bram Cohen. The BitTorrent Protocol Specification [online]. URI: http://www.bittorrent.org/beps/bep_0003.html [Retrieved on November 29, 2010].

[14] PPLive. PPLive Web Site [online]. URI: http://www.pplive.com [Retrieved on November 29, 2010].

[15] IETF. Multiple Interfaces (mif) Working Group [online]. URI: http://datatracker.ietf.org/wg/mif/charter/ [Retrieved on November 29, 2010].

[16] Jun Yao, Yi Duan, and Jianyu Pan. Implementation of a Multihoming Agent for Mobile On-board Communication. In *Proceedings of VTC*, 2006.

[17] C. Brendan S. Traw and Jonathan M. Smith. Striping within the network subsystem. *IEEE Network*, pages 22–32, August 1995.

[18] A. Habib, N. Christin, and J. Chuang. Taking advantage of multihoming with session layer striping. In *Proceedings of IEEE Global Internet 2006*, pages 102–107, April 2006.

[19] IEEE. IEEE Standard for Local and Metropolitan Area Networks – Link Aggregation [online]. URI: http://standards.ieee.org/getieee802/download/802.1AX-2008.pdf [Retrieved on November 29, 2010].

[20] Dhananjay S. Phatak and Tom Goff. A Novel Mechanism for Data Streaming Across Multiple IP Links for Improving Throughput and Reliability in Mobile Environments. In *Proceedings of INFOCOM*, June 2002.

[21] H. Han S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley. Multi-path TCP: A joint congestion control and routing scheme to exploit path diversity on the internet. *IEEE/ACM Transactions on Networking*, 14:1260–1271, 2006.

[22] Hung Y. Hsieh and Raghupathy Sivakumar. pTCP: An End-to-End Transport Layer Protocol for Striped Connections. In *IEEE International Conference on Network Protocols*. IEEE Computer Society, November 2002.

[23] H. Sivakumar, S. Bailey, and R. L. Grossman. PSockets: the case for application-level network striping for data intensive applications using high speed wide area networks. In *Supercomputing '00: Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM)*, page 37, Washington, DC, USA, 2000. IEEE Computer Society.

[24] Kevin Fall. A delay-tolerant network architecture for challenged internets. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34, New York, NY, USA, 2003. ACM.

[25] A.L. Iacono and C. Rose. Mine, mine, mine: information theory, infostation networks, and resource sharing. In *Wireless Communications and Networking Conference, 2000. WCNC. 2000 IEEE*, volume 3, pages 1541–1546. IEEE, 2000.

[26] Jörg Ott and Dirk Kutscher. A Disconnection-Tolerant Transport for Drive-thru Internet Environments. In *INFOCOM 2005. 24th IEEE International Conference on Computer Communications. Proceedings*, pages 1849–1862, 2005.

[27] H. Hartenstein, B. Bochow, A. Ebner, M. Lott, M. Radimirsch, and D. Vollmer. Position-aware ad hoc wireless networks for inter-vehicle communications: the Fleetnet project. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, page 262. ACM, October 2001.

[28] Fei Xie, Kien A. Hua, Wenjing Wang, and Yao Hua Ho. Performance study of live video streaming over highway vehicular ad hoc networks. In *VTC Fall*, pages 2121–2125, September 2007.

[29] Mohsen Sardari, Faramarz Hendessi, and Faramarz Fekri. DMRC: dissemination of multimedia in vehicular networks using rateless codes. In *INFOCOM'09: Proceedings of the 28th IEEE international conference on Computer Communications Workshops*, pages 182–187, April 2009.

[30] Pablo Rodriguez, Ian Pratt, Julian Chesterfield, Rajiv Chakravorty, and Suman Banjeree. MAR: A Commuter Router Infrastructure for the Mobile Internet. In *Proceedings of ACM Mobisys*, pages 217–230, June 2004.

[31] Alix Chow, Hao Yang, Cathy Xia, Minkyong Kim, Zhen Liu, and Hui Lei. EMS: Encoded Multipath Streaming for Real-time Live Streaming Applications. In *Proceedings of The 17th IEEE International Conference on Network Protocols (ICNP'09)*. IEEE, October 2009.

[32] Nathanael Thompson, Guanghui He, and Haiyun Luo. Flow Scheduling for End-host Multihoming. In *In Proceedings of IEEE INFOCOM*, pages 1–12, April 2006.

[33] Enhua Tan, Lei Guo, Songqing Chen, , and Xiaodong Zhang. CUBS: Coordinated Upload Bandwidth Sharing in Residential Networks. In *Proceedings of The 17th IEEE International Conference on Network Protocols (ICNP'09)*, pages 193–2002. IEEE, October 2009.

[34] Casey Carter and Robin Kravets. User Devices Cooperating to Support Resource Aggregation. In *in Proceedings of IEEE WMSCA*, pages 59–69, June 2002.

[35] X. Zhu and B. Girod. Video streaming over wireless networks. In *Proceedings of European Signal Processing Conference,(Poznan, Poland, September 3-7, 2007). EUSIPCO07*, pages 1462–1466, September 2007.

[36] Bittorrent Inc. Bittorrent page [online]. URI: `http://bittorrent.org/` [Retrieved on November 29, 2010].

[37] Robin Perkins. Zeroconf Peer Advertising and Discovery [online]. URI: `http://www.bittorrent.org/beps/bep_0026.html` [Retrieved on November 29, 2010].

[38] Hayoung Yoon, JongWon Kim, Feiselia Tan, and Robert Hsieh. On-demand Video Streaming in Mobile Opportunistic Networks. In *Proceedings of PERCOM '08*, pages 80–89, Washington, DC, USA, 2008. IEEE Computer Society. `doi:http://dx.doi.org/10.1109/PERCOM.2008.18`.

[39] Tai T. Do, Kien A. Hua1, Ning Jiang, and Fuyu Liu. PatchPeer: A scalable video-on-demand streaming system in hybrid wireless mobile peer-to-peer networks. In *Peer-to-Peer Networking and Applications Journal*. Springer, February 2009.

[40] P. Garbacki, A. Iosup, D. Epema, and M. van Steen. 2fast: Collaborative downloads in p2p networks. In *Sixth IEEE International Conference on Peer-to-Peer Computing, 2006. P2P 2006*, pages 23–30, September 2006.

[41] Michalis Faloutsos Aggelos Vlavianos, Marios Iliofotou. BiToS: Enhancing BitTorrent for supporting streaming applications. In *IEEE Global Internet Symposium*, pages 1–6, April 2006.

[42] A. Wolman, M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H.M. Levy. On the scale and performance of cooperative web proxy caching. In *Proceedings of the seventeenth ACM symposium on Operating systems principles*, page 31. ACM, 1999.

[43] Jia Wang. A survey of web caching schemes for the internet. *ACM SIGCOMM Computer Communication Review*, 29(5):46, 1999.

[44] XiaoYu Wang, WeeSiong Ng, BengChin Ooi, Kian-Lee Tan, and AoYing Zhou. BuddyWeb A P2P-Based Collaborative Web caching System. In *Web Engineering and Peer-to-Peer Computing, NETWORKING 2002 Workshops*, pages 247–251, May 2002.

[45] Bo Ling, Xiao-Yu Wang, Ao-Ying Zhou, and Wee-Siong Ng. Collaborative web caching system based on peer-to-peer architecture. In *Jisuanji Xuebao (Chinese Journal Computing)*, pages 170–178, February 2005.

[46] Sitaram Iyer, Antony Rowstron, and Peter Druschel. Squirrel: A decentralized peer-to-peer web cache. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, page 222. ACM, 2002.

[47] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware 2001: IFIP/ACM International Conference on Distributed Systems Platforms*, page 329. Springer, November 2001.

[48] Salman Baset and Henning Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. In *in Proceedings 25th IEEE International Conference on Computer Communications (INFOCOM 2006)*. IEEE, April 2006.

[49] Andy Oram. *Harnessing the Power of Disruptive Technologies*. O'Reilly Media, Sebastopol, CA, USA, 2001.

[50] Siyu Tang, Yue Lu, Javier Martín Hernández, Fernando Kuipers, and Piet Mieghem. Topology Dynamics in a P2PTV Network. In *NETWORKING '09: Proceedings of the 8th International IFIP-TC 6 Networking Conference*, pages 326–337, Berlin, Heidelberg, May 2009. Springer-Verlag.

[51] Vijay Gopalakrishnan, Samrat Bhattacharjee, K. K. Ramakrishnan, Rittwik Jana, and Divesh Srivastava. CPM: Adaptive Video-on-Demand with Cooperative Peer Assists and Multicast. In *in Proceedings of INFOCOM 2009*, pages 91–99, April 2009.

[52] wiki. Napster wiki page [online]. URI: http://en.wikipedia.org/wiki/Napster [Retrieved on November 29, 2010].

[53] wiki. Gnutella wiki page [online]. URI: http://en.wikipedia.org/wiki/Gnutella [Retrieved on November 29, 2010].

[54] wiki. eDonkey wiki page [online]. URI: http://en.wikipedia.org/wiki/EDonkey_network [Retrieved on November 29, 2010].

[55] Anthony and Son T. Yuand Vuong. MOPAR: a mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games. In *NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*, pages 99–104, New York, NY, USA, June 2005. ACM.

[56] B. Knutsson, H. Lu, W. Xu, and B. Hopkins. Peer-to-peer support for massively multiplayer games. In *Proceedings of IEEE INFOCOM*, pages 96–107, March 2004.

[57] S. Rieche, K. Wehrle, M. Fouquet, H. Niedermayer, L. Petrak, and G. Carle. Peer-to-peer-based infrastructure support for massively multiplayer online games. In *Proceedings of the 4th IEEE Consumer Communications and Networking Conference, CCNC*, pages 763–767, January 2007.

[58] Wu-chang Feng, Francis Chang, Wu-chi Feng, and Jonathan Walpole. A traffic characterization of popular on-line games. *IEEE/ACM Trans. Netw.*, 13(3):488–500, 2005.

[59] wiki. Wiki MMORPG [online]. URI: http://en.wikipedia.org/wiki/Massively_multiplayer_online_role-playing_game [Retrieved on November 29, 2010].

[60] Zattoo Europa AG. Zattoo home page [online]. URI: http://www.zattoo.com [Retrieved on November 29, 2010].

[61] Ravy Inc. Ravy home page [online]. URI: http://www.rayv.com [Retrieved on November 29, 2010].

[62] S.G. Rao Chu Yang-hua and S. Seshanand Zhang Hui. A case for end system multicast. *IEEE Journal on Selected Areas in Communications*, 20(8):1456 – 1471, 2002.

[63] Dimitrios Pendarakis, Sherlia Shi, Dinesh Verma, and Marcel Waldvogel. ALMI: An Application Level Multicast Infrastructure. In *USENIX Symposium on Internet Technologies and Systems*, March 2001.

[64] B. Zhang, S. Jamin, and L. Zhang. Host multicast: A framework for delivering multicast to end users. In *IEEE INFOCOM*, volume 3, pages 1366–1375. Citeseer, 2002.

[65] M. Castro, P. Druschel, A.M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: high-bandwidth multicast in cooperative environments. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, page 313. ACM, 2003.

[66] Xinyan Zhang, Jiangchuan Liu, and Tak shing Peter Yum. Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming. In *INFOCOM 2005. 24th IEEE International Conference on Computer Communications. Proceedings*, March 2005.

[67] F. Pianese, J. Keller, and E.W. Biersack. PULSE, a Flexible P2P Live Streaming System. In *in Proceedings 25th IEEE International Conference on Computer Communications (INFOCOM 2006)*. IEEE, April 2006.

[68] María Elisa Bertinat, Daniel De Vera, Darío Padula, Franco Robledo, Pablo Rodríguez-Bocca, Pablo Romero, and Gerardo Rubino. GoalBit: The First Free and Open Source Peer-to-Peer Streaming Network. In *LANC '09: Proceedings of the 5th international IFIP/ACM Latin American conference on Networking*, New York, USA, September 2009. ACM.

[69] Li Zhao, Jian-Guang Luo, Meng Zhang, Wen-Jie Fu, Ji Luo, Yi-Fei Zhang, and Shi-Qiang Yang. Gridmedia: A Practical Peer-to-Peer Based Live Video Streaming System. In *7th Workshop on Multimedia Signal Processing (IEEE MMSP)*, pages 1–4, November 2006.

[70] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *ICS '02: Proceedings of the 16th international conference on Supercomputing*, pages 84–95, New York, NY, USA, June 2002. ACM.

[71] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making gnutella-like p2p systems scalable. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, page 418. ACM, August 2003.

[72] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on networking*, 11(1):17–32, 2003.

[73] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. *Peer-to-Peer Systems*, pages 53–65, 2002.

[74] Azureus Project. Azureues wiki page [online]. URI: `http://azureus.sourceforge.net/` [Retrieved on November 29, 2010].

[75] Nazanin Magharei and Reza Rejaie. Mesh or multiple-tree: A comparative study of live p2p streaming approaches. In *in Proceedings of IEEE INFOCOM*, pages 1424–1432, 2007.

[76] Fabio Pianese. *PULSE - an adaptive practical live streaming system*. PhD thesis, Institute EURECOM, Sophia Antipolis, France, December 2007.

[77] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and Keith W. Ross. A Measurement Study of a Large-Scale P2P IPTV System. *IEEE Transactions on Multimedia*, December 2007.

[78] Ayalvadi J. Ganesh, Anne marie Kermarrec, and Laurent MassouliÃ©. SCAMP: Peer-to-peer lightweight membership service for large-scale group communication. In *NGC '01: Proceedings of the Third International COST264 Workshop on Networked Group Communication*, pages 44–55. Springer-Verlag, November 2001.

[79] L. Rizzo. Effective erasure codes for reliable computer communication protocols. 27(2):24–36, April 1997.

[80] Brian Cohen. Incentives build robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6. Citeseer, 2003.

[81] Miikka Lundan and Igor Curcio. Performance evaluation of the mobile peer-to-peer service. In *In Proceedings 12th International Conference on Computer Communications and Networks (ICCCN 2003)*, pages 101–106, October 2003.

[82] Igor Curcio. Mobile Streaming Services in WCDMA Networks. In *ISCC '05: Proceedings of the 10th IEEE Symposium on Computers and Communications*, pages 231–236, Washington, DC, USA, 2005. IEEE Computer Society.

[83] Almudena Diaz, Pedro Merino, Laura Panizo, and Alvaro M. Recio. Experimental analysis of peer-to-peer streaming in cellular networks. *Advanced Information Networking and Applications, International Conference on*, 0:784–791, 2007.

[84] Mikael Wiberg. FolkMusic: A Mobile Peer-to-Peer Entertainment System. In *HICSS '04: Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 9*, page 90290.2, Washington, DC, USA, 2004. IEEE Computer Society.

[85] Seung-Seok Kang and Matt Mutka. Efficient mobile access to internet data via a wireless peer-to-peer network. In *PERCOM '04: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04)*, page 197, Washington, DC, USA, March 2004. IEEE Computer Society.

[86] Ingo Gruber, Rudiger Schollmeier, and Wolfgang Kellerer. Performance evaluation of the mobile peer-to-peer service. In *CCGRID '04: Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid*, pages 363–371, Washington, DC, USA, 2004. IEEE Computer Society.

[87] Jani Peltotalo, Jarmo Harju, Marko Saukko, Lassi Väätämöinen, Imde Bouazizi, Igor Curcio, and Joep van Gassel. A real-time peer-to-peer streaming system for mobile networking environment. In *INFOCOM'09: Proceedings of the 28th IEEE international conference on Computer Communications Workshops*, pages 163–169, Piscataway, NJ, USA, 2009. IEEE Press.

[88] Jeonghun Noh, Mina Makar, and Bernd Girod. Streaming to mobile users in a peer-to-peer network. In *Mobimedia '09: Proceedings of the 5th International ICST Mobile Multimedia Communications Conference*, pages 1–7, ICST, Brussels, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[89] Jyrik Akkanen, Olli Karonen, and Jyrki Porio. Mobile Streaming Services in WCDMA Networks. In *CCNC 2008: Consumer Communications and Networking Conference*, pages 1253–1254, January 2008.

[90] Jörg Eberspächer, Hans-Joerg Vögel, Christian Bettstetter, and Christian Hartmann. *GSM - Architecture, Protocols and Services*. Wiley, 3rd edition, 2008.

[91] wiki. ICE 3 train wiki page [online]. URI: http://de.wikipedia.org/wiki/ICE_3 [Retrieved on November 29, 2010].

[92] Erik Dahlman, Hannes Ekstrom, Anders Furuskar, Ylva Jading, Jonas Karlsson, Magnus Lundevall, and Stefan Parkvall. The 3G Long-term Evolution- Radio Interface Concepts and Performance Evolution. In *2006 IEEE 63 rd Vehicular Technology Conference*, pages 7–10, 2006.

[93] Wee Lum Tan, Fung Lam, and Wing Cheong Lau. An Empirical Study on the Capacity and Performance of 3G Networks. *IEEE Trans. Mob. Comput.*, 7(6):737–750, 2008.

[94] Huaizhong Han, Srinivas Shakkottai, C. V. Hollot, R. Srikant, and Don Towsley. Multi-path tcp: a joint congestion control and routing scheme to exploit path diversity in the internet. *IEEE/ACM Trans. Netw.*, 14(6):1260–1271, 2006.

[95] Hung-Yun Hsieh and Raghupathy Sivakumar. pTCP: An End-to-End Transport Layer Protocol for Striped Connections. In *ICNP '02: Proceedings of the 10th IEEE International Conference on Network Protocols*, pages 24–33, Washington, DC, USA, 2002. IEEE Computer Society.

[96] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck. *Discrete-time signal processing (2nd ed.)*. Prentice-Hall, Inc., 1999.

[97] OpenSSH. OpenSSH Web Site [online]. URI: http://www.openssh.com [Retrieved on November 29, 2010].

[98] wiki. telnet wiki page [online]. URI: http://en.wikipedia.org/wiki/Telnet [Retrieved on November 29, 2010].

[99] Arbor Networks. IETF web page [online]. URI: http://www.ietf.org/proceedings/77/slides/plenaryt-4.pdf [Retrieved on November 29, 2010].

[100] Skype Limited. Skype web site [online]. URI: http://wwww.skype.com [Retrieved on November 29, 2010].

[101] Peter Merz, Matthias Priebe, and Steffen Wolf. Super-Peer Selection in Peer-to-Peer Networks Using Network Coordinates. In *ICIW '08: Proceedings of the 2008 Third International Conference on Internet and Web Applications and Services*, pages 385–390, Washington, DC, USA, 2008. IEEE Computer Society.

[102] wiki. MTU wiki page [online]. URI: http://de.wikipedia.org/wiki/Maximum_Transmission_Unit [Retrieved on November 29, 2010].

[103] Thomas Bonald, Laurent Massoulié, Fabien Mathieu, Diego Perino, and Andrew Twigg. Epidemic live streaming: optimal performance trade-offs. In *SIGMETRICS*, pages 325–336, 2008.

[104] Meng Zhang, Jian-Guang Luo, Li Zhao, and Shi-Qiang Yang. A peer-to-peer network for live media streaming using a push-pull approach. In *Proc. of ACM Multimedia*, November 2005.

[105] T. Nguyen and A. Zakhor. Distributed video streaming with forward error correction. In *Packet Video Workshop*, volume 2002, 2002.

[106] Jun Yao, Salil S. Kanhere, and Mahbub Hassan. An empirical study of bandwidth predictability in mobile computing. In *WINTECH*, pages 11–18, 2008.

[107] Bram Cohen. Incentives build robustness in bittorrent, 2003.

[108] S. Liu, M. Chen, S. Sengupta, M. Chiang, J. Li, and P.A. Chou. P2P streaming capacity under node degree bound. In *2010 International Conference on Distributed Computing Systems*, pages 587–598. IEEE, 2010.

[109] World Wide Web Consortium. HTML 4.01 Specification [online]. URI: http://www.w3.org/TR/html4/ [Retrieved on November 29, 2010].

[110] wiki. HTTP proxy wiki page [online]. URI: http://en.wikipedia.org/wiki/Proxy_server [Retrieved on November 29, 2010].

[111] wiki. Web 2.0 wiki page [online]. URI: http://en.wikipedia.org/wiki/Web_2.0 [Retrieved on November 29, 2010].

[112] wiki. Ajax wiki page [online]. URI: http://en.wikipedia.org/wiki/Ajax_(programming) [Retrieved on November 29, 2010].

[113] Mark Nottingham. Caching Tutorial for Web Authors and Webmasters [online]. URI: http://www.mnot.net/cache_docs/#TIPS [Retrieved on November 29, 2010].

[114] websiteoptimization.com. Average Web Page Size [online]. URI: http://www.websiteoptimization.com/speed/tweak/average-web-page/ [Retrieved on November 29, 2010].

[115] websiteoptimization.com. The Psychology of Web Performance [online]. URI: http://www.websiteoptimization.com/speed/tweak/psychology-web-performance/ [Retrieved on November 29, 2010].

[116] Fiona Nah. A study on tolerable waiting time: how long are Web users willing to wait? *Behaviour & Information Technology*, 23(3):153–163, 2004.

[117] J. Sedayao. World Wide Web network traffic patterns. *Computer Conference, IEEE International*, 0:8, 1995.

[118] Fernando Duarte, Bernardo Mattos, Azer Bestavros, Virgilio Almeida, and Jussara Almeida. Traffic Characteristics and Communication Patterns in Blogosphere. In *International Conference on Weblogs and Social Media*. online, March 2007.

[119] Fabian Schneider, Sachin Agarwal, Tansu Alpcan, and Anja Feldmann. The New Web: Characterizing AJAX Traffic. In *Proceedings of the 9th International Conference on Passive and Active Network Measurement*, volume 4979 of *Lecture Notes in Computer Science*, pages 31–40, New York, NY, USA, April 2008. Springer-Verlag Berlin Heidelberg.

[120] Facebook Inc. Facebook web site [online]. URI: http://wwww.facebook.com [Retrieved on November 29, 2010].

[121] Microsoft Corporation. Cache Array Routing Protocol (CARP) v1.0 Specifications. technical whitepaper, Microsoft Corporation, August 1999.

[122] Squid Cache Project. Squid Cache home page [online]. URI: http://www.squid-cache.org/ [Retrieved on November 29, 2010].

[123] Y. Tian, K. Xu, and N. Ansari. TCP in wireless environments: problems and solutions. *Communications Magazine, IEEE*, 43(3):S27–S32, 2005.

[124] OMNeT++ Community. OMET++ home page [online]. URI: http://www.omnetpp.org/ [Retrieved on November 29, 2010].

[125] Politecnico di Torino. P2PTVSim [online]. URI: http://www.napa-wine.eu/cgi-bin/twiki/view/Public/P2PTVSim [Retrieved on November 29, 2010].

[126] J. Banks, B.L. Nelson, and D.M. Nicol. *Discrete-event system simulation*. Prentice Hall, 2009.

[127] University Stuttgart, IKR. IKRSimLib. http://www.ikr.uni-stuttgart.de/INDSimLib/Resources/ [last checked: 2010-01-04].

[128] Simon Schütz, Lars Eggert, Stefan Schmid, and Marcus Brunner. Protocol enhancements for intermittently connected hosts. *ACM Computer Communications Review*, 35:5–18, 2005.

[129] Martin Stiemerling and Sebastian Kiesel. A System for Peer-to-Peer Video Streaming in Resource Constrained Mobile Environments. In *Proceedings of ACM Workshop on User-provided Networking U-NET'09 (co-located with the ACM CoNext 2009)*, Rome, Italy, December 2009.

[130] Martin Stiemerling and Sebastian Kiesel. Cooperative P2P Video Streaming for Mobile Peers. In *Proceedings of IEEE ICCCN 2010 Track on Multimedia and Peer-to-Peer Networking (MP2P)*, Zurich, Switzerland, August 2010.

[131] P. Dhungel, X. Hei, K.W. Ross, and N. Saxena. The pollution attack in P2P live video streaming: measurement results and defenses. In *Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV*, pages 323–328. ACM, 2007.

[132] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. Overview of the scalable video coding extension of the h.264/avc standard. *IEEE Trans. Circuits Syst. Video Techn.*, 17(9):1103–1120, 2007.

[133] V.K. Goyal. Multiple description coding: Compression meets the network. In *IEEE Signal Processing Magazine*, April 2002.

[134] E. Akyol, A.M. Tekalp, and M.R. Civanlar. A flexible multiple description coding framework for adaptive peer-to-peer video streaming. *IEEE Journal of Selected Topics in Signal Processing*, 1(2):231–245, 2007.

[HTML] W3C. HTML 4.01 Specification. W3c standard, W3C, December 1999.

[RFC 791] J. Postel. Internet Protocol. RFC 791, IETF, September 1981.

[RFC 793] J. Postel. Transmission Control Protocol. RFC 793, IETF, September 1981.

[RFC 959] J. Postel and J. Reynolds. File Transfer Protocol. RFC 959, IETF, October 1985.

[RFC 1122] R. Braden (Editor). Requirements for Internet Hosts – Communication Layers. RFC 1122, IETF, October 1989.

[RFC 1661] W. Simpson (Editor). The Point-to-Point Protocol (PPP). RFC 1661, IETF, July 1994.

[RFC 1990] K. Sklower, B. Lloyd, G. McGregor, D. Carr, and T. Coradetti. The PPP Multilink Protocol (MP). RFC 1990, IETF, August 1996.

[RFC 1998] E. Chen and T. Bates. An Application of the BGP Community Attribute in Multi-home Routing. RFC 1998, IETF, August 1996.

[RFC 2018] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgment Options. RFC 2018, IETF, October 1996.

[RFC 2045] N. Freed and N. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. RFC 2045, IETF, November 1996.

[RFC 2046] N. Freed and N. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. RFC 2046, IETF, November 1996.

[RFC 2131] R. Droms. Dynamic Host Configuration Protocol. RFC 2131, IETF, March 1997.

[RFC 2186] D. Wessels and K. Claffy. Internet Cache Protocol (ICP), version 2. RFC 2186, IETF, September 1997.

[RFC 2260] T. Bates and Y. Rekhter. Scalable Support for Multi-homed Multi-provider Connectivity. RFC 2260, IETF, January 1998.

[RFC 2326] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). RFC 2326, IETF, April 1998.

[RFC 2460] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, IETF, December 1998.

[RFC 2461] T. Narten, E. Nordmark, and W. Simpson. Neighbor Discovery for IP Version 6 (IPv6). RFC 2461, IETF, December 1998.

[RFC 2581] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC 2581, IETF, April 1999.

[RFC 2608] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service Location Protocol, Version 2. RFC 2608, IETF, June 1999.

[RFC 2616] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, IETF, June 1999.

[RFC 2756] P. Vixie and D. Wessels. Hyper Text Caching Protocol (HTCP/0.0). RFC 2756, IETF, January 2000.

[RFC 2887] M. Handley, S. Floyd, B. Whetten, R. Kermode, L. Vicisano, and M. Luby. The Reliable Multicast Design Space for Bulk Data Transfer. RFC 2887, IETF, August 2000.

[RFC 3259] J. Ott, C. Perkins, and D. Kutscher. A Message Bus for Local Coordination. RFC 3259, IETF, April 2002.

[RFC 3261] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, IETF, June 2002.

[RFC 3315] R. Droms (Editor), J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). RFC 3315, IETF, July 2003.

[RFC 3344] C. Perkins (Editor). IP Mobility Support for IPv4. RFC 3344, IETF, August 2002.

[RFC 3550] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, IETF, July 2003.

[RFC 3755] S. Weiler. Legacy Resolver Compatibility for Delegation Signer (DS). RFC 3755, IETF, May 2004.

[RFC 3927] S. Cheshire, B. Aboba, and E. Guttman. Dynamic Configuration of IPv4 Link-Local Addresses. RFC 3927, IETF, May 2005.

[RFC 4227] E. O'Tuathail and M. Rose. Using the Simple Object Access Protocol (SOAP) in Blocks Extensible Exchange Protocol (BEEP). RFC 4227, IETF, January 2006.

[RFC 4271] Y. Rekhter (Editor), T. Li (Editor), and S. Hares (Editor). A Border Gateway Protocol 4 (BGP-4). RFC 4271, IETF, January 2006.

[RFC 4301] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301, IETF, December 2005.

[RFC 4423] R. Moskowitz and P. Nikander. Host Identity Protocol (HIP) Architecture. RFC 4423, IETF, May 2006.

[RFC 4838] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-Tolerant Networking Architecture. RFC 4838, IETF, April 2007.

[RFC 4862] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration. RFC 4862, IETF, September 2007.

[RFC 4885] T. Ernst and H-Y. Lach. Network Mobility Support Terminology. RFC 4885, IETF, July 2007.

[RFC 4960] R. Stewart (Editor). Stream Control Transmission Protocol. RFC 4960, IETF, September 2007.

[RFC 5186] B. Haberman and J. Martin. Internet Group Management Protocol Version 3 (IGMPv3) / Multicast Listener Discovery Version 2 (MLDv2) and Multicast Routing Protocol Interaction. RFC 5186, IETF, May 2008.

[RFC 5206] P. Nikander, T. Henderson (Editor), C. Vogt, and J. Arkko. End-Host Mobility and Multihoming with the Host Identity Protocol. RFC 5206, IETF, April 2008.

[RFC 5246] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, IETF, August 2008.

[RFC 5348] S. Floyd, M. Handley, J. Padhye, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification. RFC 5348, IETF, September 2008.

[RFC 5401] B. Adamson, C. Bormann, M. Handley, and J. Macker. Multicast Negative-Acknowledgment (NACK) Building Blocks. RFC 5401, IETF, November 2008.

[RFC 5740] B. Adamson, C. Bormann, M. Handley, and J. Macker. NACK-Oriented Reliable Multicast (NORM) Transport Protocol. RFC 5740, IETF, November 2009.

[RFC 5770] M. Komu, T. Henderson, H. Tschofenig, J. Melen, and A. Keranen (Editor). Basic Host Identity Protocol (HIP) Extensions for Traversal of Network Address Translators. RFC 5770, IETF, April 2010.

[RFC 5771] M. Cotton, L. Vegoda, and D. Meyer. IANA Guidelines for IPv4 Multicast Address Assignments. RFC 5771, IETF, March 2010.

[mDNS] Stuart Cheshire and Marc Krochmal. Multicast DNS. Internet draft (work in progress), draft-cheshire-dnsext-multicastdns-11, IETF, March 2010.

[TS 25.306a] 3GPP WG R2. UE Radio Access capabilities. TS 25.306 ver. 7.3.0, 3GPP.

[TS 25.306b] 3GPP WG R2. UE Radio Access capabilities. TS 25.306 ver. 6.9.0, 3GPP, September 2006.

Acknowledgments

I would like to express my gratitude to Prof. Dr. Fu as my supervisor of the PhD thesis, who gave me excellent feedback and criticism. This thesis would not have been possible without the permission and support of my employer, the NEC Laboratories Europe, allowing me to work on the thesis in parallel to my full-time employment. As special thank you to Dr. Stüttgen, Dr. Quittek, Dr. Brunner, Dr. Niccolini of NEC who supported this.

I would like to thank Prof. Dr. Grabowski, Prof. Dr. Hogrefe, Prof. Dr. Kurth, Prof. Dr. Morgenstern, and Prof. Dr. Sax for being members of the thesis committee for the PhD defense, as well as Prof. Dr. Schulzrinne who served as external reviewer and provided valuable feedback.

Many thanks to Ralf Schmitz for his review of the thesis and feedback; to Sebastian Kiesel for the technical discussions, a lot of fun, and the LaTex thesis template; and to Silke Lampson for her encouraging words during the thesis.

Last but not least, a special gratitude to my wife Kirsten who supported me in an unprecedented way during the years of the thesis and to my boys Maxim and Jacob for their patience with their father of not having time to play with them.