A Pricing Approach to Distributed Resource Allocation in IP and Peer-to-Peer Networks

Vom Promotionsausschuss der
Technischen Universität Hamburg-Harburg
zur Erlangung des akademischen Grades
Doktor-Ingenieur
genehmigte Dissertation

(PhD thesis)

von
Kolja Eger
aus Hannover

2008

1.Gutachter (referee): Prof. Dr. Ulrich Killat
2.Gutachter (referee): Prof. Dr. Dieter Gollmann

Tag der mündlichen Prüfung (date of oral examination): 11.07.2008

Printed copy available from Mensch&Buch Verlag
ISBN 978-3-86664-487-8

Acknowledgements

My thanks to all of you who supported this work!

Ein herzlicher Dank geht dabei an Prof. Dr. Ulrich Killat nicht nur für die Betreuung meiner Doktorarbeit, sondern auch für die schöne Zeit an seinem Institut.

Vielen Dank an die Kollegen vom Institut für Kommunikationsnetze der TUHH für viele spannende Diskussionen und eine angenehme Arbeitsatmosphäre.

Tausend Dank an Freunde und Familie, meine Eltern und meine Frau für die bedingungslose Unterstützung.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rapid progress of computer networks and in particular the Internet poses new challenges for research and development in this field. The number of Internet users is estimated in September 2007 around 1200 million people and has increased since the turn of the millennium by around 244% [IWS]. With optical components in the core and increased availability of fast digital subscriber lines as access technology also the available bandwidth in the networks have multiplied over the last years. Further on, with the convergence of the networks to a single IP-based network, which carries voice, data and multimedia traffic and offers access to fixed as well as wireless terminals, protocols have to adapt to changing requirements.

The steady change of the technology and of the use of the Internet gives rise to continuously revising and refining existing concepts. In order to cope with this challenge sophisticated models of the networks and their functions are necessary. A mathematical framework for rate control at the transport layer proved to be effective not only for a deeper understanding of the Transmission Control Protocol (TCP) and the development of improved TCP versions but also in general for the optimisation of the resource allocation in communication networks. The basic idea is to model the resource allocation as global optimisation problem and to decompose the problem into sub-problems. These sub-problems are solved with exchange of minimal control information on different network entities and different network layers. The approach is based on economic models. Therefore, it is denoted as resource or congestion pricing. This thesis extends the resource pricing approach and deploys it to Peer-to-Peer (P2P) networks.

P2P networks gained in importance over the last years and most of today's Internet traffic is caused by different P2P protocols. In contrast to the traditional client-server architecture, which relies on a small number of powerful servers, the basic idea behind P2P is

that each peer not only accesses resources but also contributes resources to the network. Furthermore, P2P relies on self-organisation and avoids centralised control. Hence, robust, resilient and scalable P2P networks can be realised. Although P2P networks became popular with legally questionable applications, meanwhile serious business cases emerge which realise services cost effective and more efficient as compared to the client-server architecture. For a further success of the P2P paradigm sophisticated models of P2P networks and the design of optimal protocols are necessary. By applying the resource pricing framework to P2P networks this work presents a novel approach to the resource allocation in these architectures. The proposed protocols proved to be efficient, fair, stable and scalable and improve the state of the art in this field.

1.1 Contribution

The seminal paper [KMT98] by F. Kelly et al. proposes resource (or congestion) pricing for rate control in IP networks. It provides the basis of extensive research on rate control and similar models for the resource allocation in communication networks. Also this work builds on the resource pricing approach in [KMT98]. It studies the extension of the model in [Kel97, KMT98] to dynamic routing at the network layer. It is shown by simulations that the proposed distributed algorithm is efficient and fair for multi-path routing. However, for single-path routing the algorithm shows instability. The work in this area is published in [EK06b].

The main contributions of this thesis are in the field of P2P networks. Although P2P applications are very popular and cause large amounts of today's Internet traffic most of the widely adopted P2P protocols do not follow sophisticated design criteria. Properties like efficiency and fairness of the corresponding networks are unknown or are investigated after implementation when protocols became popular. Further on, the protocol design might be driven by objectives of the users or P2P service providers. These can disagree with the objectives of the network operators. To sum up, the need for sophisticated models of P2P networks becomes obvious. This thesis addresses this issue by applying resource pricing to P2P networks. It models the resource allocation in these overlays as optimisation problem. By applying Lagrange multipliers the global problem is decomposed into sub-problems. Further on, the sub-problems are used to derive distributed algorithms. The basic idea of applying resource pricing to P2P networks is published for the first time in [EK05]. Further publications in [EK06c] and [EK07a] present the P2P network model and the development of the distributed algorithm in more detail. An analytical

result under connectivity assumptions and the stability of the distributed algorithm are shown in [EK07b].

Additionally, the resource pricing approach is discussed for P2P Content Distribution Networks (CDNs) and is compared to the tit-for-tat scheme of the BitTorrent implementation. The steady-state performance with resource pricing ensures fairness and outperforms the BitTorrent implementation. Fairness is of great importance because it provides an incentive to peers to contribute resources to the network. Furthermore, the steady-state performance of resource pricing provides a Nash equilibrium. This means a peer can not improve its performance by changing its own strategy. Hence, resources can be allocated in a fair manner even when peers behave selfishly. In summary, the proposed distributed algorithm converges to the solution of the global optimisation problem. Hence, the algorithm provides an efficient, fair, stable and scalable way for the resource allocation in P2P CDNs. These findings were published in [EK06a] and [EK08]. Additionally, [EHBK07] gives insights into the simulation of BitTorrent-like networks.

A further contribution of this thesis is to study P2P over IP networks as a cross-layer optimisation problem. Since resource pricing for rate control and for P2P networks is based on the same global optimisation problem both functions can be studied in a single model. We propose a novel TCP variant that includes information from the P2P network. Hence, fairness between peers is ensured even when some routes between peers are congested. In such a case a sending peer shifts traffic from a congested route to an uncongested one. The change in the rate allocation is balanced by other peers. This shows that overlay networks have the potential to improve the performance of the underlying IP network and to overcome the rigid design of IP routing. This part of the thesis is published in [EK07c].

1.2 Outline

This work is structured as follows. Chapter 2 gives an introduction to IP and P2P networks. It presents important concepts on which this work builds on. These are the congestion control of TCP and routing protocols in IP networks. Furthermore, functions of the network are implemented nowadays more frequently at the network edges. P2P networks are a good example. Different P2P architectures and applications are discussed and compared to the traditional client-server model. Especially, the basic principles of P2P content distribution networks and BitTorrent as an example are important for this work and are outlined. Chapter 2 ends with the basic idea of the resource pricing framework, its strengths and weaknesses and different application scenarios.

Chapter 3 discusses resource pricing in more detail and presents the mathematical background. The model includes the routing of the IP or P2P network. It gives a short introduction to dual theory and presents the Lagrange function of the optimisation problem, which is the basis for a decomposition of the problem. For a simplified model, which is frequently adopted for P2P networks, the solution of the optimisation problem is computed under specific connectivity assumptions. Additionally, different fairness criteria are presented.

Chapter 4 applies resource pricing for a combined approach of rate control and routing in IP networks. The resource allocation is modelled as cross-layer optimisation problem of the network and transport layer. The distributed algorithms for rate control from the literature are extended to routing. In the first step multi-path routing is investigated. Here, the sending rates over multiple routes are adapted to network congestion and the fairness criteria. In the second step single-path routing with and without Equal-Cost Multi-Path (ECMP) is studied.

Chapter 5 applies the resource pricing framework to P2P networks. The optimisation problem is adapted to entities of the P2P network and new distributed algorithms are derived from the Lagrange function. Efficiency and stability of the distributed algorithm are shown analytically and by simulations. By using the proposed algorithm for P2P content distribution fairness between peers can be ensured and BitTorrent's tit-for-tat strategy is outperformed. Furthermore, the interaction of resource pricing and the piece selection is studied.

Chapter 6 studies P2P over IP networks as a cross-layer optimisation problem. It combines the distributed algorithms from Chapter 4 and Chapter 5 and proposes a TCP variant for P2P networks. Hence, routing in the P2P network is used to avoid congestion in the IP network and to ensure fairness between peers at the same time. The basic functionality of this approach is shown by simulations.

Chapter 7 summarises this work and highlights the main findings.

Chapter 2

State of the art

This chapter gives an introduction to algorithms and concepts of today's Internet, which are of importance for this work. We discuss the rate control algorithm of the Transmission Control Protocol (TCP) and routing concepts used in IP (Internet Protocol) networks. Both functions have major influence on the usage of resources in an IP network.

Additionally, we discuss Peer-to-Peer (P2P) networks, which gained in popularity in recent years. Many different kinds of applications are provided by P2P networks. In this work we concentrate on file-sharing and content distribution applications. P2P protocols form overlay networks and control the allocation of resources of peers. Hence, functions in P2P networks affect the algorithms of the underlying IP network and vice versa.

The topics discussed in this work cover different network layers. As depicted in Figure 2.1 IP and TCP are protocols of the network and transport layer of the ISO OSI reference

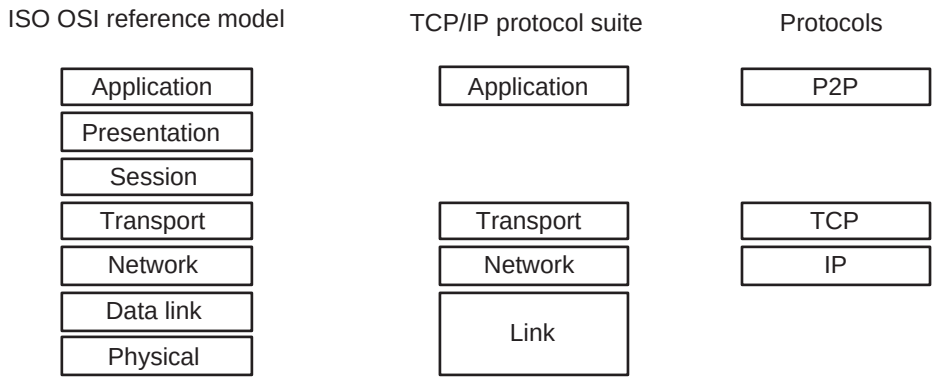


Figure 2.1: Network layers

model [Tan02] as well as the TCP/IP protocol suite [Ste94], respectively. P2P networks are realised as overlays at the application layer.

In general, distributed algorithms avoid centralised control and disperse the functions over different network entities. Hence, they have the potential to scale well. *Scalability* refers to the ability to adapt to different network sizes and to operate on a wide range of different scales. This is an important design criterion for Internet protocols because the Internet is steadily growing with respect to nodes, users and traffic volume. A mathematical framework for the allocation of resources in a distributed environment is resource (or congestion) pricing. Firstly used to study TCP-like algorithms it is extended and adapted to a wide range of other scenarios. Before discussing resource pricing for IP and P2P networks in the following chapters we present the basic idea as well as the strengths and weaknesses of this framework in general.

2.1 IP Networks

IP networks and the Internet, its most popular example, are packet-switched networks, which use the Internet Protocol (IP). They are used to carry a wide range of different kinds of traffic and enable various services. Email, web browsing, file transfer and instant messaging as well as newer services like Voice over IP (VoIP), IPTV and Video on Demand (VoD) are popular examples. In all cases the data to be sent is split into packets and is routed on a path from source to destination. Packets share the medium with packets from other traffic sources. Hence, they might be queued in a buffer when an intermediate router is busy or even be dropped when congestion occurs.

In general, the traffic from different kinds of applications is characterised as elastic or inelastic traffic [She95]. Elastic traffic (e.g. for a file transfer) tolerates delays and packet drops. Hence, its transmission rate can be adapted to the network conditions. On the other hand inelastic traffic (e.g. for VoIP) has more stringent requirements with respect to delay as well as the sending rate. Hence, these applications prefer constant bit rates and use protocols like UDP (User Datagram Protocol), which do not react to congestion signals*.

*More sophisticated approaches for streaming traffic like adaptive encoding [RHE00] which reacts on congestion in the network by changing the codec are out of the scope of this work.

2.1.1 Congestion Control

Congestion control for elastic traffic in IP networks is mainly done by TCP, which controls the sending rate of a flow by limiting the window size, the amount of outstanding but unacknowledged data. The basic principles are outlined in various textbooks (e.g. [PD07, Tan02, Ste94]). Briefly, it is a self-clocking mechanism, which adapts the window size on receiving acknowledgements. In general, TCP consists of three phases: Slow start, congestion avoidance and exponential backoff. At start-up and after multiple packet losses the so-called slow start algorithm controls the congestion window, whereas exponential backoff is used under severe congestion. Hence, loosely speaking congestion avoidance controls the window size under normal network conditions.

Different variants of TCP are proposed in the literature and are used in the Internet. Most important are TCP Tahoe (based on [Jac88]), TCP Reno (extends Tahoe with ideas from [Jac90]), TCP NewReno (extends Reno by ideas from [Hoe95] and is specified in [FH99]) and TCP Vegas [BP95]. In recent years TCP variants for networks with high bandwidths and large propagation delays are developed (e.g. Scalable TCP [Kel03], High-Speed TCP [Flo03] and FAST TCP [JWL04]). The review in [DBEB06] presents an overview of the development and the modifications of TCP.

All TCP variants have a basic design principle in common, which is described in [Jac88]. Basically, the congestion control consists of two components. The routers in the network measure the level of congestion and signal this information to the end-points of the connection. When congestion signals are received by the sender it has to decrease its sending rate. If no congestion signal is received, the sending rate can be increased. TCP algorithms differ in the rules for adapting the sending rate as well as the used congestion signal. TCP Vegas and FAST TCP use queuing delay as a measure of congestion, whereas the other algorithms mentioned above use packet loss. Basically, TCP Vegas and FAST TCP determine the difference between the smallest round-trip time (RTT) observed so far to the actual RTT. Hence, a deficiency is the decrease in rate due to RTT changes which are not caused by congestion, e.g. due to re-routing. On the other hand delay variations are multi-bit information of the congestion, which enables a diverse reaction to it.

Packet loss gives binary information only and occurs when the buffer of a router overflows. With Active Queue Management (AQM) an earlier notification of congestion is realised. An example is Random Early Detection (RED) [FJ93], where packets are dropped with a certain probability before the buffer overflows. Here, the dropping probability increases with increasing average queue length. Alternatively, packets can be marked to notify the sender about congestion. With [RFB01] the TCP/IP protocol stack is extended

by the Explicit Congestion Notification (ECN). Thus, single bit information about congestion can be conveyed in the network. Briefly, routers set the congestion experienced bit in the IP header and the receiving TCP agent copies this bit to the TCP header of the acknowledgement and sends it to the traffic source (see [RFB01] for details).

A large body of research (e.g. [GK99b, AL00, ZK02]) studies marking strategies to improve the rate control of TCP by using the ECN bit. Furthermore, other approaches like the eXplicit Control Protocol (XCP) [KHR02] use explicit congestion information in the packet headers. However, these schemes have to be supported by both end-points of the connection as well as the intermediate routers. For the size of the Internet and with its steady growth the adoption of new technologies is slow. For instance, ECN is a proposed standard by the Internet Engineering Task Force (IETF) since 2001 [RFB01], but a measurement study [MAF05] in 2005 shows that only 2.1% of the inspected web servers support it.

The design of transport protocols at packet-level can be augmented by the analysis at flow-level. Here, flow-level models abstract from the details of every packet and consider flows with specific sending rates instead. Especially, properties like fairness and stability are easier to study at flow-level. Hence, TCP protocols designed at packet-level are converted and analysed at flow-level [Key01, LPD02] and new protocols like FAST TCP [JWL04] are based on a flow-level model and afterwards are implemented at packet-level. Frequently, resource pricing is used as flow-level model.

2.1.2 Routing

The Internet is an interconnection of a large number of smaller networks. One or multiple of these networks are denoted as an autonomous system (AS), when it is controlled by a single entity [Tan02]. Routing packets from source to destination in a single AS is based on different protocols as compared to routing between different autonomous systems. Interior gateway protocols are used inside a single AS whereas exterior gateway protocols like BGP (Border Gateway Protocol) specify the routing between autonomous systems. Furthermore, exterior gateway protocols highly depend on policies and negotiations between the operators of different autonomous systems.

This work concentrates on interior gateway protocols. Here, routing is based on shortest path algorithms. Hence, with respect to a chosen metric the shortest route between source and destination is computed (e.g. with the Dijkstra algorithm). The OSPF (Open Shortest Path First) protocol [Moy98] is a popular example for an interior gateway protocol which

uses the Dijkstra algorithm. It belongs to the class of link state routing algorithms. The basic concept of link state routing is as follows. Each node periodically broadcasts information to the network about its direct neighbours and the respective costs to reach them. Thus, each node accumulates information about all other nodes and can reconstruct the topology of the network as a weighted directed graph[†]. Based on this network map the shortest path to every other node is computed and the routing table, which specifies the next hop to every possible destination, is constructed.

The shortest path in a network might not be unique and instead of storing only one of the shortest paths Equal-Cost Multi-Path (ECMP) can be employed. With ECMP the next hop of several (up to all) routes with the shortest path are stored and used for forwarding. This provides some kind of load-balancing, but can cause also problems on higher layers. For example, multiple routes might have different delays and packets are received out of order at the sender. This influences the congestion control of TCP.

Link state routing is a hop-by-hop routing scheme, where each router decides about the next hop of a packet. Although hop-by-hop routing is the common practice in the Internet, also source routing is supported by the IP protocol [Tan02]. Here, the source specifies the route for the packet in the IP header. Furthermore, source routing is used in alternative routing architectures like Nimrod [CCS96].

2.2 Application-oriented Protocols

The Internet is a steadily growing and evolving network of networks. In 2007 it is used by more than 1200 million people [IWS] and connects more than 433 million hosts world wide [ISC]. Furthermore, the Internet carries nowadays not only a much larger amount of traffic but also traffic from applications with demands different from those in the early days. Moreover the design of new protocols might be primarily driven by the requirements of the applications. These could disagree with the objectives of the network design.

One example is the usage of multiple parallel TCP connections. Among other things, this technique is used by GridFTP [All03] to transfer data up to the sizes of tera- and petabytes. Although the throughput can be increased with parallel connections, it is unfair to other flows and a trade-off between efficiency and fairness has to be done [HNA04]. Consequently, a protocol designer can act selfishly by maximising the throughput for its application or cooperatively by taking network-wide considerations into account.

[†]OSPF is more complicated than the basic link state concept, because it supports further level of abstractions by defining network areas. See [Moy98] for details.

Furthermore, parallel TCP connections and others challenge the design criteria of protocols in general. As an example, fairness is commonly studied with respect to flows. The number of flows used by a single user is not taken into account. According to [Bri07] flow fairness in today's Internet is inappropriate. Instead cost fairness should be used, which depends on the congestion each user generates with all its flows. This describes how much the transfers of each user restrict other transfers. Furthermore, the idea of cost fairness is part of the resource pricing approach.

Similar observations with respect to fairness are true for routing in overlay networks like Resilient Overlay Network (RON) [ABKM01]. Here, an overlay network is formed and data transfers can be relayed by a node of this overlay. By probing the direct and relayed connections the best path between source and destination is chosen. Thus, routing is done on different levels with different objectives. In the IP network the operator optimises the operation of the network whereas routing in the overlay is driven by selfish objectives of the application. The consequences of selfish routing strategies is studied in [QYZS06]. Here, it is shown that traffic engineering by the network operator and selfish routing by e.g. RON can decrement the performance of the network considerably.

Selfish behaviour can be analysed with game theory. Instead of assuming cooperation among users the game theory is based on players with different interests. Thus, players behave selfishly and adopt strategies that maximise their benefit. Game theory provides tools to analyse the effects of non-cooperative behaviour and evaluates the effectiveness of incentive schemes for behaving cooperatively. Game theoretical approaches are used to study congestion control [ASK+02, LAPF05] and routing [QYZS06]. Hence, with the adoption of application-oriented protocols it becomes an important tool for network design.

Application-oriented protocols are widely used in the Internet already. Popular examples are the Skype application, which relays voice calls [SFKT06], or P2P networks in general, which are discussed in detail in the next section. In conclusion, more applications use specific protocols and/or overlay networks to achieve selfish objectives. The interaction with and the consequences for the underlying IP network are not fully understood. Furthermore, design criteria are not yet adapted to these new approaches.

2.3 Peer-to-Peer Networks

The term Peer-to-Peer (P2P) is used for a wide field of different technologies and applications. A general definition can be found in [SE05]: *A peer-to-peer system is a self-*

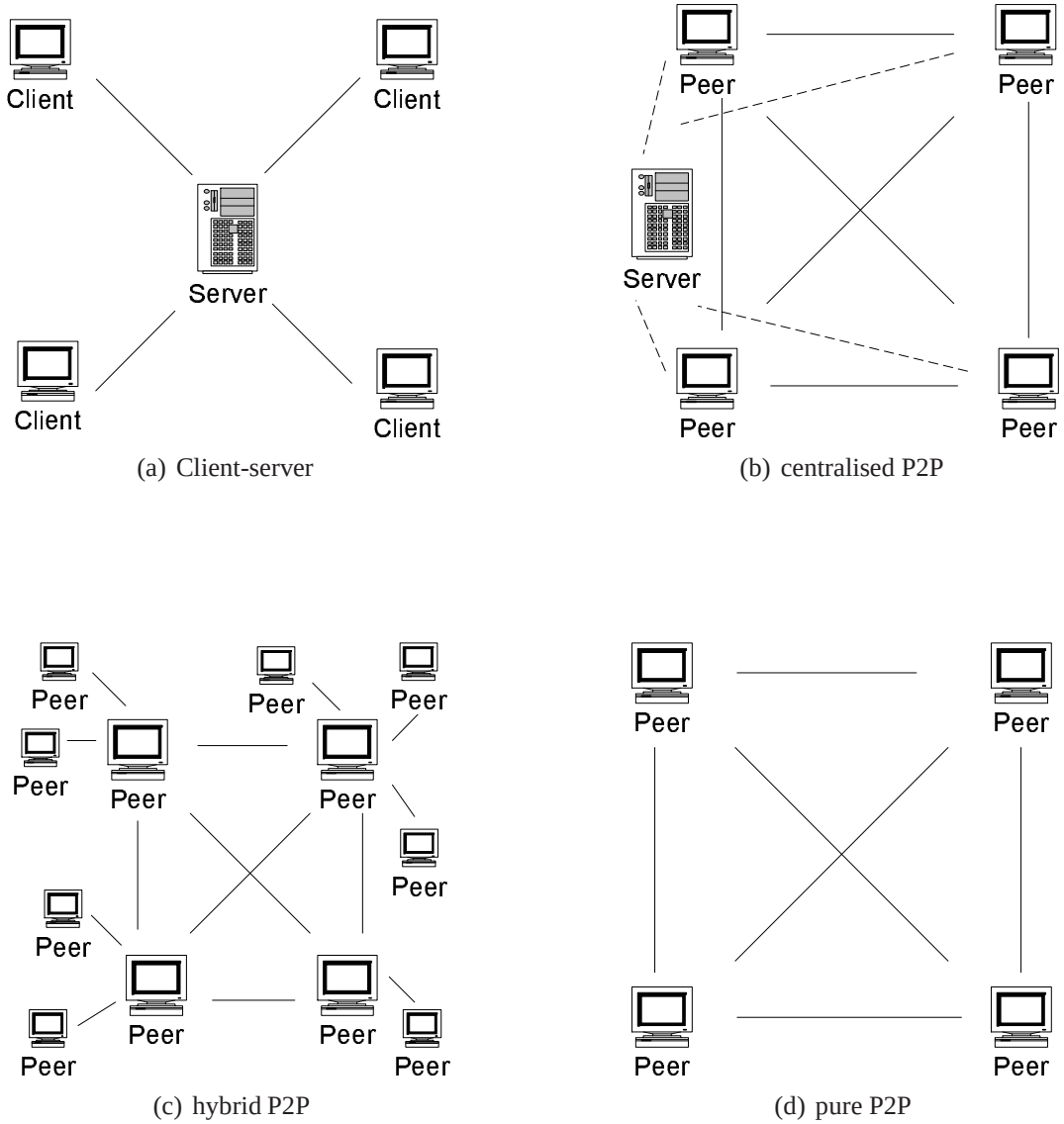
organising system of equal, autonomous entities (peers) which aims for the shared usage of distributed resources in a networked environment avoiding central services.

Furthermore, P2P networks are realised as logical overlay networks. The services are not provided by specific servers as in the client-server architecture, but each peer requests and provides services. Hence, each peer is a client and a server. By not only consuming resources but also providing them to the network, P2P networks have the potential to scale well. Further on, by replicating services in a P2P network the resilience can be increased as compared to centralised systems like the client-server model. But not all applications which are denoted as P2P rely solely on the P2P concepts. Also many applications use a centralised P2P architecture where servers run specific tasks. Further on, in hybrid P2P architectures some of the peers adopt dynamically special functionalities. Thus, the topology of the overlay network is hierarchical. As depicted in Figure 2.2 the client-server, centralised, hybrid and pure P2P architectures are a change from centralised over partially to fully decentralised designs. Examples of the different P2P architectures are given in the following.

P2P networks became popular with file-sharing applications. In 1999 home users started to offer content with Napster [Nap]. The original Napster implementation is a good example for a centralised P2P application. The shared files are indexed on a server and also all search requests are directed to the server. The server replies with a list of IP addresses of those peers, which store the requested content. Thereon, the transfer of the content is done directly between the peers without any support of the server.

Gnutella 0.4 [Gnu] is an example for a pure P2P application. Content is searched fully decentralised by broadcasting (or flooding) query messages throughout the overlay. Although each query message is relayed by other peers only for a limited number of times, Gnutella 0.4 produces large control overhead [Mal06]. To reduce the signalling traffic the succeeding Gnutella 0.6 version introduced ultrapeers and leaf nodes. In this hierarchical design a leaf node connects to one ultrapeer (or to overcome failures to several ultrapeers) and informs it about the shared content. Also search requests are sent to the ultrapeer. The ultrapeers are connected among each other and perform search requests for their leaf nodes. The communication between ultrapeers is similar to Gnutella 0.4. Since not all peers behave like ultrapeers the signalling overhead is reduced as compared to Gnutella 0.4.

In general, a P2P system for file-sharing has to define functions for constructing and maintaining the overlay topology, to search and request for content and to transfer the content between peers. Some P2P applications like BitTorrent [Coh03] are optimised for



(a) Client-server (b) centralised P2P

(c) hybrid P2P (d) pure P2P

Figure 2.2: From centralised to decentralised architectures

the dissemination of the content. Hence, they are also denoted as P2P Content Distribution Networks (CDNs). This work discusses algorithms for P2P CDNs. Thus, concepts for these networks are discussed in detail in the following section.

Besides file-sharing P2P concepts are also used for other application types. A promising approach is application multicast. Since multicast is not widely provided by the IP network, the peers form an overlay to distribute streaming content between each other. Commonly, tree structures are formed where the content is sent from the root relayed by the interior nodes to all leaf nodes. [YLE04] provides a survey about this topic.

Also in the field of Grid computing (distributed computing) the term P2P is used [SE05]. Here, the resource of interest is computing power. During idle times peers provide its computing resources to the network and request additional processor cycles from other machines in high load situations.

Another intensive field of P2P research is about Distributed Hash Tables (DHT). DHTs provide a distributed look-up service. P2P networks with DHTs are denoted as structured P2P networks, because a common address space is used to map nodes and data items into a logical structure. Hence, each node maintains connections to specific other nodes and is responsible of storing specific data items. DHTs show good scalability and route requests over a small number of other peers. They are used in different P2P applications, e.g. as an extension of BitTorrent [BTb] to exchange information about other peers in the network and in the application-level multicast application Scribe [CDKR02] to build the multicast tree. Further information about DHTs can be found for example in [LCP+05] and [SE05].

2.3.1 P2P Content Distribution Networks

The objective of P2P Content Distribution Networks (CDNs) is to disseminate one large file (or a composition of large files) to a large number of users in an efficient way. BitTorrent [Coh03] is a popular example for P2P content distribution. According to [Par05] BitTorrent accounted for 30% of all traffic on the Internet in 2005.

A download with BitTorrent starts by getting a metainfo file (*torrent*), e.g. from a web server. The metainfo file provides information about the file itself, hash values to verify data integrity and the IP address of a so-called tracker. The tracker is a centralised component which stores information about all peers in the overlay network. A new peer, that enters the network, asks the tracker for a list of active peers in the overlay. The tracker returns a random subset to the requesting peer. Furthermore, an active peer contacts the

tracker from time to time to obtain information about new peers in the network. An extension [BTb] of the protocol also incorporates the exchange of information about other peers in the network between connected peers. This is realised by a DHT and is often stated as trackerless BitTorrent.

BitTorrent specifies the messages between the tracker and a peer and between peers themselves. Furthermore, it implements two important concepts which improve the file dissemination. These are an incentive scheme and the multi-source download (or swarming principle). Since both concepts are not part of the protocol but in fact part of the first specification, applications, which use the BitTorrent protocol, implement them differently. The following descriptions are based on the original specification [Coh03, BTa].

Incentive scheme

P2P networks face the problem of free-riding [AH00] where peers consume resources solely without contributing anything to the network. To reduce free-riding BitTorrent implements an incentive scheme.

In general, the network consists of two types of peers. On the one hand peers that have a full copy of the file and on the other hand peers that are still downloading the file. These peers are denoted in the BitTorrent specification [BTa] as seeds and leechers, respectively. Hence, a leecher becomes a seed when it completes the download. The basic idea of BitTorrent is that the download performance depends on the upload rate. This provides an incentive to leechers even when they behave selfishly and try to minimise their download time. Consequently, seeds have no incentive and upload to others altruistically, because they have completed the download of the file already.

To realise the incentive scheme each peer controls to whom it uploads data. This peer selection is called *choking/unchoking*. When a remote peer is selected for upload an *UNCHOKE* message is sent. An upload is stopped with a *CHOKE* message. Each peer uploads to a fixed number of other peers (the default value is four) and chooses to upload to others from which it gets the highest download rates. This principle is called tit-for-tat, because it is based on reciprocity. Unchoking by download rates is not applicable for seeds. Therefore, the peer selection of the seeds is based on the upload rates to the connected peers. This is driven by the idea that high upload rates are only achieved when no one else uploads to the peer. Hence, those peers should be preferred which are not served by others.

By default this tit-for-tat strategy is run every ten seconds by every peer, whereby the

download rates are determined by a moving average over the last 20 seconds. To discover new peers with better performance a so-called optimistic unchoke is done additionally. Here, one of the peers is unchoked independently of its rate. The optimistic unchoke is changed every 30 seconds to provide enough time to be possibly unchoked by the remote peer in return. Another rule in BitTorrent is to choke a peer when it has sent no data message in the last minute. This is called anti-snubbing.

Multi-Source Download

For a *multi-source download* the file of interest is fragmented into pieces or chunks[‡]. When a peer completes the download of a single piece, it offers it to other peers which so far have not downloaded this piece. Thus, peers exchange pieces with each other although they did not finish the download of the complete file.

Based on the piece selection rules each peer determines which piece is requested when a peer is unchoked by a remote peer. The decision process in BitTorrent follows the following rules: Firstly, when some bytes are received from a specific chunk the remaining parts of that chunk are requested. This scheme is called strict priority. Since peers forward only complete chunks (where data integrity is verified) to other peers, this mechanism ensures that chunks are completed fast. When strict priority is not applicable, the rarest chunk is requested. Since a peer has only a local view of the network it can only estimate rarity based on the information of its neighbours. This information is available to the peer by the BitTorrent messages *BITFIELD* and *HAVE* (see [BTa] for details).

When a peer has no chunk at the beginning of the download, BitTorrent deviates from the rarest-first policy and the new peer requests a piece randomly. This is intended to ensure a faster completion of the first piece such that the upload bandwidth of that peer can be used by others.

Normally, one *REQUEST* message asks for a data portion which is smaller than the chunk size. The default values in the original implementation are 256 KB as chunk size and 16 KB per request. To prevent that the sender runs out of requests and has to wait for a new request from another peer, the first requests after an unchoke are sent as a batch. By default the batch size is five requests. In normal mode a peer requests each part only once. This can become a problem at the end of the download. When the rest of the file is requested at a very slow peer, the downloading peer has to wait long although other peers may handle the request faster. Therefore, a peer can switch to the endgame mode, where it

[‡]In this work the terms piece and chunk are used interchangeable.

requests the same parts at multiple peers. Although, a peer can cancel requests at remote peers the endgame mode can consume additional bandwidth by transferring redundant parts.

With multi-source download the resources in the P2P network are used more efficiently and the network also scales for large peer populations with respect to download times. Especially, for *flash-crowd* scenarios the advantage of a multi-source download becomes obvious. Here, initially only one seed and a number of leechers are in the network (or a number of leechers enter the network in a short-time). This represents an extraordinary burden on the network because only one peer can upload data to others at the beginning.

Assume new seeds stay in the network until all peers have finished their download. For the case of peers with the same upload capacity C the download time can be estimated analytically. We denote the file size, the chunk size and the number of parallel uploads as S_F, S_C and U, respectively. After S_F/C the whole file is available in the network. The last U chunks uploaded by the seed are the rarest chunks in the network as each of them is only available at the seed and one other peer. For a uniform dissemination of the rarest chunks the seed uploads each rarest chunk once. The other peers upload the rarest chunk they hold U times. Thus, the number of peers which hold a rarest chunk increases by $\lceil (1+1/U)(U+1)^i \rceil$, where i is the number of time intervals it takes to upload a full chunk to U peers. This time interval can be computed with $U \cdot S_C/C$. With $P > 1$ peers in the network (including the initial seed) the minimal total download time is

$$t_{\text{flashcrowd}} = \begin{cases} \dfrac{S_F}{C} + \dfrac{S_C}{C} \left\lceil \log_2 \dfrac{P}{2} \right\rceil & \text{if} \quad U = 1 \\ \dfrac{S_F}{C} + \dfrac{U S_C}{C} \left\lceil \log_{U+1} \left\lfloor \dfrac{P}{1+\dfrac{1}{U}} \right\rfloor \right\rceil & \text{if} \quad U \geq 2, \end{cases} \tag{2.1}$$

where $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ denotes the ceiling and the floor function, respectively. Equation (2.1) holds when P is a multiple of $(1+1/U)(U+1)$. For other values of P it is exact if each chunk is downloaded from a single source. Similar results are derived in [YdV06] for $U = 1$. Furthermore, [MWW06] presents an approximation for heterogeneous capacities.

In [EHBK07] we showed that the performance with BitTorrent is near to the theoretical values given by (2.1). We simulated a 100 MB download with varying number of peers, which have an upload capacity of $C = 1024$ kbps. Figure 2.3 shows the mean download time and the 95% confidence intervals for 5 simulation runs. BitTorrent is studied at flow-level as well as packet-level. For the flow-level simulations only the access link bandwidth

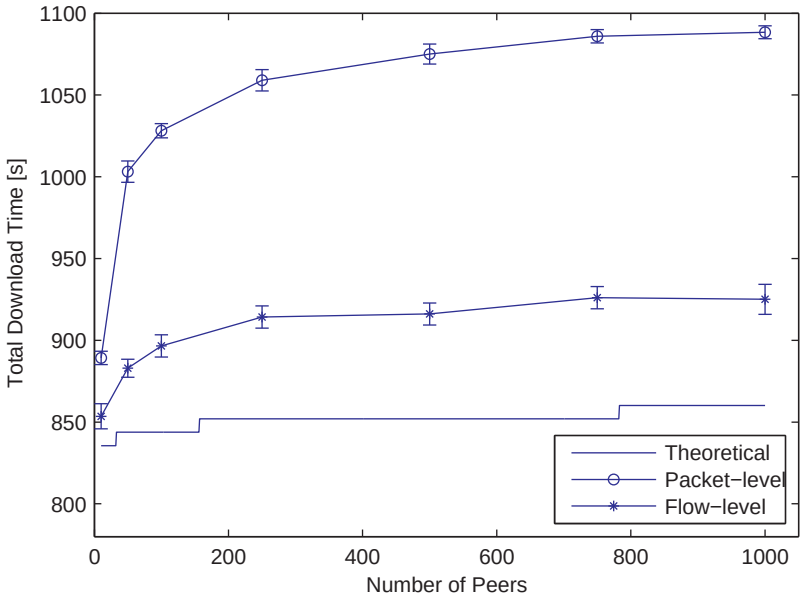


Figure 2.3: Download time with BitTorrent for a flash-crowd

is taken into account and the data transfers are modelled as flows. The packet-level simulations are more detailed and model also the TCP behaviour between the peers. Hence, the flow-level results show the overhead of the BitTorrent application whereas the packet-level results include also the overhead due to TCP.

The original content provider needs at least 819 s to upload the file to the network. Theoretically, the distribution of the data to all peers in the network is very fast and scales with an increasing number of peers. Also the simulation results reveal that the content distribution with BitTorrent scales well. At flow-level the results are only up to 9% higher than the theoretical values. Packet-level results are up to 27% higher. At flow-level the total download time for 250 peers and more remains nearly constant. Although the increase of the download time at packet-level is small, it increases by 6% from 100 to 1000 peers.

The advantage of P2P for the fast distribution of large content in the Internet as compared to the client-server architecture is obvious. While in the client-server architecture the total load must be carried by the server(s), it is distributed among the users in a P2P network. Hence, content providers save costs because traffic is shifted from their servers to the P2P network. In particular, the costs for over-provisioning the servers are saved because the P2P network is self-scaling. Hence, as the P2P technology matures also business models are developed for it. [LL07] describes the integration of P2P technologies into an IP Multimedia Subsystem (IMS). P2P concepts are used for efficient and scalable content distribution, whereas the IMS is used for Authentication, Authorisation and Ac-

counting (AAA). Further on, [ZLCS08] discusses a digital rights management scheme for BitTorrent-like networks.

2.4 A Pricing Framework

Hardin describes in [Har68] *the tragedy of the commons* with an example of grassland and herdsmen: The grassland is owned by the community, therefore called commons, and is open to all. Assume herdsmen behave rationally and are interested in maximising their own benefit. Hence, each herdsman determines his utility when increasing the number of his animals. The profit for the animals is fully realised by the herdsman whereas the cost for overgrazing the grassland is shared by the community. Thus, a rational herdsman will increase the number of his animals until the grassland is completely grazed.

These kinds of problems arise when no incentive for individuals exist when they consume freely available resources. And an appropriate remedy for these problems is to charge for the use of resources. As argued by [MMV94] also the Internet faces the tragedy of the commons, because end-users are charged prices which are independent of the use of resources in the network (e.g. a flat-rate charge). Hence, users have no incentive for appropriate congestion control. Further on, as a possible solution the authors of [MMV94] propose a smart market, where a charge per packet has to be paid by the user in the case of congestion.

These approaches are called resource or congestion pricing. A mathematical framework for congestion pricing was developed by F. Kelly et al. in [Kel97, KMT98]. The strengths of this framework are twofold:

- Optimisation: It models the resource allocation as global optimisation problem. The objective is to maximise the sum of the utility of all users. The utility depends on the rate of the user and by defining appropriate utility functions different fairness criteria between the users can be modelled. The optimisation problem is constrained by the finite resources of the network.
- Decomposition: To realise a decentralised control the global optimisation problem is split into sub-problems. These sub-problems find the global optimum with locally available information and minimum control information.

A decentralised control is inevitable because the algorithm has to scale for large networks like the Internet. Hence, in the proposed distributed algorithm in [KMT98] users specify

their willingness-to-pay and set their sending rates accordingly. However, they are charged for the congestion they cause in the network. This charge is computed by the routers and communicated as price to the end-points.

Although monetary charges are discussed in the literature as well [KMBL99, GK99a], Kelly's approach became more important as mathematical framework in general and is extended beyond congestion control. Therefore, it is often denoted as Network Utility Maximisation (NUM) problem (see e.g. the survey in [CLCD07]).

It is used for reverse engineering TCP algorithms designed at packet-level for a better understanding at flow-level [Key01, LPD02]. Packet loss or delay can be interpreted as congestion price and the algorithms only differ in their utility function. Additionally, new TCP algorithms are developed based on this mathematical framework, e.g. FAST TCP [JWL04].

The optimisation problem for rate control is fairly tractable. In general, a concave utility function is assumed and capacity constraints are linear. Constraints are internalised with Lagrange dual variables (see Chapter 3 for details). Hence, the resulting problem has a unique optimum and the local optimum is also the global one. Furthermore, optimising the primal or dual variables results in the same optimum, i.e. the duality gap is zero. Hence, most of the literature for rate control discusses open issues for the decomposition of the problem. For example, the convergence and stability of the distributed algorithm [Vin02, Mas02] or single-bit price information [GK99b, AL00, ZK02].

The decomposition of the optimisation problem over different geographically disparate network elements is denoted as horizontal decomposition [CLCD07]. When the optimisation problem is split over multiple network layers, it is called a vertical decomposition. One example is the extension of the rate control problem to routing, which is discussed also in this work. In the case of multi-path routing the problem is not unique with respect to the rates on the routes but only with respect to the total rates of a user. This makes the mathematical framework more complicated, but distributed algorithms exist which solve the global optimisation problem. For single-path routing the optimisation problem becomes an integer problem. However, integer constraints may introduce a duality gap (see Chapter 4). Hence, other decomposition techniques may be preferable, e.g. Generalised Bender's Decomposition described in [EMW06]. Alternatively, the formulation of the optimisation might be modified in a sophisticated way to a problem that is easier to solve (see [Chi07] for different examples).

Another limitation is the restriction to concave utility functions. Although concave functions model elastic traffic they are inappropriate for inelastic traffic with specific de-

mands [She95]. This limitation is addressed in [LMS05, FC05] where other utility functions are discussed. However, also non-concave utility functions can introduce a duality gap, which causes oscillations and instability. In [LMS05] these oscillations are avoided where some users with sigmoidal utility functions are self-regulating, i.e. they stop their transmission under specific conditions.

In the context of wireless networks the optimisation problem is adapted to the properties of the wireless medium. Here, the capacity is time-varying, the wireless interface is a multi-access medium and users interfere with each other. Hence, cross-layer optimisation problems are studied which discuss the modulation, coding and power control at the physical layer, the scheduling at the medium access layer, the routing at the network layer and the rate control at the transport layer. Two surveys [LSS06, CLCD07] summarise the recent progress in this field.

In conclusion, resource pricing or network utility maximisation proved to be effective for convex optimisation problems, where distributed algorithms drive the network to a global optimum. The framework encounters problems when the optimisation problem becomes non-convex, e.g. due to integer constraints or non-concave utility functions. These problems are an ongoing research effort. In this work we discuss convex optimisation problems only. The approach is applied to combined rate control and routing in IP networks and for the resource allocation in P2P networks.

Chapter 3

Resource Pricing

This thesis discusses several resource allocation problems in communication networks. The models are based on the resource pricing (or congestion pricing) approach [Kel97, KMT98]. This approach is widely used to derive algorithms for rate control at the transport layer. Here, a flow consumes bandwidth at each hop along its route to the sink. But the routers charge a price for the usage of their resources, which is summed up to a path price for each flow and controls the future sending rates. By defining appropriate pricing mechanisms fair and efficient rate control can be assured.

In the original work [KMT98] the routing is assumed to be fixed and an optimal solution is only found on the transport layer and not on both, the network and the transport layer. We extend the original model and study the cross-layer optimisation for the transport and network layer in Chapter 4. The second application scenario of resource pricing in this thesis discusses overlay networks, which are formed by P2P applications. This approach is presented in Chapter 5. Based on the limitations of combined rate control and routing at lower layers, we propose an alternative approach that exploits routing in the overlay. This is discussed in Chapter 6.

All three approaches are based on the same global optimisation problem. Only the separation into sub problems to derive distributed algorithms is different. This global optimisation problem is discussed in this chapter.

3.1 Network Model

The allocation of bandwidth in communication networks is modelled as global optimisation problem. Assume the network consists of the set of links $\mathcal{L}$, whereby a link $l \in \mathcal{L}$

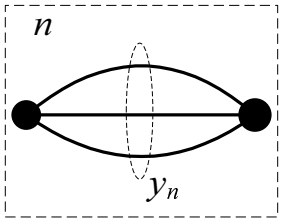


Figure 3.1: Multi-path routing

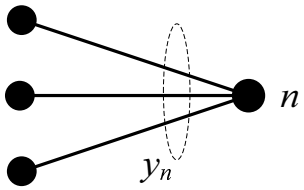


Figure 3.2: Multi-source download

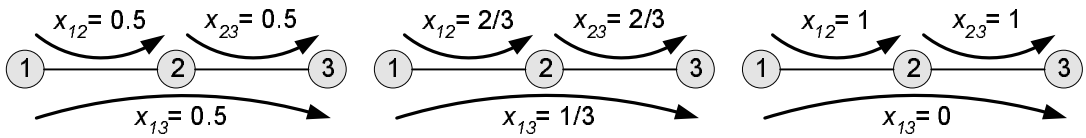
has a capacity of C_l. A set of users $\mathcal{N}$ makes use of the resources in the network. Assume the user $n \in \mathcal{N}$ receives an aggregated rate of y_n and depending on this rate has an utility $U_n(y_n)$. Here, the aggregated rate is the sum over the rates over different routes.

The term user is interpreted differently in the following chapters. In Chapter 4 we associate with a user a source-sink pair. With multi-path routing different routes can be used to connect the source-sink pair. The interpretation of the term user is different to the aforementioned in Chapter 5 and Chapter 6. Here, for P2P networks with multi-source download a user receives data from different senders. However, there is only one route from each sender to the sink. Figure 3.1 and Figure 3.2 illustrate the differences between both models. In this chapter we use the terminology for multi-path routing. The presented results are also valid for P2P networks.

We introduce the set of routes $\mathcal{R}$. A subset of $\mathcal{R}$ is $\mathcal{R}(n)$, which consists of all possible routes for the source-sink pair associated with user n. Furthermore, the rate on a route $r \in \mathcal{R}$ is x_r. According to [Kel97] the resource allocation is modelled as optimisation problem, where the total utility over all users is maximised. The problem is constrained by the capacity at the links, i.e. the aggregated rate of the data flows which use a link has to be less or equal to the capacity of this link. Hereby, the set $\mathcal{R}(l)$ includes all routes that use link l. Hence, the optimisation problem is given by

$$\begin{aligned}
&\text{SYSTEM}: \\
&\qquad \text{maximise} \quad \sum_{n\in\mathcal{N}} U_n(y_n) && (3.1)\\
&\qquad \text{subject to} \quad \sum_{r\in\mathcal{R}(n)} x_r = y_n, \quad \forall n \in \mathcal{N} && (3.2)\\
&\qquad\qquad\qquad \sum_{r\in\mathcal{R}(l)} x_r \leq C_l, \quad \forall l \in \mathcal{L} && (3.3)\\
&\qquad\qquad \text{over} \quad x_r \geq 0. && (3.4)
\end{aligned}$$

A large body of research [KMT98, LL99, GK99b, Key01, LPD02, Zim05] concentrates



(a) Max-min fair (b) Proportional fair (c) Max. aggr. throughput

Figure 3.3: Example for fairness criteria

on the transport layer and assumes the routing is fixed. Hence, the flow constraints in (3.2) simplify to a single route and the objective of the optimisation problem in (3.1) depends on a rate over a single route for each user.

Often a concave utility function is used in the literature. This implies a diminishing marginal utility [PR05]: As more and more bandwidth is allocated to a flow, allocating additional amounts will yield smaller and smaller additions to utility. Such utility functions are appropriate for users with elastic traffic demands [She95]. With a concave objective function in (3.1) and linear constraints in (3.2) and (3.3) the optimisation problem belongs to the class of convex* optimisation problems. Furthermore, the optimum is unique with respect to y_n and a local optimum coincides with the global optimum. However, the rates x_r are not necessarily unique at the optimum, because different rate allocations may sum up to the same total download rate y_n. Thus, many possible rate allocations may exist with respect to x_r.

3.2 Fairness

To ensure some kind of fairness with respect to the allocation of resources in a network the utility function in (3.1) is set adequately. In general, different fairness criteria deviate from each other, because they trade-off between fairness and efficiency differently. For example, consider two concatenated links with equal capacity as depicted in Figure 3.3. One source uses the first link, another source uses the second link and the third source uses both links to reach a sink. One understanding of fairness (called *max-min fairness*) is to allocate the resources equally to the three sources (see Figure 3.3(a) for a link capacity equal to one). However, this does not maximise the aggregated rate over all sources, which is 1.5 in this example. The aggregated rate is maximised when the capacity of the links is allocated to the sources, which use only one link (see Figure 3.3(c)). Here, the aggregated throughput is 2. However, the rate of the third user is zero and the allocation seems to be

*A function f is convex if $f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y)$ with $0 \leq \theta \leq 1$. A function f is concave if $-f$ is convex [BV06].

unfair. This illustrates that a fairness criterion can not be discussed separately, but only with its effect on efficiency.

The concept of *proportional fairness* [KMT98] makes a widely accepted compromise between fairness and efficiency and is used frequently in congestion pricing algorithms. For the discussed example an allocation is proportional fair when the source-sink pairs, which use one link, get twice the resources as the flow over both links (see Figure 3.3(b)). In this case the flow of each source-sink pair consumes the same amount of accumulated resources, which is the sum of the used resources over all links of each route.

The concept can be extended to *weighted proportional fairness*. Here, a user n has a willingness-to-pay W_n. Mathematically, a rate allocation $y = (y_n, n \in \mathcal{N})$ is weighted proportional fair if it is feasible (i.e. not violating the constraints (3.2)-(3.4) of the optimisation problem) and if for any other feasible rate allocation y' the aggregate of weighted proportional changes is non-positive [KMT98]:

$$\sum_{n \in \mathcal{N}} W_n \frac{y'_n - y_n}{y_n} \leq 0. \tag{3.5}$$

In contrast to max-min fairness, which gives absolute priority to small flows, proportional fairness depends on the proportional changes of two rate allocations. Hence, smaller flows are treated less favourably and a proportional fair allocation lies between max-min fairness and maximum aggregated throughput. Furthermore, the willingness-to-pay weighs the value of a user. For integral weights an illustration is given below. By setting the utility function in (3.1) to

$$U_n(y_n) = W_n \ln(y_n) \tag{3.6}$$

weighted proportional fairness is realised [KMT98]. It is also denoted as proportionally fair rates per unit charge. Since the objective in (3.1) is to maximise the aggregated utility of the total rate, the utility function in (3.6) ensures proportional fairness with respect to y_n. This realises fairness with respect to users, because the rate allocation to users is independent of their number of routes in the network. This is in most cases different from flow rate fairness, where the rates of flows on the same route are fair to each other. Hence, the presented model addresses the objections in [Bri07] against flow fairness.

Proportional fairness can be equivalent to max-min fairness. An example is presented in Section 3.4, where each route traverses one bottleneck link only. A rate allocation is called max-min fair if it is feasible and, for each $n \in \mathcal{N}$, y_n cannot be increased without decreasing the rate $y_{n'}$ of a user n' with $y_{n'} \leq y_n$ [BG87]. This concept maximises the rate of the source with the minimal rate allocation. Also max-min fairness can be associated

with a specific utility function, which is described in [Kel97] in detail.

To evaluate the fairness in large, multi-user systems Jain's Fairness Index [Jai91] can be used. In our context, for a vector of aggregated rates $y = (y_n, n \in \mathcal{N})$ the fairness index is computed with

$$f(y) = \frac{(\sum_{n\in\mathcal{N}} y_n)^2}{N \sum_{n\in\mathcal{N}} y_n^2}, \tag{3.7}$$

whereby N denotes the cardinal number of the set $\mathcal{N}$. This metric lies between 0 and 1 and higher values indicate better fairness. For example, if all peers receive equal service rates, the index is 1. Hence, Jain's Fairness Index follows the concept of max-min fairness. Furthermore, Jain's Fairness Index is $1/N$ if all resources are allocated to one user only and the other $(N-1)$ users receive nothing at all.

We extend Jain's Fairness Index to the case of weighted fairness. Therefore, we repeat the illustration of weighted fairness in [KMT98]: When all weights are integral each user $n \in \mathcal{N}$ can be replaced by W_n identical sub-users. Assume a proportional fair allocation over the resulting $\sum_{n\in\mathcal{N}} W_n$ sub-users. When each user n receives the aggregated rate of its W_n sub-users, then this allocation is weighted proportional fair.

Based on this explanation we deduce from (3.7) the Weighted Fairness Index

$$\begin{aligned} f_W(y, W) &= \frac{\left(\sum_{n\in\mathcal{N}} W_n \frac{y_n}{W_n}\right)^2}{\sum_{n\in\mathcal{N}} W_n \cdot \sum_{n\in\mathcal{N}} W_n \left(\frac{y_n}{W_n}\right)^2} \\ &= \frac{(\sum_{n\in\mathcal{N}} y_n)^2}{\sum_{n\in\mathcal{N}} W_n \cdot \sum_{n\in\mathcal{N}} \frac{y_n^2}{W_n}}, \end{aligned} \tag{3.8}$$

whereby $W = (W_n, n \in \mathcal{N})$ is a vector of the willingness-to-pay variables of all users. Also the Weighted Fairness Index lies between 0 and 1 and has similar properties as Jain's Fairness Index. It is used to evaluate the fairness between users in simulations in Chapter 5.

3.3 Duality

The optimisation problem in (3.1)-(3.4) is denoted generally as primal problem and the vectors $x = (x_r, r \in \mathcal{R})$ and $y = (y_n, n \in \mathcal{N})$ as *primal variables*. To find the optimum of

the constrained problem *Lagrange multipliers* (or *dual variables*) are used (see e.g. Chapter 4 in [BV06] or Chapter 10.3 in [Lue03]). Hence, a Lagrange function (or *Lagrangian*) is defined by the objective and the constraints of the optimisation problem. Constraints are internalised by weighted sums. For the optimisation problem in (3.1)-(3.4) the Lagrangian is

$$L(x,y,\lambda,\mu,m) = \sum_{n\in\mathcal{N}}\left(U_n(y_n) - \lambda_n\left(y_n - \sum_{r\in\mathcal{R}(n)} x_r\right)\right) + \sum_{l\in\mathcal{L}}\mu_l\left(C_l - \sum_{r\in\mathcal{R}(l)} x_r - m_l\right), \tag{3.9}$$

where $\lambda = (\lambda_n, n\in\mathcal{N})$ and $\mu = (\mu_l, l\in\mathcal{L})$ are vectors of Lagrange multipliers (or dual variables) and $m = (m_l \geq 0, l\in\mathcal{L})$ is a vector of slack variables. *Slack variables* are used to re-write an inequality constraint as an equality constraint. For the capacity constraint in (3.3) we write $\sum_{r\in\mathcal{R}(l)} x_r + m_l = C_l$. Hence, m_l has to be non-negative and can be interpreted as the spare capacity of server l.

Lagrange functions are frequently used in economic theory. Here, the behaviour of different market participants is modelled as a constrained optimisation problem [PR05]. An example is a consumer who maximises his utility subject to his limited budget. The utility depends on the quantities of different goods the consumer purchases at specific prices. In this context, Lagrange multipliers are also denoted as shadow prices, because they express the price for an additional unit of the limited resource [BV06]. For a better illustration of the proposed algorithms in this work we will use frequently economical interpretations of the resource allocation problem and its sub-problems. Like in [KMT98] Lagrange multipliers are interpreted as prices for the usage of resources. Hence, for the Lagrange multipliers in (3.9) we say λ_n is the price per unit bandwidth offered by the user n and μ_l is the price per unit bandwidth charged by the link l. The idea of prices is elaborated further when distributed algorithms are derived, e.g. in Section 4.1 for combined rate control and routing.

For the case of static single-path routing discussed in [KMT98] and the logarithmic utility function in (3.6) the Lagrangian simplifies to

$$L_{\mathrm{SPR}}(x,\mu,m) = \sum_{n\in\mathcal{N}} W_n \ln x_{r(n)} + \sum_{l\in\mathcal{L}}\mu_l\left(C_l - \sum_{r\in\mathcal{R}(l)} x_r - m_l\right), \tag{3.10}$$

where $r(n)$ denotes the route of user n. Here, the total rate is equal to the rate of the single-path route. Thus, the second summand in (3.9) is equal to zero.

The function L_{SPR} in (3.10) is a concave function for $x_{r(n)}$ and has a unique maximum with respect to it. Hence, the local maximum is also the global one. Furthermore, the identity $\sum_{l\in\mathcal{L}} \mu_l \sum_{r\in\mathcal{R}(l)} x_r = \sum_{n\in\mathcal{N}} x_{r(n)} \sum_{l\in\mathcal{L}(r(n))} \mu_l$ holds for single-path routing. Thus, with

$$\frac{\partial L_{\text{SPR}}(x,\mu)}{\partial x_{r(n)}} = \frac{W_n}{x_{r(n)}} - \sum_{l\in\mathcal{L}(r(n))} \mu_l \tag{3.11}$$

the dual function [BV06] is determined by

$$g(\mu) = \sup_x L_{\text{SPR}}(x,\mu) \tag{3.12}$$

$$= \sum_{n\in\mathcal{N}} \left(W_n \ln \frac{W_n}{\sum_{l\in\mathcal{L}(r(n))} \mu_l} - W_n \right) + \sum_{l\in\mathcal{L}} \mu_l C_l, \tag{3.13}$$

where the set $\mathcal{L}(r(n))$ denotes the links used by the route of user n. As outlined in the next section m_l is zero if μ_l is greater zero. Hence, it is omitted in the dual function. Furthermore, by omitting also the constant terms the dual optimisation problem[†] is

DUAL :

$$\text{minimise} \quad \sum_{l\in\mathcal{L}} \mu_l C_l - \sum_{n\in\mathcal{N}} W_n \ln \sum_{l\in\mathcal{L}(r(n))} \mu_l \tag{3.14}$$

$$\text{over} \quad \mu_l \geq 0. \tag{3.15}$$

Here, μ_l is non-negative because this dual variable is associated with an inequality constraint. A main result of dual theory is *weak duality*. Here, for any primal feasible vector x and any dual feasible vector μ, it holds

$$\sum_{n\in\mathcal{N}} W_n \ln x_{r(n)} \leq g(\mu). \tag{3.16}$$

Hence, the dual function is an upper bound for a feasible value of the objective of the primal maximisation problem. Furthermore, this inequality holds also for the optimal values x^* and μ^*. Hence,

$$\sum_{n\in\mathcal{N}} W_n \ln x^*_{r(n)} \leq g(\mu^*). \tag{3.17}$$

[†]Like in [KMT98] the same dual problem can also be stated as maximisation problem of the negative objective.

The difference of the inequalities in (3.16) and (3.17) is called the *duality gap* and optimal duality gap, respectively. When the optimal duality gap is zero, *strong duality* holds. Hence, the computation of the optimum for the primal problem or for the dual problem results in the same solution. For convex optimisation problems with linear constraints and a feasible solution strong duality holds [BV06]. This is also the case for the general optimisation problem in (3.1)-(3.4) with a concave utility function. The linear constraints define a convex feasible region. Additionally, the objective is chosen to be concave. Hence, a local optimum is also the global one. Although the dual problem can not be stated explicitly for (3.1)-(3.4) necessary and sufficient conditions for the optimality are derived in [Kel97]:

Assume $(y^*, x^*, \lambda^*, \mu^*, m^*)$ is primal and dual optimal. Hence,

$$\sum_{r \in \mathcal{R}(n)} x_r^* = y_n^*, \qquad \forall n \in \mathcal{N} \tag{3.18}$$

$$\sum_{r \in \mathcal{R}(l)} x_r^* \leq C_l, \qquad \forall l \in \mathcal{L} \tag{3.19}$$

$$x_r^* \geq 0, \qquad \forall r \in \mathcal{R} \tag{3.20}$$

$$\mu_l^* \geq 0, \qquad \forall l \in \mathcal{L} \tag{3.21}$$

$$\mu_l^* \left(C_l - \sum_{r \in \mathcal{R}(l)} x_r^* \right) = 0, \qquad \forall l \in \mathcal{L} \tag{3.22}$$

$$U'(y_n^*) - \lambda_n^* \begin{cases} = 0 & \text{if } y_n^* > 0 \\ \leq 0 & \text{if } y_n^* = 0 \end{cases} \qquad \forall n \in \mathcal{N} \tag{3.23}$$

$$\lambda_n^* - \sum_{l \in \mathcal{L}(r(n))} \mu_l^* \begin{cases} = 0 & \text{if } x_r^* > 0 \\ \leq 0 & \text{if } x_r^* = 0 \end{cases} \qquad \forall n \in \mathcal{N}, \ \forall r \in \mathcal{R}(n) \tag{3.24}$$

$$\mu_l^* \begin{cases} = 0 & \text{if } m_l^* > 0 \\ \geq 0 & \text{if } m_l^* = 0 \end{cases} \qquad \forall l \in \mathcal{L} \tag{3.25}$$

In general, this set of conditions is known as the Karush-Kuhn-Tucker (KKT) condition for optimality, where (3.18), (3.19) and (3.20) represent primal feasibility, (3.21) represents dual feasibility and (3.22) complementary slackness. The conditions (3.23), (3.24) and (3.25) are the derivatives of the Lagrange function in (3.9) for the primal variables y_n and x_r and the slack variable m_l, respectively. They are derived in more detail for a simplified model in the next section.

3.4 Parallel Bottleneck Model

In the following we simplify the optimisation problem in (3.1)-(3.4) to a model, where each route faces a single bottleneck link only. However, multiple bottlenecks may exist in the network and different routes of the same user may traverse different bottleneck links. Hence, we denote the model as parallel bottleneck model. Such a simplification is done frequently in the context of P2P networks, where the access links from the users to the Internet service providers (also denoted as uplink) are assumed to be the bottleneck links in the whole network.

In this section we compute analytically the optimum of the simplified problem under an additional condition of connectivity. The result is used in Chapter 5. For the terminology of P2P networks the same result is presented in [EK07b]. By setting the partial derivatives of (3.9) to zero we obtain the necessary conditions for an optimum of (3.9). We use these results to simplify the Lagrangian and to compute finally the unique optimum of the parallel bottleneck model. From

$$\frac{\partial L(x,y,\lambda,\mu,m)}{\partial m_l} = -\mu_l = 0. \tag{3.26}$$

we deduce that μ_l is zero, if $m_l > 0$. This means the capacity constraint at the link l is inactive and hence can be ignored. On the other hand, if $m_l = 0$ the constraint is active. This result is identical to the condition in (3.25). In the following we consider only links, where the capacity constraint is active and denote the set for these bottleneck links as $\mathcal{L}_\mathcal{B}$. Hence, we set $m_l = 0, \forall l \in \mathcal{L}_\mathcal{B}$ in the following.

Inserting the chosen utility function (3.6) into (3.9) we obtain the partial derivative with respect to y_n

$$\begin{aligned} \frac{\partial L(x,y,\lambda,\mu)}{\partial y_n} &= \frac{W_n}{y_n} - \lambda_n = 0 \\ \Rightarrow y_n &= \frac{W_n}{\lambda_n}. \end{aligned} \tag{3.27}$$

Equation (3.27) holds if $y_n > 0$. Hence, for the chosen utility function the result is identical to (3.23). Inserting (3.27) into (3.9) we get a simplified Lagrange function

$$\tilde{L}(x,\lambda,\mu) = \sum_{n\in\mathcal{N}} \left(W_n \ln\left(\frac{W_n}{\lambda_n}\right) - W_n + \lambda_n \sum_{r\in\mathcal{R}(n)} x_r \right) + \sum_{l\in\mathcal{L}_\mathcal{B}} \mu_l \left(C_l - \sum_{r\in\mathcal{R}(l)} x_r \right) \tag{3.28}$$

and its partial derivative with respect to λ_n is

$$\frac{\partial \tilde{L}(x,\lambda,\mu)}{\partial \lambda_n} = -\frac{W_n}{\lambda_n} + \sum_{r\in\mathcal{R}(n)} x_r = 0$$
$$\Rightarrow \lambda_n = \frac{W_n}{\sum_{r\in\mathcal{R}(n)} x_r}. \tag{3.29}$$

Inserting (3.29) into (3.28)

$$\grave{L}(x,\mu) = \sum_{n\in\mathcal{N}} W_n \ln\left(\sum_{r\in\mathcal{R}(n)} x_r\right) + \sum_{l\in\mathcal{L}_\mathcal{B}} \mu_l \left(C_l - \sum_{r\in\mathcal{R}(l)} x_r\right) \tag{3.30}$$

and it follows

$$\frac{\partial \grave{L}(x,\mu)}{\partial x_r} = \frac{W_n}{\sum_{r\in\mathcal{R}(n)} x_r} - \mu_l = 0$$
$$\Rightarrow \sum_{r\in\mathcal{R}(n)} x_r = \frac{W_n}{\mu_l}. \tag{3.31}$$

It can be seen from (3.31) that μ_l depends on the total rate of a user. Therefore, at the optimum $\mu_{l_1} = \mu_{l_2}$ holds for two links $l_1, l_2 \in \mathcal{L}_\mathcal{B}$ if both are a link on routes of the same user n with a positive flow. Hence, $\mu_{l_1} = \mu_{l_2}$ if $l_1 \in \mathcal{L}_\mathcal{B}(r_1)$ and $l_2 \in \mathcal{L}_\mathcal{B}(r_2)$ with $r_1, r_2 \in \mathcal{R}(n)$ and $x_{r_i} > 0, \ \forall i = 1,2$.

A bipartite graph can be composed of the two sets $\mathcal{N}$ and $\mathcal{L}_\mathcal{B}$, where an edge denotes a route with a non-zero rate allocation. If the bipartite graph is connected, $\mu_l = \mu, \forall l \in \mathcal{L}_\mathcal{B}$ holds. (If the bipartite graph is not connected, an optimisation can be run for every disjoint connected sub-graph, separately.) Hence,

$$\frac{\partial \grave{L}(x,\mu)}{\partial \mu} = \frac{\partial}{\partial \mu}\left[\sum_{n\in\mathcal{N}} W_n \ln\left(\sum_{r\in\mathcal{R}(n)} x_r\right) + \mu \sum_{l\in\mathcal{L}_\mathcal{B}} C_l - \mu \sum_{n\in\mathcal{N}} \sum_{r\in\mathcal{R}(n)} x_r\right], \tag{3.32}$$

where the identity $\sum_{l\in\mathcal{L}_\mathcal{B}} \sum_{r\in\mathcal{R}(l)} x_r = \sum_{n\in\mathcal{N}} \sum_{r\in\mathcal{R}(n)} x_r$ is used.

Inserting (3.31) into (3.32)

$$\frac{\partial \grave{L}(\mu)}{\partial \mu} = \frac{\partial}{\partial \mu}\left[\sum_{n\in\mathcal{N}} \left(W_n \ln\left(\frac{W_n}{\mu}\right) - W_n\right) + \mu \sum_{l\in\mathcal{L}_\mathcal{B}} C_l\right] \tag{3.33}$$
$$= -\frac{\sum_{n\in\mathcal{N}} W_n}{\mu} + \sum_{l\in\mathcal{L}_\mathcal{B}} C_l = 0$$

$$\Rightarrow \mu = \frac{\sum_{n\in\mathcal{N}} W_n}{\sum_{l\in\mathcal{L}_\mathcal{B}} C_l}. \tag{3.34}$$

With (3.27), (3.29), (3.31) and (3.34)

$$y_n = \sum_{r\in\mathcal{R}(n)} x_r = \frac{W_n \sum_{l\in\mathcal{L}_\mathcal{B}} C_l}{\sum_{m\in\mathcal{N}} W_m} \tag{3.35}$$

and with (3.27) and (3.35)

$$\lambda_n = \lambda = \mu, \quad \forall n \in \mathcal{N}. \tag{3.36}$$

The total rate of a user in (3.35) depends on the ratio of the total capacity of the bottleneck links to the total willingness-to-pay weighted by the willingness-to-pay of the user. Hence, the optimal resource allocation is weighted proportional fair. For $W_n = W, \ \forall n \in \mathcal{N}$ the equation (3.35) is also max-min fair.

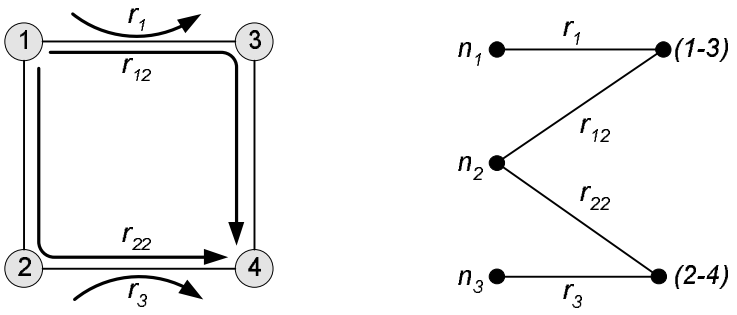
Based on the interpretation that Lagrange multipliers are prices the result in (3.36) represents market equilibrium, where offered and charged prices are equal. Hence, the resource allocation model represents a market with perfect competition.

3.4.1 Example

In this section we study a simple example and illustrate an exception for which the bipartite graph is not connected. The network of interest and the corresponding bipartite graph are given in Figure 3.4. Three users compete for resources. The users n_1 and n_3 use the single route $r_1 = (1-3)$ and $r_3 = (2-4)$, respectively. Additionally, n_2 uses two routes $r_{12} = (1-3-4)$ and $r_{22} = (1-2-4)$. Thus, assuming the links have equal capacity two bottlenecks arise: the links (1-3) and (2-4). However, all routes traverse only one of the bottleneck links. With $C = 1$ for all links and $W = 1$ for all sources the optimal rate allocation is

$$x^* = \begin{pmatrix} x^*_{r_1} & x^*_{r_{12}} & 0 \\ 0 & x^*_{r_{22}} & x^*_{r_3} \end{pmatrix} = \begin{pmatrix} 2/3 & 1/3 & 0 \\ 0 & 1/3 & 2/3 \end{pmatrix}. \tag{3.37}$$

Each source in the network gets the same total rate $y^* = 2/3$, which is the sum of each column of (3.37). Consequently, the rate for a source is independent of the number of flows to the sink. For this simple example, (3.37) is also the unique solution with respect to x_r. However, this changes when users n_1 and n_2 use also the three-hop routes $r_{1b} = (1-2-4-3)$ and $r_{2b} = (2-1-3-4)$. Besides (3.37) also the allocation where each



(a) Network (b) Bipartite graph

Figure 3.4: Example

route consumes a rate of $x = 1/3$ on the bottleneck links ensures a total rate of $y^* = 2/3$. Thus, there is not always a unique optimum with respect to x_r.

The influence of the willingness-to-pay can be seen for the example in Figure 3.4 for $W = [1,3,1]$, where the optimal rates are

$$x^* = \begin{pmatrix} 0.4 & 0.6 & 0 \\ 0 & 0.6 & 0.4 \end{pmatrix}. \tag{3.38}$$

The rate allocation in (3.38) is weighted proportional fair, because user n_2 gets three times the total rate (sum of the second column) than user n_1 and n_3, which is consistent with the ratios of their willingness-to-pay variables.

Assume the capacity of one of the bottleneck links is larger than twice the capacity of the other one, e.g. for the network in Figure 3.4(a) $C_{13} = 0.49$ for link $(1-3)$ and $C = 1$ for all other links. With $W = 1$ for all users the optimal rate allocation is

$$x^* = \begin{pmatrix} 0.49 & 0 & 0 \\ 0 & 0.5 & 0.5 \end{pmatrix}. \tag{3.39}$$

The bipartite graph is not connected because $x_{r_{12}}$ is zero. Thus, the total service rate in (3.39) deviates from (3.35), because a fair allocation is infeasible.

Chapter 4

Combined Rate Control and Routing

In this chapter the resource pricing approach is used not only for rate control at the transport layer but also for the routing decisions at the network layer. Two different concepts are investigated: multi-path and single-path routing.

For multi-path routing we assume that routing is separated into structural and dynamic information as proposed in [ZGC03]. The structural information informs about the topology of the network whereas the dynamic information indicates about the actual quality of the link, e.g. about the link utilisation. Thus, for multi-path routing we assume a user knows about different routes to the destination. The resource pricing algorithm enables to capture the congestion state at the routers, the signalling of the congestion state to the end-users by using prices and rate control of the end-users over multiple routes. Similar work is discussed in [KST01, WPL03, KV05, Pag06]. Primarily, our work is distinguishable from them by the extension of the algorithm to ensure fairness per user, by a detailed description of the packet-level implementation and experimental verification of the approach.

Since predominantly single-path routing is used in today's Internet, we investigate the compatibility of it with the proposed pricing approach. We use the link state protocol for routing. Similar to other studies for dynamic routing [KZ89, WLLD05] oscillations can occur. Therefore, also equal-cost multi-path is discussed. Additionally, the problem with oscillations is interpreted with economic theory. Main parts of this chapter are presented in [EK06b].

4.1 Distributed Algorithm

In the following we divide the global optimisation problem in (3.1)-(3.4) into sub-problems to develop a scalable, distributed algorithm, where each entity works with limited information about the network conditions and users in the network. We rearrange the Lagrangian in (3.9) to

$$L(x,y,\lambda,\mu,m) = \sum_{n\in\mathcal{N}} \left(U_n(y_n) - \lambda_n y_n\right) + \sum_{r\in\mathcal{R}} x_r \left(\lambda_{n(r)} - \sum_{l\in\mathcal{L}(r)} \mu_l\right) + \sum_{l\in\mathcal{L}} \mu_l (C_l - m_l), \tag{4.1}$$

where $n(r)$ denotes the user of route r. By looking at the Lagrangian in (4.1) we see that the global optimisation problem is separable into sub-problems for the user, the routes and the links. Furthermore, maximising the total of a sum is equivalent to maximising each summand. Hence, we can decompose the Lagrangian into the sub-problems

USER n :

$$\text{maximise} \quad U_n(y_n) - \lambda_n y_n \tag{4.2}$$
$$\text{over} \quad y_n \geq 0 \tag{4.3}$$

ROUTE r :

$$\text{maximise} \quad x_r \left(\lambda_{n(r)} - \sum_{l\in\mathcal{L}(r)} \mu_l\right) \tag{4.4}$$
$$\text{over} \quad x_r \geq 0 \tag{4.5}$$

LINK l :

$$\text{maximise} \quad \mu_l (C_l - m_l) \tag{4.6}$$
$$\text{over} \quad \mu_l \geq 0. \tag{4.7}$$

The economical interpretation of the sub-problems is as follows. A user is selfish and tries to maximise its own utility, which depends on the rate y_n. However, the user has to pay a price for using bandwidth. Since λ_n is a price per unit bandwidth, the product $\lambda_n y_n$ in (4.2) reflects the total price that user n is willing to pay.

The route problem in (4.4) determines the sending rate. Here, the sending rate x_r on route r depends on the price $\lambda_{n(r)}$ offered by the corresponding user n and the charged prices μ_l by the links of that route. Furthermore, the links maximise their revenue. Since the slack

variable m_l in (4.6) can be interpreted as the spare capacity on the link, subtracting m_l from the capacity C_l reflects the total rate of forwarded traffic of this link. By multiplying it with the price per unit bandwidth charged by the link we obtain the revenue of that link. Hence, the derived sub-problems are similar to standard problems in economics. The user problem is denoted as utility maximisation of a consumer and the route problem as well as the link problem is known as profit maximisation of a producer [PR05].

The distributed algorithm works as follows. A link l computes the charged price based on its input rate z_l. The input rate is

$$z_l(t) \leq \sum_{r \in \mathcal{R}(l)} x_r(t) \tag{4.8}$$

and (4.8) holds with equality when the sending rates on the routes using this link are not limited by congestion on previously traversed links. Different price rules developed for single-path routing are also applicable for multi-path routing. For example, we choose price rule PC1 proposed in [AL00]

$$\mu_l(t+1) = \max(0, \mu_l(t) + \kappa(z_l(t) - C_l)), \tag{4.9}$$

whereby κ is a small positive step size. According to (4.9) the price of a link is only greater zero when the link capacity is fully or over utilised. It increases if the load of the link is greater than its capacity and decreases otherwise.

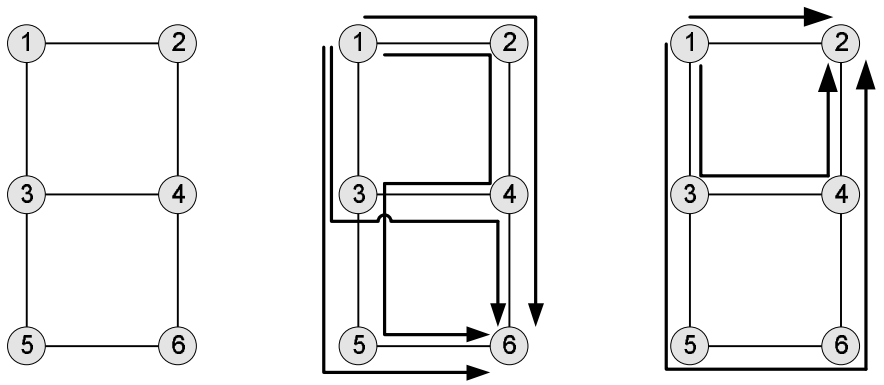
We associate with a user a traffic source, which uses one sending agent for each route between the source-sink pair. The sending agents adapt their rate over route r with

$$x_r(t+1) = \max\left(\varepsilon, x_r(t) + \gamma x_r(t)\left(\lambda_{n(r)}(t) - \sum_{l \in \mathcal{L}(r)} \mu_l(t)\right)\right), \tag{4.10}$$

where γ is a small positive step size and ε is a small positive constant. Thus, the sending rate x_r does not fall below ε. This models a kind of probing, which is necessary to receive information about the link prices of the route. Assume each user has a logarithmic utility function weighted by the willingness-to-pay as in (3.6) then a user n adapts its price offer with

$$\lambda_n(t) = \frac{W_n}{y_n(t)} = \frac{W_n}{\sum_{r \in \mathcal{R}(n)} x_r(t)}. \tag{4.11}$$

Hereby, we make use of the conditions (3.18) and (3.23) from the Karush-Kuhn-Tucker condition for optimality. The rule for the sending rate in (4.10) is a scaled gradient algorithm [BT89] of the route sub-problem in (4.4). Furthermore, it extends the pri-



(a) Network (b) Flows of source 1 (c) Flows of source 2

Figure 4.1: Example

mal algorithm in [KMT98] for multiple routes. If only a single route is available to a user, the multi-path algorithm in (4.10) and (4.11) simplifies to the single-path algorithm in [KMT98].

In general, a distributed algorithm that updates the primal variable is denoted as *primal algorithm* whereas a *dual algorithm* adapts the dual variable [BV06]. In our case the primal variables are the sending rates and the dual variables are the offered prices. Since the sending rate in (4.10) and the link price in (4.9) are computed with difference equations the proposed system is a *primal-dual algorithm*. Furthermore, λ_n is inversely proportional to x_r. By inserting (4.11) into (4.10) we see that in the absence of congestion the rate increases not faster than with a constant term W_n. However, when congestion occurs on a link l of the used route the link price μ_l is positive. Then the sending rate is reduced proportional to the rate which caused congestion. Hence, (4.10) realises the additive increase/multiplicative decrease (AIMD) principle also used in the original TCP version [Jac88] and other variants.

The equilibrium point of (4.10) coincides with the optimum in (3.24). Furthermore, (3.23) can be derived from (4.11) and a stable non-zero link price is achieved when it is fully utilised. Asymptotic stability for a similar primal-dual algorithm is shown in [Voi05].

4.2 Flow-level Simulations

Independent of any protocol implementation we discuss the performance of the proposed algorithm for a simple example. Consider the network of six nodes in Figure 4.1. Additionally, assume two sources at node 1 send traffic to the sinks at node 6 and node 2,

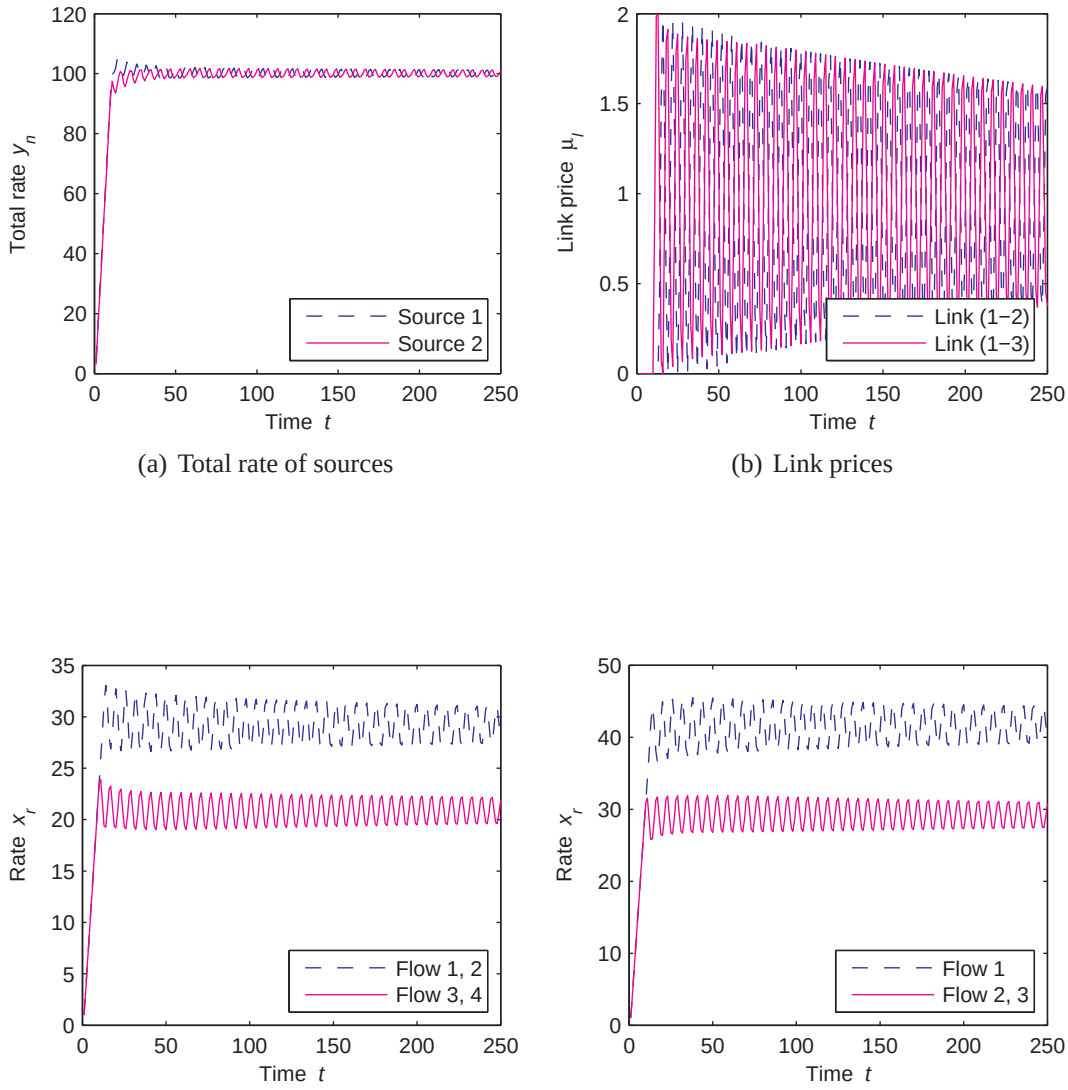
respectively. For both source-sink pairs multiple routes are possible. As depicted in Figure 4.1(b) source 1 can use four different routes. These are the routes (1-2-4-6), (1-2-4-3-5-6), (1-3-4-6) and (1-3-5-6), which are denoted in the following as flows 1 to 4 of source 1, respectively. The routes of source 2 are (1-2), (1-3-4-2) and (1-3-5-6-4-2) (see Figure 4.1(c)) and are denoted as flow 1 to 3 of source 2, respectively.

In the flow-level simulations no delays are taken into account and variables are updated based on rounds. In each round sending rates are adapted with (4.10) and a user computes its price offer with (4.11). Therefore, the sending rates depend on delayed information about the link prices from the last round. Furthermore, link prices are set with (4.9), where the rate information is also delayed by one round. Assume a capacity of $C = 100$ for all links and a willingness-to-pay of $W = 100$ for both sources. The step sizes are set to $\gamma = \kappa = 0.1$ and sending rates are initialised with $x_r = 1$. Figure 4.2 summarises the results. (Rates on different routes overlap with each other. In this case only one curve is shown.)

The total rates in Figure 4.2(a) of both sources increases fast to a fair allocation of resources, which avoids congestion in the network. The increase at the beginning has a slope of γW_n. Since all link prices are zero, this becomes obvious from the formulae by summing up over all sending rates in (4.10) of a user and inserting (4.11). For high-capacity networks a linear increase might be too slow. Here, the proposed algorithm should be extended by the slow-start algorithm [Jac88], which increases the rates of new connections exponentially. Thus, the proposed algorithm is intended to be an alternative to the congestion avoidance algorithms in today's TCP versions.

In this simulation setup the chosen values for the step sizes κ and γ are a good compromise between rate of convergence and instability. Setting both values to 0.2 causes instability because the variables vary considerably, especially the link prices, and fail to find the optimum. On the other hand convergence is slower with e.g. $\gamma = \kappa = 0.01$. Hereby, the overshoot is similar as with 0.1. For general results the stability can be studied analytically with control theory. Stability results for similar distributed algorithms can be found in [Voi05, HSH+03]. Because of the limitations of the approach in general, which are outlined in the following, stability of the proposed algorithm is not studied analytically.

Especially, for the link prices in Figure 4.2(b) and rates of the flows in Figure 4.2(c) and 4.2(d) oscillations occur. These are caused by adapting link prices, sending rates and price offers in each round. Convergence is improved by adapting the price offers not in every round. This ensures that link prices and sending rates face constant price offers over multiple rounds. However, for a constant value of λ_n over an update interval the sending



(a) Total rate of sources

(b) Link prices

(c) Flow rates of source 1

(d) Flow rates of source 2

Figure 4.2: Results for flow-level simulation

rate will increase multiplicatively, which might cause instability. Instead of using λ_n, we introduce v_r, the willingness-to-pay of route r. The variable v_r is updated every T_n time units and the equations (4.10) and (4.11) are replaced by

$$x_r(t+1) = \max\left(\varepsilon, x_r(t) + \gamma\left(v_r(t) - x_r(t)\sum_{l\in\mathcal{L}(r)}\mu_l(t)\right)\right) \tag{4.12}$$

and

$$v_r(t) = \begin{cases} \dfrac{W_n}{y_n(t)}x_r(t) & \text{if } t = aT_n,\ \ a = 0,1,2,\ldots \\ v_r(t-1) & \text{otherwise,} \end{cases} \tag{4.13}$$

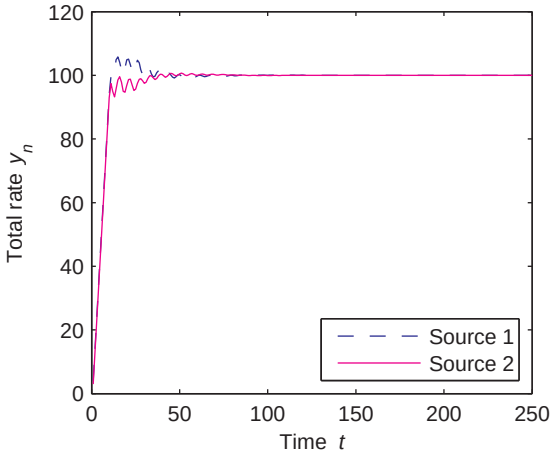
respectively. For $T_n = 1$ the algorithm defined by (4.12) and (4.13) is identical to (4.10) and (4.11). Results for the new algorithm are depicted in Figure 4.3 for the case where the update interval of v_r is $T_n = 25$. With the changed algorithm the oscillations of the quantities in Figure 4.3 disappear fast as compared to the previous results in Figure 4.2.

Also for the proposed distributed algorithm total rates and link prices are unique and coincide with the optimal values of the optimisation problem in (3.1)-(3.4). Since each flow traverses only one bottleneck link the results for the steady-state can be computed with (3.34) and (3.35). Only the rates of the different flows are not unique and as observed in additional simulations for different initialisations different steady-states are reached.

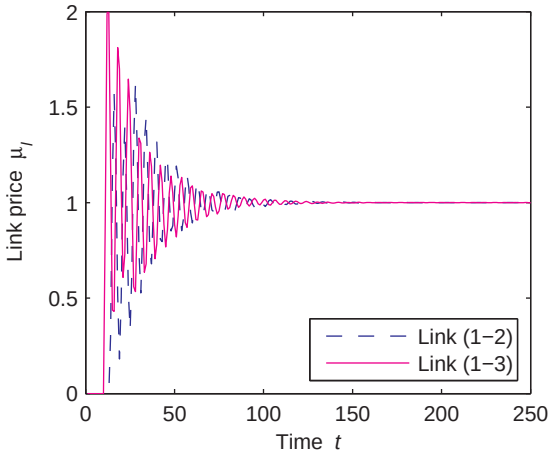
4.3 Packet-level Implementation

The revised distributed algorithm from Section 4.2 is converted to the packet-level. At packet-level the congestion window reflects the sent but yet unacknowledged data and controls how many new packets are injected into the network. Whereas the sending rate at flow-level is adapted per time interval, the congestion window is adapted on receiving acknowledgements. Hence, the control of the congestion window is a self-clocking mechanism. The rule for the sending rate in (4.10) becomes the rule for the congestion window *cwnd*, which is updated by

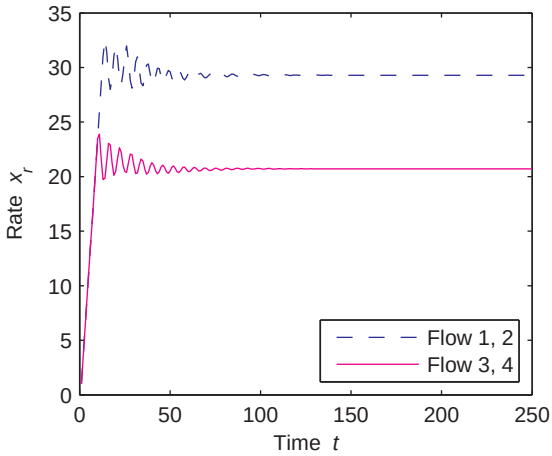
$$\Delta cwnd_r(t) = \kappa_r \frac{RTT_r(t)^2}{cwnd_r(t)}\left(v_r(t) - \sum_{l\in\mathcal{L}(r)}\mu_l(t)\cdot\frac{cwnd_r(t)}{RTT_r(t)}\right) \tag{4.14}$$



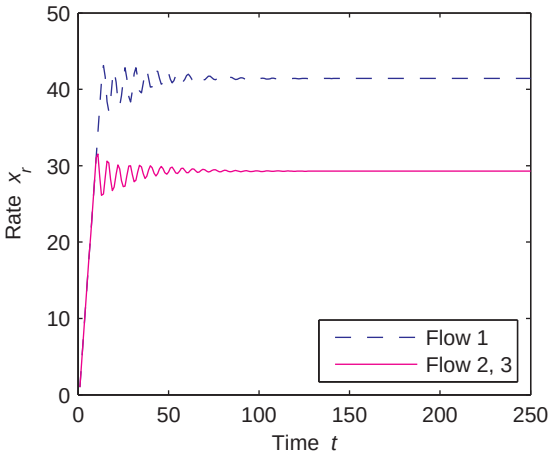
(a) Total rate of sources



(b) Link prices



(c) Flow rates of source 1



(d) Flow rates of source 2

Figure 4.3: Results for flow-level simulation with delayed update of v_r

on every acknowledgement. Hereby, RTT_r is the round-trip time of the route r. The conversion is based on three aspects: Firstly, $cwnd = RTT \cdot x$. Secondly, the sending rate in (4.10) is adapted per time interval whereas the congestion window on the reception of an acknowledgement. Thus, per time interval $cwnd/RTT$ acknowledgements are received and (4.14) is scaled accordingly. Thirdly, the conversion is based on the assumption that the round-trip time is constant over small time intervals.

Also the difference of (4.14) to its single-path equivalent in [ZK02, Zim05] is the usage of v_r, the willingness-to-pay per route, instead of W_n, the willingness-to-pay of a user n. The rules for adapting v_r in (4.13) and μ_l in (4.9) depend at packet-level on the estimated values of x_r and z_l, respectively. These are denoted as $\hat{x}_r$ and $\hat{z}_l$ and are measured as exponentially moving averages with

$$\hat{x}_r(t+T_n) = (1-\delta_n)\hat{x}_r(t) + \delta_n \frac{data}{T_n} \tag{4.15}$$

and

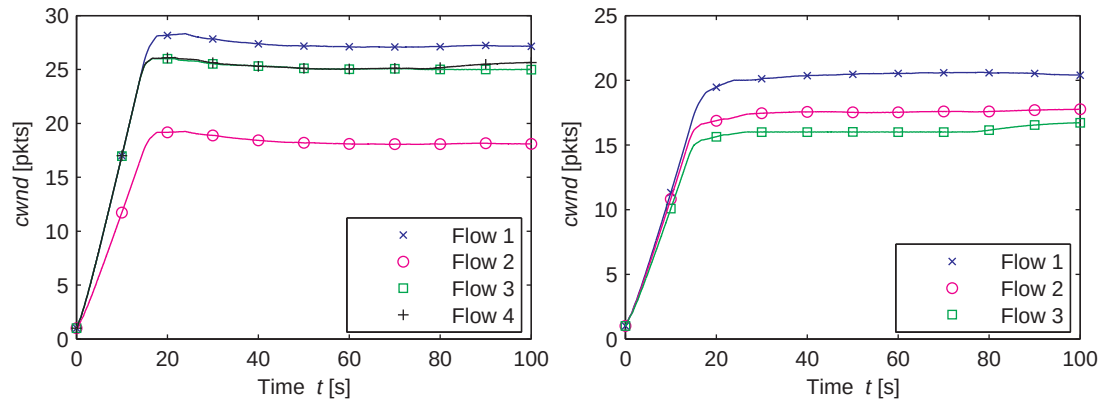
$$\hat{z}_l(t+T_l) = (1-\delta_l)\hat{z}_l(t) + \delta_l \frac{data}{T_l} \tag{4.16}$$

per time interval T_n and T_l, respectively. Here, δ is a smoothing factor and $data$ is the amount of data, which is acknowledged at the sender and observed at the input queue of the link, respectively.

Simulations at packet-level are conducted with the network simulator ns-2 [ns2]. For the example in Figure 4.1 all links have a capacity of $C = 10\,\text{Mbps}$ and propagation delay of $t_\text{d} = 10\,\text{ms}$. The willingness-to-pay of both sources is $W = 1000\,\$/\text{s}$. Link prices and the willingness-to-pay per route are updated every $T_l = 10\,\text{ms}$ and $T_n = 250\,\text{ms}$ with $\delta_l = \delta_n = 0.1$ for all links and routes, respectively. Furthermore, $\gamma = \kappa = 0.1$. Results of the simulation are depicted in Figure 4.4

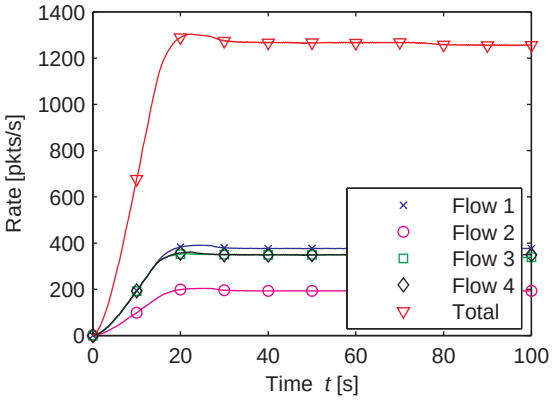
The congestion windows of the two sources converge at around 20 s to a steady-state after a linear increase. Correspondingly, also the rates converge. The total rate is nearly identical for both sources and is greater than 1200 packets/s. With a packet size of 1000 Bytes the total rate with 9.6 Mbps is near to the link capacity. The links (1-2) and (1-3) are the bottlenecks in the network. Their link prices in Figure 4.4(e) oscillate around 0.8 \$/pkts, which corresponds to the optimal link price in (3.34). The queue size of the bottlenecks in Figure 4.4(f) is around 10 packets.

The price rule in (4.9) does not control the queue size. Therefore, the price rule PC3 is introduced in [AL00]. With a target queue size of b_0 and an actual queue size $b_l(t)$ the

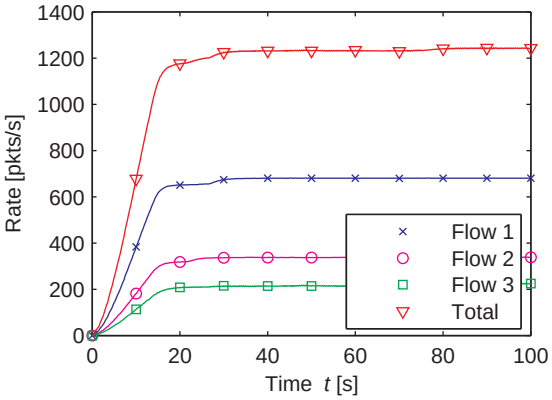


(a) Congestion window of source 1

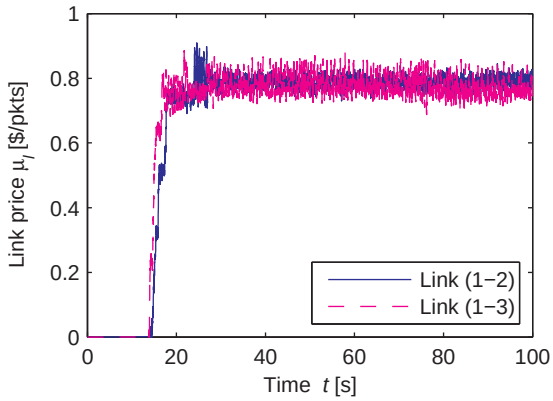
(b) Congestion window of source 2



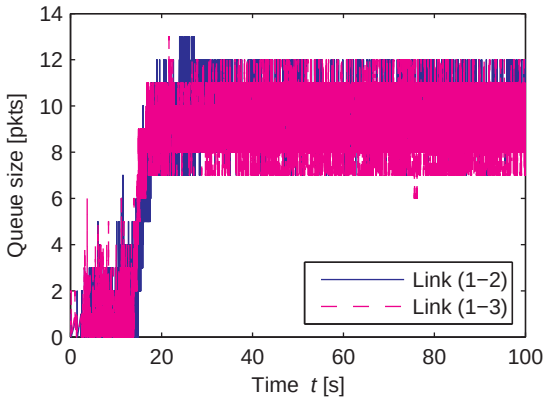
(c) Rates of source 1



(d) Rates of source 2

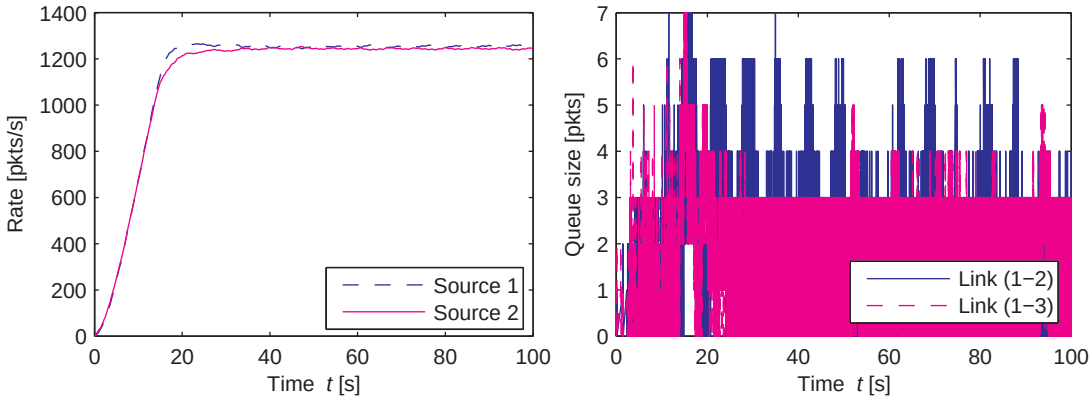


(e) Link price



(f) Queue size

Figure 4.4: Results for packet-level simulation with PC1



(a) Total rate of sources (b) Queue size

Figure 4.5: Results for packet-level simulation with PC3

link price is computed with

$$\mu_l(t+1) = \max\left(0, \mu_l(t) + \kappa\left(\alpha\left(b_l(t) - b_0\right) + z_l(t) - C_l\right)\right), \tag{4.17}$$

where $\alpha > 0$ weighs the influence of the queue size. Simulation results with PC3 (with $b_0 = 2$ and $\alpha = 0.01$) are depicted in Figure 4.5 for the discussed example. As compared to Figure 4.4(f) the queue size in Figure 4.5(b) is reduced and oscillates around the target queue size. The rate allocation in Figure 4.5(a) shows slightly more oscillations as with PC1. However, the rate allocation is fair and efficient. By defining a queue length which is much larger as the target queue size packet losses can be avoided. Furthermore, the queuing delay is also controlled by the target queue size and does not increase without bounds.

With PC1 and PC3 explicit prices are communicated between links and sending agents. This demonstrates the functionality of the proposed approach in general, but does not conform to the IP and TCP standards. However, with [RFB01] the TCP/IP stack is extended by the Explicit Congestion Notification (ECN). Thus, single bit information about congestion can be conveyed in the network. A large body of research (e.g. [GK99b, AL00, ZK02]) studies marking strategies to improve the rate control of TCP by using ECN. These concepts can also be used for multi-path routing. For Single Bit Resource Marking (SBRM) proposed in [ZK02, Zim05] simulation results for the discussed example are presented in [EK06b]. These results are similar to the findings for PC1 and PC3.

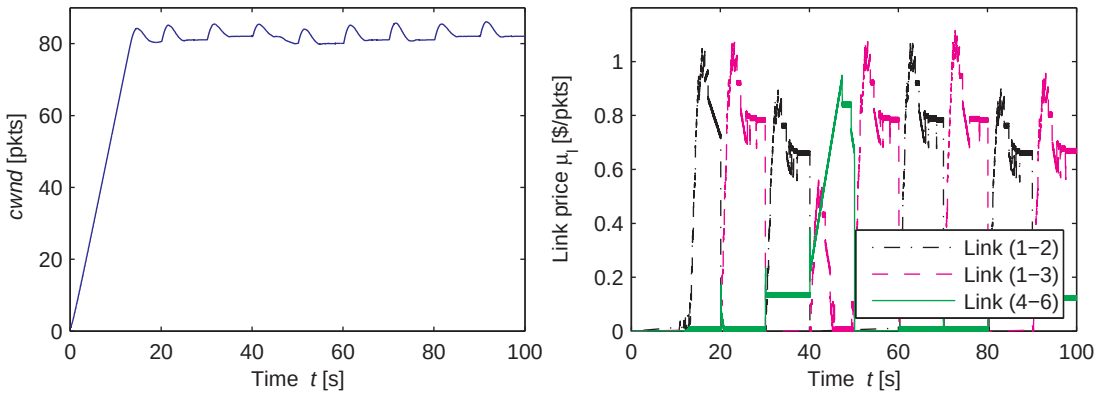
4.4 Single-path Routing

In the first part of this chapter we assume that the sources have topological information about the network and determine the routes to the sink. Source routing is supported by IP [Pos81], but it is rarely used in today's Internet. The reasons are security concerns and a slower processing at the routers for packets with variable header length, which is needed to store the routing information set by a source [PD07]. Instead, hop-by-hop routing protocols are used. One class are link state (LS) protocols. Here, routers broadcast the costs of their links to all other routers in the network. Hence, the nodes learn about the topology of the network and by using the Dijkstra algorithm they find the shortest path to every other node. A dynamic routing can be realised when the link costs depend on the level of congestion. If a link price rule like PC1 or PC3 is used to compute the link costs, then also for hop-by-hop routing the same metric controls the resource allocation at the network and transport layer.

With hop-by-hop routing a single route is chosen between a source-sink pair. As a consequence, when elastic traffic uses this route it consumes all available resources at the bottleneck link and the cost of that link will increase. Hence, the shortest path between the source-sink pair might change and traffic is re-routed. For the same reason, the link price will increase on the bottleneck link of the new route in the following and will decrease on the links of the old, now unused, route. Thus, traffic is re-routed again and oscillations occur.

This behaviour is presented for the example discussed throughout this chapter. Firstly, assume only source 1 sends traffic to the sink in node 6. The price rule PC3 in (4.17) is used for rate control and routing. Furthermore, link costs are updated every $t_{\mathrm{LS}} = 10\,\mathrm{s}$ and broadcasted in the network. Since the link price with PC3 is zero for non-congested links, a static value is added to the link costs for routing. In simulations the static portion of the link costs are set to 1. The congestion window of the source and the link prices are depicted in Figure 4.6. In the simulation the routing shows oscillations. Between 0-20, 30-40, 60-70 and 80-90 seconds the route (1-2-4-6) is chosen as shortest path, whereas between 20-30, 40-50, 70-80 and 90-100 seconds route (1-3-4-6) and between 50-60 seconds route (1-3-5-6) is used.

Since all links have the same capacity, re-routing hardly affects the congestion window in Figure 4.6(a). Just when traffic is re-routed the congestion window increases because the link prices of the new route are zero. However, as depicted in Figure 4.6(b) the price increases fast on the congested link for a large congestion window and the congestion



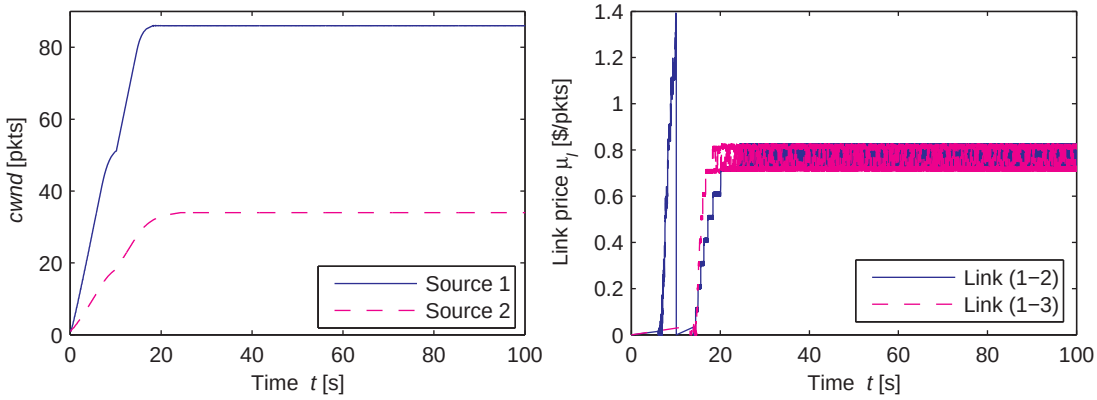
(a) Congestion window (b) Link price

Figure 4.6: Results for single-path routing with one source

window decreases rapidly again. In detail, in the time intervals 10-20, 30-40, 60-70 and 80-90 seconds the link (1-2) is the congested link and its price in Figure 4.6(b) is greater zero. Between 20-30, 50-60, 70-80 and 90-100 seconds link (1-3) is overloaded. Interestingly, between 40-50 seconds the price of link (4-6) increases faster than on link (1-3). This can happen because link (4-6) was used also by the route (1-2-4-6) in the previous interval. Thus, with a queue size greater zero, the link price might increase faster than on link (1-3).

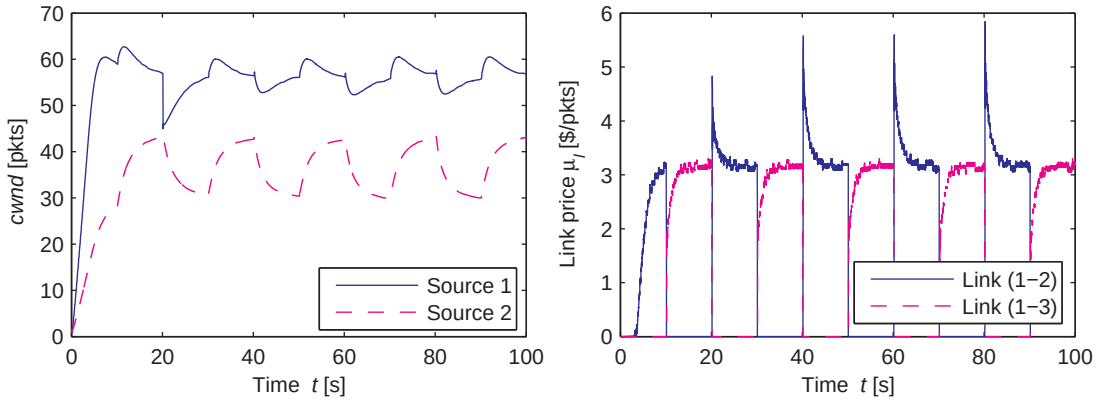
With Equal-Cost Multi-Path (ECMP) traffic is distributed on routes with the same costs. However, this does not avoid oscillations, because traffic is distributed equally on the routes. This can be seen for the aforementioned example of a source in node 1. If the network is uncongested, traffic is split in node 1 equally and routed on link (1-2) and link (1-3). However, the flow traversing node 3 is split again over link (3-4) and (3-5). Hence, the load on link (4-6) is higher than on link (5-6), because at node 4 traffic from node 2 and node 3 is aggregated. The link price of (4-6) is higher than on other links and after broadcasting the new link costs traffic might be not routed on this link and oscillations occur in the following.

For the example with a single source no routing exists which finds the optimum of the optimisation problem in (3.1)-(3.4). However, for the example with two sources in node 1 and sinks in node 2 and node 6, respectively, a optimal solution for single-path routing exists. When source 1 routes traffic over (1-3-4-6) (or (1-3-5-6)) and source 2 uses (1-2) resources are used efficiently and fair and (3.1)-(3.4) is maximised. Results for the example with two sources are depicted in Figure 4.7 and Figure 4.8 for $W = 1000$ and $W = 2000$, respectively.



(a) Congestion window (b) Link price

Figure 4.7: Results for single-path routing with two sources ($W = 1000$)



(a) Congestion window (b) Link price

Figure 4.8: Results for single-path routing with two sources ($W = 2000$)

These results indicate that the occurrence of oscillations depend on the chosen parameters. With $W = 1000$ the optimal solution is found. The traffic of both sources is routed on the link (1-2) at the start of the simulation. Hence, as depicted in Figure 4.7(b) the link price of (1-2) increases. After 10 seconds link costs are updated and traffic from node 1 to node 6 is re-routed on link (1-3). In the following link prices on (1-2) and (1-3) as well as the congestion windows of the two sources in Figure 4.7(a) converge to the optimal values (by considering also the RTTs).

By changing the willingness-to-pay to $W = 2000$ oscillations occur for this example. The link price on (1-2) in Figure 4.8(b) is larger as with $W = 1000$. Thus, when link costs are updated both flows are re-routed and avoid the link (1-2). However, on the alternative link (1-3) the price increases and oscillations occur over the whole simulation time. Also the congestion windows in Figure 4.8(a) vary considerably, because the routes have different round-trip times. This causes additional variations of the link prices (e.g. the peaks at multiples of 20 seconds).

Besides the mentioned problem of route oscillations further concerns argue against dynamic single-path routing. For example, switching between routes with different propagation delays may reorder TCP packets. This may heavily affect the control of the congestion window. Furthermore, with heterogeneous capacities re-routing may cause congestion on links with low resources. Both problems can cause additional instabilities of the network.

4.5 Perfect Competition and Market Failure

The resource pricing approach depends on the same conditions as for real markets, which are well known from economic theory (e.g. in [Dem90]). Important for this work is the access to market information and the perfect competition. Market failures can occur if both conditions are not satisfied.

In the first case price information has to be available to the customers. For the discussed approach a sender obtains the price information of a route with the acknowledgements from the receiver. Hence, a small probing rate in (4.10) for the multi-path routing is needed to obtain these prices. For hop-by-hop routing also the routers need information about the prices. By using a link state protocol the link costs are broadcasted. Hence, the access to market information is available. Depending on the delays in the network and broadcasting intervals the information is valid or outdated. Hence, fluctuations can occur and differences in the speed of convergence are observed in the simulations.

The condition of perfect competition is satisfied for multi-path routing, because the link prices converge to the market equilibrium. Since each user sends traffic over multiple routes no link has the market power and influences the prices. This is different to the original model, where the routing is fixed and links have a monopoly.

With single-path routing market failure occurs. In our model we assume the TCP senders are price taking. This means they do not control prices [CW03]. But as seen in Figure 4.6 users have influence on the prices. As consequence oscillations and inefficiency occur. With hop-by-hop routing flows are not routed independently. Hence, market failure will not disappear for a large number of users. The case, where users are not price taking but price anticipating is studied for a similar model in [Joh04].

4.6 Conclusion

The main findings of this chapter are twofold. The simulation results for multi-path routing show that congestion pricing is an applicable concept for the resource allocation in networks and for the development of a combined model for rate control and routing. By sending over multiple routes in parallel the necessary price information of the different routes is collected. Furthermore, by adapting the rate on each route gradually, also the link prices are affected gradually. The proposed distributed algorithm converges to an efficient and fair resource allocation.

For single-path routing different scenarios are presented for which oscillations and inefficiencies occur. In the discussed pricing model entities are price taking and do not anticipate the influence of their decisions on the prices. Thus, traffic is re-routed although link prices are affected by the own sending rate only and persistent oscillations occur. Also with ECMP oscillations are not avoided because traffic is split equally and might be aggregated at other links again. Even when the optimum for the global optimisation problem is achievable with single-path routing this might not be reached by the distributed algorithm.

In conclusion, congestion pricing is suitable for a combined approach of rate control and routing, but not with single-path routing. Hence, the following chapters will discuss congestion pricing for overlay networks, where multiple routes can be used.

Chapter 5

Resource Pricing for Peer-to-Peer Networks

In this chapter resource pricing is applied to Peer-to-Peer (P2P) networks to ensure a fair allocation of resources. The differences to congestion pricing in IP networks are described and a distributed algorithm based on locally available information is proposed. Proofs for the stability of the global optimum for the proposed algorithm are presented. The approach is applied to P2P content distribution networks and is extended to improve the availability of pieces in a multi-source download.

In resource pricing at the transport layer routers charge a price for the usage of their resources and the sending agents of the users adapt their sending rates depending on the aggregated path price. Rate control in P2P networks is different. A peer chooses freely the peers it provides service to and allocates resources for the service (e.g. upload bandwidth for file-sharing applications or processor cycles for distributed computing). Therefore, resources are used efficiently at a peer when at least one other peer requests a service from it. However, a requesting peer is normally serviced by several providers in parallel. Hence, its total service rate is accumulated over its providers. This causes unfairness in the network with respect to the total service rate, where some peers are better off when connected to a large number of service providing peers or to peers with high capacity.

In this chapter resource pricing for P2P networks is proposed to ensure fairness where service requesting peers offer prices for a service, which they consume. Prices are only used as a control signal for allocating the resources at a provider, whereby the achieved performance of a serviced peer solely depends on its willingness-to-pay. In addition, if the willingness-to-pay depends on the contributed resources of a peer, this scheme provides

an incentive to peers. The chosen approach is based on the same global optimisation problem as for IP networks. The proposed distributed algorithm is applied to P2P content distribution networks, but it is valid also for other resource allocation problems (e.g. Grid computing). This chapter is based on the ideas and results of [EK05, EK06c, EK06a, EK07b, EK07a, EK08].

5.1 P2P Network Model

The resource allocation in P2P networks is based on the network model in Section 3.1. However, the notation is slightly changed. Thus, the global optimisation problem in (3.1)-(3.4) is restated in this section in the context of P2P networks.

Consider a P2P network consisting of a set of peers $\mathcal{P}$ and a set of services, whereby each peer $p \in \mathcal{P}$ is interested in one or several services and/or offers different services. Providing a service consumes resources. In this work we concentrate on resources (e.g. access bandwidth) which are divisible and where any allocation of resources has a benefit for a requesting peer. We assume that the resource is scarce such that competition is present and denote the capacity of this resource at peer p as C_p.

To differentiate between service providing and service requesting peers in our mathematical model we introduce the set of service providers or servers $\mathcal{S}$ and the set of service customers or clients $\mathcal{C}$. The terms server and client are also used for P2P networks. In this context, a server is a peer which offers at least one service and a client is a peer which requests at least one service. A peer can be a server and a client at the same time.

Since each peer has only a partial view of the whole P2P network, a service requesting peer is not aware of all peers that provide this service, and vice versa. We define the set of peers, which offer at least one service to the client c as the set of servers $\mathcal{S}(c)$ of client c. The other way round $\mathcal{C}(s)$ is the set of the known clients of the server s. Furthermore, if $c \in \mathcal{C}(s)$ then also $s \in \mathcal{S}(c)$ holds. The client-server architecture can be interpreted as a special case of a P2P application where the set $\mathcal{S}(c)$ consists of one single server.

Suppose the utility of a client c is defined by a utility function U_c, which depends on the total service rate y_c. Hence, similar to (3.1)-(3.4) the optimisation problem for P2P

networks is

$$\text{P2P SYSTEM}:$$

$$\text{maximise} \quad \sum_{c\in\mathcal{C}} U_c(y_c) \tag{5.1}$$

$$\text{subject to} \quad \sum_{s\in\mathcal{S}(c)} x_{sc} = y_c, \quad \forall c \in \mathcal{C} \tag{5.2}$$

$$\sum_{c\in\mathcal{C}(s)} x_{sc} \leq C_s, \quad \forall s \in \mathcal{S} \tag{5.3}$$

$$\text{over} \quad x_{sc} \geq 0. \tag{5.4}$$

Maximising the aggregated utility of the service rate y_c over all clients is the objective of the whole system, where y_c is the total of the rates x_{sc} of all servers s of c. This problem is constrained by the resource capacity at the servers.

Comparing the global optimisation problem for P2P networks to the problem in (3.1)-(3.4) following differences can be seen. The term user becomes client. In the basic model it makes no difference to differentiate between the sender and the sink. Each sender serves one sink and each sink is served by one sender. Only multiple routes exist between the sender and the sink. This is different for P2P networks. Here, each client is served by multiple servers. Thus, the objective is not with respect to a user but more precisely defined with respect to the client. Furthermore, the P2P model takes only the overlay network into account. Hence, only the capacity constraints at the servers are considered.

With a concave, strictly increasing utility function both optimisation problems have a unique optimum with respect to the total service rate y_c. Furthermore, when the utility function in (3.6) is used also for P2P networks the results in Section 3.3 apply and a weighted proportional fair resource allocation is realised. Additionally, when the P2P network is connected also the results from Section 3.4 apply. Since we use results from Section 3.4 in the following we restate them for the terminology of P2P networks: For a connected P2P network and the utility function in (3.6) the optimum of (5.1)-(5.4) with respect to the total download rate y_c and the offered price λ_c is

$$y_c^* = \sum_{s\in\mathcal{S}(c)} x_{sc}^* = \frac{W_c \sum_{s\in\mathcal{S}} C_s}{\sum_{d\in\mathcal{C}} W_d} \tag{5.5}$$

and

$$\lambda_c^* = \frac{\sum_{d\in\mathcal{C}} W_d}{\sum_{s\in\mathcal{S}} C_s}, \tag{5.6}$$

respectively. These results are based on (3.34), (3.35) and (3.36).

With the logarithmic utility function in (3.6) we assume the law of diminishing returns applies for peers. This means that the performance enhancement decreases with increasing total service rate. Such a utility function represents users with elastic traffic demands [She95] and is applicable when users have no delay requirements. It is used often for data transfers [Kel97]. Hence, it is relevant for P2P file-sharing or distributed computing without specific deadlines.

5.2 Distributed Algorithm

Identical to the procedure in Chapter 4 the global optimisation problem is divided into sub-problems. These are derived with the Lagrangian

$$L^{\mathrm{P2P}}(x,y,\lambda,\nu,n) = \sum_{c\in\mathcal{C}}\left(U_c(y_c)-\lambda_c\left(y_c-\sum_{s\in\mathcal{S}(c)}x_{sc}\right)\right)+\sum_{s\in\mathcal{S}}\nu_s\left(C_s-\sum_{c\in\mathcal{C}(s)}x_{sc}-n_s\right), \tag{5.7}$$

where $n_s \geq 0$ is the slack variable for the inequality constraint in (5.3) for server s.

For P2P networks sub-problems for the clients and servers are defined as

CLIENT c :

$$\text{maximise} \quad U_c(y_c)-\lambda_c y_c \tag{5.8}$$

$$\text{over} \quad y_c \geq 0 \tag{5.9}$$

SERVER s :

$$\text{maximise} \quad \sum_{c\in\mathcal{C}(s)}\lambda_c x_{sc} \tag{5.10}$$

$$\text{subject to} \quad \sum_{c\in\mathcal{C}(s)}x_{sc}\leq C_s \tag{5.11}$$

$$\text{over} \quad x_{sc}\geq 0. \tag{5.12}$$

Also the capacity constraint of the SERVER problem can be internalised and the Lagrange function of the SERVER s is

$$L_s^{\mathrm{SERVER}}(x,\lambda,\nu_s,n_s) = \sum_{c\in\mathcal{C}(s)}\lambda_c x_{sc}+\nu_s\left(C_s-\sum_{c\in\mathcal{C}(s)}x_{sc}-n_s\right). \tag{5.13}$$

The rate x_{sc} is set by the server s and also the capacity constraint is related to the servers.

Hence, we propose a primal algorithm for x_{sc} instead of a primal-dual approach as in Chapter 4. A simple method to optimise the Lagrange function is by iterative descent algorithms (see Section 3.2 of [BT89]). The gradient algorithm (or steepest descent) for maximising (5.13) with respect to x_{sc} is

$$x_{sc}(t+1) = x_{sc}(t) + \tilde{\gamma}\frac{\partial L_s^{\text{SERVER}}}{\partial x_{sc}}(t) = x_{sc}(t) + \tilde{\gamma}(\lambda_c(t) - \nu_s(t)), \tag{5.14}$$

where $\tilde{\gamma}$ is a positive step size. The offered price λ_c is computed by setting the derivative with respect to y_c of the CLIENT problem in (5.8) to zero (compare also with (3.27)). As in Chapter 4 we use the logarithmic utility function in (3.6). To ensure a bounded price offer if the total service rate is zero a small positive constant η is introduced. Hence,

$$\lambda_c(t) = \frac{W_c}{\max(\eta, y_c(t))} = \frac{W_c}{\max\left(\eta, \sum_{s\in\mathcal{S}(c)} x_{sc}(t)\right)}. \tag{5.15}$$

The charged price ν_s can not be derived from partial derivatives. Thus, ν_s is estimated by the average offered price per unit capacity. Hence, with non-negative sending rates (5.14) becomes

$$x_{sc}(t+1) = \max\left(0, x_{sc}(t) + \tilde{\gamma}\left(\lambda_c(t) - \frac{\sum_{d\in\mathcal{C}(s)} x_{sd}(t)\,\lambda_d(t)}{C_s}\right)\right). \tag{5.16}$$

An alternative is a scaled version of (5.16) where the difference between the offered and average offered price is scaled by x_{sc} and a positive step size γ. Hence,

$$x_{sc}(t+1) = \max\left(\varepsilon, x_{sc}(t) + \gamma x_{sc}(t)\left(\lambda_c(t) - \frac{\sum_{d\in\mathcal{C}(s)} x_{sd}(t)\,\lambda_d(t)}{C_s}\right)\right). \tag{5.17}$$

Here, the parameter ε is a small positive constant. Hence, the sending rate x_{sc} does not fall below ε. This models a kind of probing, which is necessary to receive information about the offered price of the client.

A scaled gradient algorithm is also used for the primal algorithm in [KMT98], on which the rule in (4.10) for the sending rate in Chapter 4 is based. Furthermore, (4.10) as well as (5.17) realise the additive increase/multiplicative decrease (AIMD) principle also used in TCP [Jac88]. With AIMD the rate control is conservative and severe congestion (and at worst a congestion collapse) is avoided [FF99]. The presented approach in this chapter is extended to a TCP variant for P2P networks in Chapter 6. Therefore, to realise AIMD the scaled version of the distributed algorithm is studied in the following.

Both rate control algorithms are interpreted from an economical perspective as follows: Since λ_c is the price per unit bandwidth, the product $x_{sc}\lambda_c$ is the total price which is paid by peer c. Hence, the sum over all clients in the numerator of (5.16) and (5.17) represents the total revenue of server s. Dividing the total revenue by C_s returns the average price server s obtains per unit capacity. If the offered price by client c is higher than the average price, then server s increases its upload rate to c and decreases it otherwise. The scaled version in (5.17) additionally scales the price difference by the rate x_{sc}. Hence, it is based on the total price rather than the price per unit bandwidth. This model reflects a market for service capacity where prices control the rate allocations. Since the service capacity is a non-storable commodity the market is a spot market, where the current demand influences the future prices.

In the following we show that the scaled gradient algorithm in (5.17) fulfils fundamental properties for the resource allocation in a distributed environment. We derive conditions for the efficiency of the algorithm and prove that the equilibrium point of (5.17) coincides with the optimum of the global optimisation problem in (5.1)-(5.4). Furthermore, global asymptotic stability of the equilibrium point is proven for a continuous model without delays as well as a discrete model with delays.

5.2.1 Efficiency

The distributed algorithm is efficient if the total upload capacity is used. For the scaled version in (5.17) this can be easily shown. Since any small positive number for ε is sufficient to realise a probing, it is neglected in the following. This simplification is very similar to the approach in [KV05] for joint routing and rate-control.

Theorem 1 *The distributed algorithm in (5.17) fully utilises the upload capacity under the assumption that in the previous step all resources are used. I.e.,* $\sum_{c\in\mathcal{C}(s)} x_{sc}(t) = C_s$.

Proof: Summing up over all upload rates of server s

$$\sum_{c\in\mathcal{C}(s)} x_{sc}(t+1)$$

$$= \sum_{c\in\mathcal{C}(s)} \left(x_{sc}(t) + \gamma x_{sc}(t) \left(\lambda_c(t) - \frac{\sum_{d\in\mathcal{C}(s)} x_{sd}(t)\lambda_d(t)}{C_s} \right) \right) \tag{5.18}$$

$$= \sum_{c\in\mathcal{C}(s)} x_{sc}(t) + \gamma \left(\sum_{c\in\mathcal{C}(s)} x_{sc}(t)\lambda_c(t) - \frac{\sum_{c\in\mathcal{S}(s)} x_{sc}(t)}{C_s} \sum_{d\in\mathcal{C}(s)} x_{sd}(t)\lambda_d(t) \right) \tag{5.19}$$

Making use of the assumption $\sum_{c\in\mathcal{C}(s)} x_{sc}(t) = C_s$ we get

$$\sum_{c\in\mathcal{C}(s)} x_{sc}(t) + \gamma\left(\sum_{c\in\mathcal{C}(s)} x_{sc}(t)\lambda_c(t) - \frac{\sum_{c\in\mathcal{S}(s)} x_{sc}(t)}{C_s}\sum_{d\in\mathcal{C}(s)} x_{sd}(t)\lambda_d(t)\right)$$

$$= \sum_{c\in\mathcal{C}(s)} x_{sc}(t) \tag{5.20}$$

$$= C_s \tag{5.21}$$

■

Hence, the distributed algorithm is also efficient in all rate allocations when the initial rate allocation is efficient. This can be realised by splitting the upload capacity evenly over all connected peers at the start (under the assumption that protocols on lower layers (e.g. TCP) allows this). Thus, the distributed algorithm produces a feasible solution with respect to the capacity constraints in the next iteration, if the current solution is feasible.

Furthermore, a value for the step size γ can be computed, which ensures efficiency at each iteration step. Denoting this step size by γ_e and demanding for $C_s = \sum_{c\in\mathcal{C}(s)} x_{sc}(t+1)$ we compute γ_e with (5.17). Thus,

$$C_s = \sum_{c\in\mathcal{C}(s)} x_{sc}(t+1) \tag{5.22}$$

$$= \sum_{c\in\mathcal{C}(s)} x_{sc}(t) + \gamma_e\left(1 - \frac{\sum_{c\in\mathcal{C}(s)} x_{sc}(t)}{C_s}\right)\sum_{d\in\mathcal{C}(s)} x_{sd}(t)\lambda_d(t) \tag{5.23}$$

$$\Rightarrow \gamma_e = \frac{C_s}{\sum_{d\in\mathcal{C}(s)} x_{sd}(t)\lambda_d(t)}. \tag{5.24}$$

By using (5.24) the equation (5.17) reduces to

$$x_{sc}(t+1) = \max\left(\varepsilon, \frac{x_{sc}(t)\lambda_c(t)}{\sum_{d\in\mathcal{C}(s)} x_{sd}(t)\lambda_d(t)} C_s\right). \tag{5.25}$$

Although (5.25) ensures efficiency in a continuous model, oscillations and/or instability can occur when delayed information is used in a real environment where peers adjust prices and service rates in discrete steps. Therefore, a smaller value for γ_e should be used. By introducing $\overline{\gamma}$ with $\overline{\gamma} = \gamma/\gamma_e$ the rule for the sending rate in (5.17) becomes

$$x_{sc}(t+1) = \max\left(\varepsilon, x_{sc}(t)\,(1-\overline{\gamma}) + \overline{\gamma}\frac{x_{sc}(t)\lambda_c(t)}{\sum_{d\in\mathcal{C}(s)} x_{sd}(t)\lambda_d(t)} C_s\right), \tag{5.26}$$

whereby $0 < \bar{\gamma} \leq 1$.

In the following the original version in (5.17) is used, because it is closer related to the optimisation problems in (5.1)-(5.4) and (5.8)-(5.12) and to the distributed algorithm in Section 4 and to similar work in [KMT98, Key01, KV05]. Furthermore, it is shown by simulations in Chapter 5.3 that (5.17) shows fast convergence to an efficient solution for different scenarios.

5.2.2 Stability

The distributed algorithm is derived from local sub-problems and these sub-problems are derived from a global optimisation problem. Now, we show that the distributed algorithm finds the solution of the global optimisation problem. This means, we prove firstly that the equilibrium point of the difference equation is equal to the optimum of the global problem and secondly that the equilibrium point is globally asymptotically stable.

For an ordinary difference equation $x(n+1) = f(x(n))$ with the initial condition $x(n_0) = x_0$ the *equilibrium point* (or critical point, fixed point) x^* is defined by $f(x^*) = x^*$ [Ela05]. Stability theory investigates the behaviour of the difference equation around an equilibrium point. In this work we will discuss global asymptotic stability. An equilibrium point x^* is globally asymptotically stable if $f(x(n)) \rightarrow x^*$ for $n \rightarrow \infty$ for any initial condition x_0 [Ela05]. Many different approaches exist to prove stability of a system. Here, we will follow a procedure similar to the one used in the congestion pricing literature for TCP-like algorithms. Here, stability was shown on different levels of abstraction. Firstly, it is proven for a continuous model without delays [KMT98] and secondly for a discrete model with delays [JT01]. Considering delays makes the stability proof more difficult and frequently additional assumptions are made.

In the following we will determine the equilibrium point of the scaled distributed algorithm. Like in Section 5.2.1 the influence of ε is also neglected here. Additionally, also any small positive number for η is sufficient. Therefore, also η is neglected in the following.

Theorem 2 *The equilibrium point (y^*, λ^*) of the system (5.15) and (5.17) is identical to the solution of the global optimisation problem in (5.5) and (5.6) for a connected network.*

Proof: Assuming $x^*_{sc}(t) > \varepsilon$ the equilibrium point of (5.17) is obtained when the term $x_{sc}(t+1) - x_{sc}(t)$ is equal zero.

Hence,

$$0 = \gamma x^*_{sc}(t)\left(\lambda^*_c(t) - \frac{\sum_{d\in\mathcal{C}(s)} x^*_{sd}(t)\,\lambda^*_d(t)}{C_s}\right) \tag{5.27}$$

$$\Rightarrow C_s = \frac{\sum_{d\in\mathcal{C}(s)} x^*_{sd}\lambda^*_d}{\lambda^*_c}. \tag{5.28}$$

By summing up over all servers and using (5.15)

$$\sum_{s\in\mathcal{S}} C_s = \sum_{s\in\mathcal{S}} \frac{\sum_{d\in\mathcal{C}(s)} x_{sd}\lambda^*_d}{\lambda^*_c} \tag{5.29}$$

$$= \frac{y^*_c}{W_c}\sum_{d\in\mathcal{C}} W_d \tag{5.30}$$

$$\Rightarrow y^*_c = \frac{W_c\sum_{s\in\mathcal{S}} C_s}{\sum_{d\in\mathcal{C}} W_d}. \tag{5.31}$$

Hence, the equilibrium point y^*_c of the total service rate is equal to (5.5). Furthermore, inserting (5.31) into (5.15) we get the price offer in equilibrium

$$\lambda^*_c = \lambda^* = \frac{\sum_{d\in\mathcal{C}} W_d}{\sum_{s\in\mathcal{S}} C_s}, \tag{5.32}$$

which is identical to (5.6). ■

Furthermore, the equilibrium point is efficient, because with (5.28) and (5.32) we get

$$C_s = \sum_{d\in\mathcal{C}(s)} x^*_{sd}. \tag{5.33}$$

The claim of global asymptotic stability of the equilibrium point in (5.31) and (5.32) is based on a Lyapunov function for a continuous model without delays. The same idea is also used in [KMT98] for a rate control algorithm at the transport layer. Suppose a system has an equilibrium point at the origin. Briefly, Lyapunov's method says the equilibrium point is asymptotically stable if there exists a function $V(x)$ that is positive definite* and the partial derivative $\dot{V}$ along the trajectories of the system is negative definite. It is globally asymptotically stable if additionally $V(x) \to \infty$ if $x \to \infty$ (see Theorem 9.6.1 in [BD92] and Chapter 4 in [Kha00] for details).

Based on the difference equation (5.17) the differential equation for a continuous model

*V is positive definite on some domain D if $V(0) = 0$ and $V(x) > 0$ for all other points in D. It is negative definite if $V(0) = 0$ and $V(x) < 0$ for all other points in D [BD92]

is given by

$$\frac{d}{dt}x_{sc}(t) = \gamma x_{sc}(t)\left(\lambda_c(t) - \frac{\sum_{d\in\mathcal{C}(s)} x_{sd}(t)\,\lambda_d(t)}{C_s}\right). \tag{5.34}$$

We prove asymptotic stability in two steps. This means, stability is shown for the total service rate provided by a service provider s (i.e. $z_s = \sum_{c\in\mathcal{C}(s)} x_{sc}$) followed by a proof for the total service rate y_c of a customer c. We define the differential equation with respect to the total service rate provided by a peer as

$$\frac{d}{dt}z_s(t) = \sum_{c\in\mathcal{C}(s)} \frac{d}{dt}x_{sc}(t). \tag{5.35}$$

The convergence of (5.35) to (5.33) is proven in Theorem 3.

Theorem 3 *Considering the differential equation (5.35) the unique equilibrium point $z_s^* = \sum_{c\in\mathcal{C}(s)} x_{sc}^* = C_s$ is globally asymptotically stable for all $s \in \mathcal{S}$.*

Proof: The function

$$V(z) = \sum_{s\in\mathcal{S}} \frac{1}{2}(C_s - z_s)^2 \tag{5.36}$$

is a positive definite function, which is zero if the total service rate of each server is equal its capacity. I.e., at the equilibrium point only. Furthermore, if $z \to \infty$ then $V(z) \to \infty$. The derivative of $V(z)$ along the trajectories of the differential equation is

$$\begin{aligned}
\dot{V}(z) &= \sum_{s\in\mathcal{S}} \frac{\partial V(z)}{\partial z_s}\cdot\frac{dz_s(t)}{dt} && (5.37)\\
&= -\sum_{s\in\mathcal{S}} (C_s - z_s(t)) \sum_{c\in\mathcal{C}(s)} \gamma x_{sc}(t)\left(\lambda_c(t) - \frac{\sum_{d\in\mathcal{C}(s)} x_{sd}(t)\lambda_d(t)}{C_s}\right) && (5.38)\\
&= -\gamma\sum_{s\in\mathcal{S}} (C_s - z_s(t)) \sum_{c\in\mathcal{C}(s)} x_{sc}(t)\lambda_c(t)\left(1 - \frac{\sum_{d\in\mathcal{C}(s)} x_{sc}(t)}{C_s}\right) && (5.39)\\
&= -\gamma\sum_{s\in\mathcal{S}} C_s\left(1 - \frac{z_s(t)}{C_s}\right)^2 \sum_{c\in\mathcal{C}(s)} x_{sc}(t)\lambda_c(t) && (5.40)\\
&\leq 0
\end{aligned}$$

The equation (5.40) is a negative definite function, which is zero at the equilibrium point only. Hence, (5.36) provides a strict Lyapunov function for the differential equation (5.35) and the equilibrium point $z_s^* = C_s, \;\; \forall s \in \mathcal{S}$ is globally asymptotically stable. ■

We have proven in Theorem 1 that the distributed algorithm remains efficient if it is efficient in the preceding update step. Furthermore, we have shown in Theorem 3 that the efficient equilibrium point is globally asymptotically stable. We use these two results in the following.

Theorem 4 *Considering the system (5.15) and (5.34) the unique equilibrium point (5.31) and (5.32) is asymptotically stable for a connected network under the assumption that* $z = (z_s \,|\, z_s = C_s, s \in \mathcal{S})$ *in all iteration steps.*

Proof: The function

$$V(y) = \sum_{c \in \mathcal{C}} (W_c \ln y_c^* - W_c \ln y_c) \tag{5.41}$$

is a positive definite function, if $\sum_{c \in \mathcal{C}(s)} x_{sc} \leq C_s, \ \forall s \in \mathcal{S}$. It is zero at the equilibrium point only. The derivative of $V(y)$ along the trajectories of the differential equation is

$$\dot{V}(y) = -\sum_{c \in \mathcal{C}} \frac{\partial V(y)}{\partial y_c} \cdot \frac{dy_c(t)}{dt} \tag{5.42}$$

$$= -\sum_{c \in \mathcal{C}} \frac{W_c}{y_c(t)} \sum_{s \in \mathcal{S}(c)} \gamma x_{sc}(t) \left(\lambda_c(t) - \frac{\sum_{d \in \mathcal{C}(s)} x_{sd}(t) \lambda_d(t)}{C_s} \right). \tag{5.43}$$

By inserting (5.15) into (5.43) we get

$$\dot{V}(y) = -\gamma \sum_{c \in \mathcal{C}} \sum_{s \in \mathcal{S}(c)} \lambda_c^2(t) x_{sc}(t) + \gamma \sum_{s \in \mathcal{S}} \frac{\left(\sum_{d \in \mathcal{C}(s)} x_{sd}(t) \lambda_d(t) \right)^2}{C_s} \tag{5.44}$$

$$= -\gamma \sum_{c \in \mathcal{C}} \sum_{s \in \mathcal{S}(c)} \lambda_c^2(t) x_{sc}(t) \left(1 - \frac{x_{sc}(t)}{C_s} \right) + \gamma \sum_{s \in \mathcal{S}} \sum_{c \in \mathcal{C}(s)} \sum_{d \in \mathcal{C}(s) \setminus \{c\}} \frac{x_{sc}(t) \lambda_c(t) x_{sd}(t) \lambda_d(t)}{C_s}. \tag{5.45}$$

By adding $\lambda_c^2(t) x_{sc}(t) \sum_{d \in \mathcal{C}(s) \setminus \{c\}} x_{sd}(t) / C_s$ to every provider-customer pair in the first part of (5.45) and subtracting the same term from the second part

$$\dot{V}(y) = -\gamma \sum_{c \in \mathcal{C}} \sum_{s \in \mathcal{S}(c)} \lambda_c^2(t) x_{sc}(t) \left(1 - \frac{\sum_{d \in \mathcal{C}(s)} x_{sd}(t)}{C_s} \right)$$
$$+ \gamma \sum_{s \in \mathcal{S}} \sum_{c \in \mathcal{C}(s)} \sum_{d \in \mathcal{C}(s) \setminus \{c\}} \left(\frac{x_{sc}(t) \lambda_c(t) x_{sd}(t) \lambda_d(t)}{C_s} - \lambda_c^2(t) x_{sc}(t) \frac{x_{sd}(t)}{C_s} \right) \tag{5.46}$$

and

$$\dot{V}(y) = -\gamma \sum_{c\in\mathcal{C}} \sum_{s\in\mathcal{S}(c)} \lambda_c^2(t) x_{sc}(t) \left(1 - \frac{\sum_{d\in\mathcal{C}(s)} x_{sd}(t)}{C_s}\right) - \gamma \sum_{s\in\mathcal{S}} \sum_{c\in\mathcal{C}(s)} \sum_{d\in\mathcal{C}(s)\setminus\{c\}} \frac{x_{sc}(t)x_{sd}(t)}{2C_s} \left(\lambda_c(t) - \lambda_d(t)\right)^2 . \tag{5.47}$$

Finally, based on the assumption that $z_s(t) = \sum_{d\in\mathcal{C}(s)} x_{sd}(t) = C_s$ the equation (5.47) becomes

$$\begin{aligned} \dot{V}(y) &= -\gamma \sum_{s\in\mathcal{S}} \sum_{c\in\mathcal{C}(s)} \sum_{d\in\mathcal{C}(s)\setminus\{c\}} \frac{x_{sc}(t)x_{sd}(t)}{2C_s} \left(\lambda_c(t) - \lambda_d(t)\right)^2 \qquad (5.48) \\ &\leq 0 \end{aligned}$$

Equation (5.48) is a negative definite function. It is only zero, if $\lambda_c(t) = \lambda_d(t)$ holds for all clients of all servers, i.e. at the equilibrium point in (5.32). Hence, (5.41) provides a strict Lyapunov function for the system (5.15) and (5.34) and the unique equilibrium point (5.31) and (5.32) is asymptotically stable. ■

5.2.3 Delay Stability

The convergence of the distributed algorithm is affected by asynchronous updates of the peers and delayed price and rate information. Thus, additionally to the results in Section 5.2.2 for a continuous model without delays the stability for a discrete model with delays is studied. Assume each peer updates its sending rates or price offers each time interval T and peers are not synchronised. Hence, a server uses delayed price information from its clients and a client uses delayed sending rates to compute its price offer. Communication delays between the peers are not explicitly considered in the following, because it is assumed that communication delays are much smaller than the time interval T. For example, in the context of P2P content distribution networks discussed in Section 5.4 a time interval $T = 10\,\mathrm{s}$ is used. This is the default parameter in BitTorrent and the end-to-end delay between peers in the Internet is in the general case in the order of tenth and hundreds of milliseconds.

Also for discrete models with delays different methods can be used to prove the stability of an equilibrium point. Often control theory is applied and the problem is transformed to the Laplace domain [JT01, Zim05]. In our model, we assume variables are updated per time interval. Hence, the delay between peers is bounded. This simplifies the stability

proof. In [BT89] different algorithms and their convergence are discussed. We will use one proposition from [BT89] to prove the stability of the proposed distributed algorithm with delayed price and rate information in this work.

The following proof is based on (5.26) and is generalised to (5.17). As in previous sections the influence of ε and η is neglected in the following. Assume τ_{sc} and τ_{cs} are the time intervals between update steps of the server s and the client c and vice versa. Furthermore, $T = \tau_{sc} + \tau_{cs}$ and the asynchronous version of (5.26) and (5.15) is

$$x_{sc}(t+T) = \max\left(\varepsilon, x_{sc}(t)\,(1-\overline{\gamma}) + \overline{\gamma}\frac{x_{sc}(t)\lambda_c(t+\tau_{sc})}{\sum_{d\in\mathcal{C}(s)} x_{sd}(t)\lambda_d(t+\tau_{sd})}C_s\right) \tag{5.49}$$

and

$$\lambda_c(t+\tau_{sc}) = \frac{W_c}{\max\left(\eta, \frac{1}{T}\sum_{s\in\mathcal{S}(c)} x_{sc}(t-T)\,\tau_{cs} + x_{sc}(t)\,\tau_{sc}\right)}, \tag{5.50}$$

respectively. Each peer runs an update once during time interval T and the price information and sending rates used by any peer is delayed by at most T time units. Thus, by adapting the rate with (5.49) rate information from other peers that are implicitly used in the price information is delayed by at most $2T$ and the delay is bounded. Hence, the distributed algorithm belongs to the class of partially asynchronous algorithms discussed in Chapter 7 of [BT89]. Here, the authors show in Proposition 2.2 that partial asynchronous algorithms defined by

$$f(x) = (1-\gamma)x + \gamma h(x) \tag{5.51}$$

with $\gamma \in (0,1)$ converge to an equilibrium or fixed point if the following assumptions hold for the function $h(x)$:

Assumption 2.1 [BT89, p.490]

(a) The set of fixed points $X^* = \{x \in \mathbb{R}^n \,|\, h(x) = x\}$ is nonempty.

(b) The function $h(x)$ is continuous.

(c) The function $h(x)$ is nonexpansive, that is, it satisfies

$$\|h(x) - x^*\|_\infty \leq \|x - x^*\|_\infty$$

and

Assumption 2.2 [BT89, p.492]

(a) The set of fixed points X^* is convex.

(b) Define the set of indices of coordinates of x that are farthest away from x^* as

$$I(x;x^*) = \{i \,|\, |x_i - x_i^*| = \|x - x^*\|_\infty\}.$$

Additionally, define a set of vectors y that agree with x in the components that are farthest away from x^* as

$$U(x;x^*) = \{\, y \in \mathbb{R}^n \,|\, y_i = x_i \;\; \forall i \in I(x;x^*), \\ \text{and} \;\; |y_i - x_i^*| < \|x - x^*\|_\infty \;\; \forall i \notin I(x;x^*)\}.$$

For every $x \in \mathbb{R}^n$ and $x^* \in X^*$ such that $\|x - x^*\|_\infty = g(x) > 0$, there exists some $i \in I(x;x^*)$ such that $h_i(y) \neq y_i$ for all $y \in U(x;x^*)$.

Proposition 2.2 from [BT89] is used in the following. Like in Section 5.2.2 we present the convergence proof for the total upload rate z_s of server s and the total download rate y_c of client c in two separate steps. Analogue to Theorem 1 it is provable that also (5.26) is efficient in all succeeding update steps if the current update step is efficient. Hence, the two step approach is applicable.

Theorem 5 *The sequence $\{z(t)\}$ generated by the partial asynchronous iteration $z(t+T) = \big(z_s(t+T) \,|\, z_s(t+T) = \sum_{c \in \mathcal{C}(s)} x_{sc}(t+T), \;\; s \in \mathcal{S}\big)$ with $x_{sc}(t+T)$ defined in (5.49) converges to $z^* = (z_s \,|\, z_s = C_s, s \in \mathcal{S})$.*

Proof: The set of fixed points consists of the unique fixed point $z^* = (C_s, s \in \mathcal{S})$. Hence, the set is nonempty and convex[†]. Assumption 2.1(a) and Assumption 2.2(a) hold. Furthermore, by neglecting ε and using (5.49) z_s reduces to

$$z_s(t+T) = z_s(t)\,(1 - \overline{\gamma}) + \overline{\gamma} C_s. \tag{5.52}$$

With $h(z) = (C_s, s \in \mathcal{S})$ the function $h(z)$ is a constant function which is defined over all real numbers. Hence it is continuous and Assumption 2.1(b) is fulfilled. The maximum norm $||h(z) - z^*||_\infty$ is zero and $h(z)$ satisfies Assumption 2.1(c) and Assumption 2.2(b). Hence, Proposition 2.2 in [BT89] is applicable and $z(t)$ converges to $z^* = (C_s, s \in \mathcal{S})$. ■

[†]A set $\mathcal{X}$ is convex if for any x_1, $x_2 \in \mathcal{X}$ and any θ with $0 \leq \theta \leq 1$, we have $\theta x_1 + (1-\theta)x_2 \in \mathcal{X}$ [BV06]. Hence, a set consisting of a single element is always convex.

Theorem 6 *The sequence* $\{y(t)\}$ *generated by the partial asynchronous algorithm* $y(t+T) = \left(y_c(t+T) \,|\, y_c(t+T) = \sum_{s\in\mathcal{S}(c)} x_{sc}(t+T),\ \ c \in \mathcal{C}\right)$ *with* $x_{sc}(t+T)$ *defined in (5.49) and* $\lambda_c(t+\tau_{sc})$ *defined in (5.50) converges for a connected network to (5.31) and (5.32) under the assumption that the willingness-to-pay for each client is integral and* $z = (z_s \,|\, z_s = C_s, s \in \mathcal{S})$ *holds in each iteration step.*

Proof: The set of fixed points consists of the unique fixed point y^* defined in (5.31). Hence, the set is nonempty and convex. Assumption 2.1(a) and Assumption 2.2(a) hold. Furthermore, by neglecting ε and using (5.49) y_c reduces to

$$y_c(t+T) = y_c(t)\,(1-\overline{\gamma}) + \overline{\gamma} \sum_{s\in\mathcal{S}(c)} x_{sc}(t)\lambda_c(t+\tau_{sc}) \frac{C_s}{\sum_{d\in\mathcal{C}(s)} x_{sd}(t)\lambda_d(t+\tau_{sd})}. \tag{5.53}$$

With the assumption $z = \left(\sum_{d\in\mathcal{C}(s)} x_{sd}, s \in \mathcal{S}\right) = (C_s, s \in \mathcal{S})$ the fraction in (5.53) is the inverse of an average of λ weighted by x. Denote the weighted average of the price offers at server s with $\overline{\lambda}_s$ and

$$\overline{\lambda}_s = \frac{\sum_{d\in\mathcal{C}(s)} x_{sd}\lambda_d}{\sum_{d\in\mathcal{C}(s)} x_{sd}} = \frac{\sum_{d\in\mathcal{C}(s)} x_{sd}\lambda_d}{C_s}. \tag{5.54}$$

According (5.51) and with (5.54) $h(y)$ for (5.53) is

$$h(y) = (h_c(y), c \in \mathcal{C}) = \left(\sum_{s\in\mathcal{S}(c)} \frac{x_{sc}\lambda_c}{\lambda_s}, c \in \mathcal{C} \right). \tag{5.55}$$

Assumption 2.1(b) is fulfilled, because $h(y)$ in (5.55) is a continuous function.

In the following a client c is modelled as W_c distinct sub-clients. Denote the total rate of sub-client i of client c as y_c^i and the corresponding price offer is computed with $\lambda_c^i = 1/y_c^i$. If rates are initialised with $x_{sc}^i = x_{sc}/W_c$ then the total rate of client y_c is

$$y_c(t+T) = \sum_{i=1}^{W_c} y_c^i(t+T) = W_c y_c^i(t+T). \tag{5.56}$$

Hence, a client c with a willingness-to-pay of W_c behaves like W_c sub-clients and without loss of generality it is sufficient to prove the assumptions for peers with a willingness-to-pay of $W_c = 1,\ \ \forall c \in \mathcal{C}$.

For a network with a uniform willingness-to-pay the peer with the maximal total down-

load rate y_c in the network has also the minimal price offer λ_c and vice versa. Furthermore,

$$\|h(y)-y^*\|_\infty = \max_{c\in\mathcal{C}}\left|\sum_{s\in\mathcal{S}(c)}\frac{x_{sc}\lambda_c}{\bar{\lambda}_s}-y_c^*\right| \leq \|y-y^*\|_\infty \tag{5.57}$$

holds, because scaling by $\lambda_c/\bar{\lambda}_s$ will increase the rate of c if it is smaller than the rate in equilibrium, or decrease it when it is larger. By using a weighted average for scaling the total download rate will not fall into the other extreme and the inequality in (5.57) holds. Hence, $h(y)$ satisfies Assumption 2.1(c).

Furthermore, we prove Assumption 2.2(b) by contradiction. If Assumption 2.2(b) fails to hold, then for every c in $I(y,y^*) = \{c\,||y_c-y_c^*| = ||y-y^*||_\infty\}$, there exists a vector $y' \in U(y,y^*)$ with $h_c(y') = y'_c$. Equivalently,

$$\sum_{s\in\mathcal{S}(c)}\frac{x'_{sc}\lambda'_c}{\lambda'_s} = \sum_{s\in\mathcal{S}(c)} x'_{sc}. \tag{5.58}$$

Since $\lambda'_c = \lambda_c$ is the extreme value in the network the equality in (5.58) holds only if $\lambda'_c = \bar{\lambda}'_s,\ \forall s\in\mathcal{S}(c)$. However, for a connected network with $z=(C_s, s\in\mathcal{S})$ this is only true if $\lambda_d = \lambda^*,\ \forall d\in\mathcal{C}$ and the algorithm converged to the fixed point. Hence, $\|h(y)-y^*\|_\infty = 0$, which contradicts the initial condition in Assumption 2.2(b). We conclude that $h(y)$ in (5.55) satisfies Assumption 2.2(b). ■

We have proven that the system (5.49) and (5.50) converges for $0<\bar{\gamma}<1$. Furthermore, we can generalise the convergence proof also to the partial asynchronous version of (5.17) and (5.15). From (5.24) it is easily seen that the partial asynchronous algorithm of (5.17) and (5.15) converges if

$$0<\gamma_s(t+T)<\Gamma_s(t+T) = \frac{C_s}{\sum_{c\in\mathcal{C}(s)} x_{sc}(t)\lambda_c(t+\tau_{sc})}. \tag{5.59}$$

Since the variables are locally available convergence can be ensured in a distributed environment.

5.3 Evaluation

In this section we present simulation results for the proposed distributed algorithm. Starting with validating the stability condition from the last section, results for different scenarios are presented. Large P2P networks as well as dynamics in the peer population with

different churn rates are studied. The term *churn* is frequently used in the P2P literature and stands for the continuous process of the arrival and departure of peers [RGRK04]. For clarity we summarise the proposed distributed algorithm in Algorithm 1 and denote it as Resource Pricing (RP). Here, T is the time interval between consecutive updates of

Algorithm 1 Resource Pricing (RP)

CLIENT c :

$$\lambda_c(t+\tau_{sc}) = \frac{W_c}{\max\left(\eta, \frac{1}{T}\sum_{s\in\mathcal{S}(c)} x_{sc}(t-T)\tau_{cs} + x_{sc}(t)\tau_{sc}\right)} \tag{5.60}$$

SERVER s :

$$x_{sc}(t+T) = \max\left(\varepsilon, x_{sc}(t) + \gamma x_{sc}(t)\left(\lambda_c(t+\tau_{sc}) - \frac{\sum_{d\in\mathcal{C}(s)} x_{sd}(t)\lambda_d(t+\tau_{sd})}{C_s}\right)\right) \tag{5.61}$$

the sending rates and τ_{sc} and τ_{cs} the time interval between update steps at the server s and client c and vice versa.

5.3.1 Basic functionality

To validate the stability condition from Section 5.2.3 a simple example similar to the one discussed in Section 3.4.1 is studied. Assume three client peers and two server peers are present in the P2P network and clients are interested in a service, which is offered by the servers. Furthermore, clients and servers are connected according the bipartite graph in Figure 5.1, which is identical to Figure 3.4(b). Assume service rates on a connection are

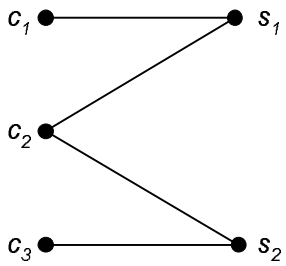
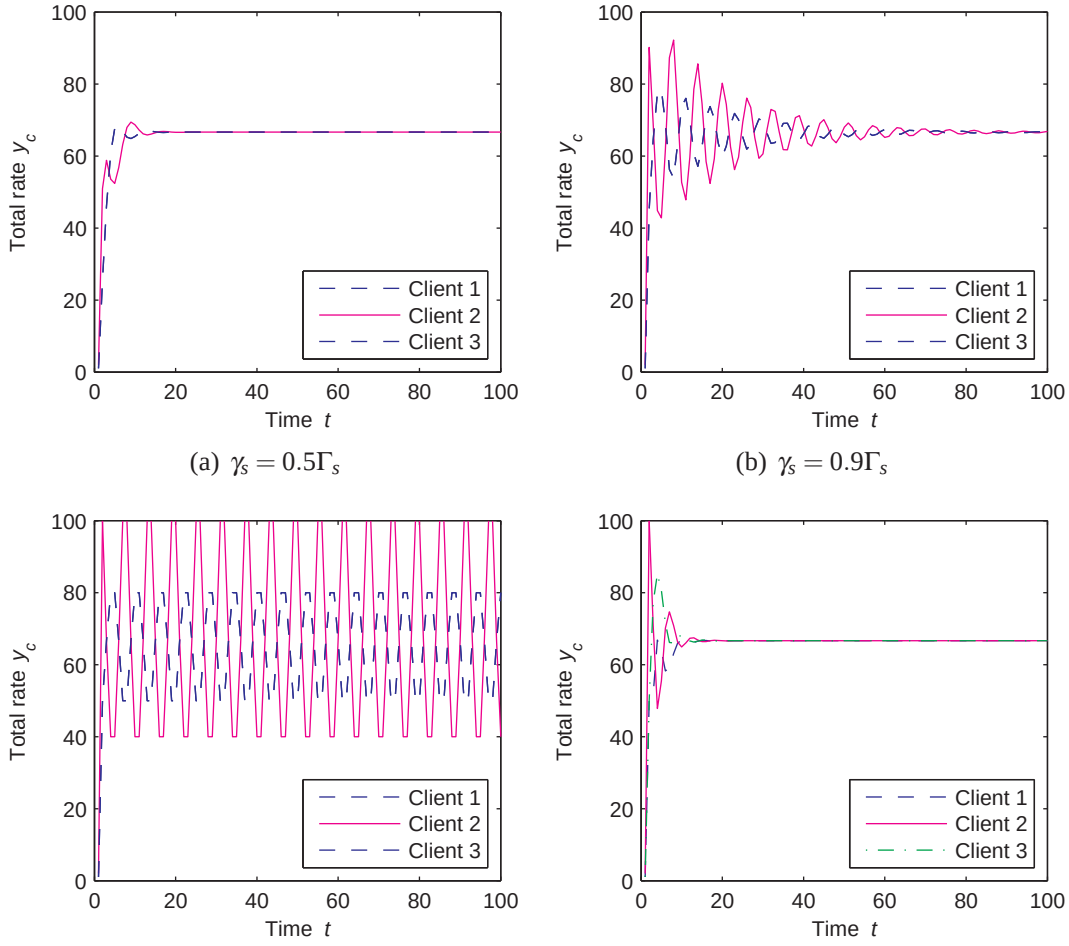


Figure 5.1: Bipartite graph of a small P2P network

set initially to $x_{sc} = 1$ and $\eta = \varepsilon = 1$ and the update interval is $T = 1$. To validate the result in (5.59) by simulation we set the capacity of the two servers and the willingness-to-pay

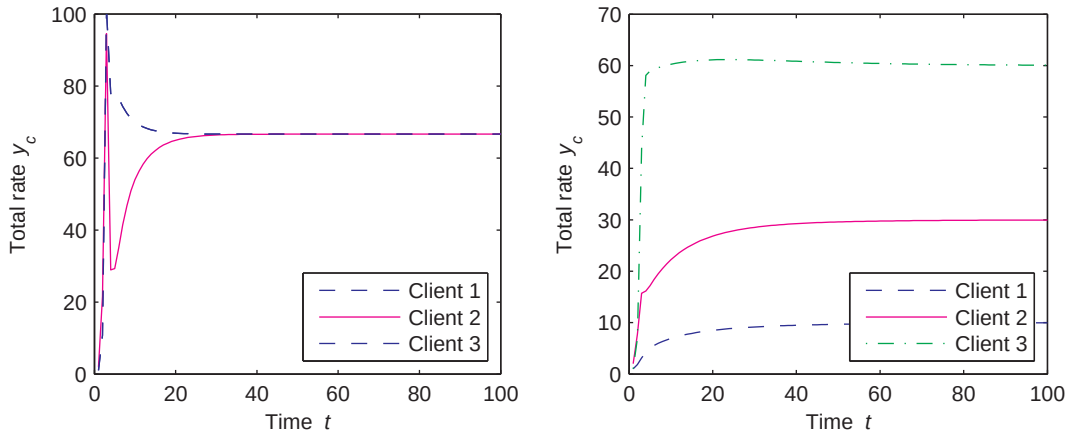


(a) $\gamma_s = 0.5\Gamma_s$ (b) $\gamma_s = 0.9\Gamma_s$

(c) $\gamma_s = \Gamma_s$ (d) $\gamma_s = \Gamma_s$ (random delays)

Figure 5.2: Validation of the stability condition in Eq. (5.59)

of all clients to $C = 100$ and $W = 100$, respectively. The total service rate y_c of the three clients is depicted in Figure 5.2 for different values of γ. Figure 5.2(a)-5.2(c) depict the results for $\tau_{sc} = T$. This means, the rates used to compute the offered prices are delayed by the maximal value of T, whereas the price offers are available at the servers immediately. As determined analytically in (5.59) the total rate of the three clients converges for $\gamma_s < \Gamma_s$ to the optimal value of $y \approx 66.7$. With an increasing value of γ the algorithm converges faster to an efficient solution, i.e. with respect to the total upload rate. It takes 22 and 8 iterations until the total upload capacity is allocated with $\gamma_s = 0.5\Gamma_s$ and $0.9\Gamma_s$, respectively. However, convergence is faster for smaller values of γ with respect to y_c. In agreement with (5.59) oscillations occur for $\gamma_s = \Gamma_s$ in Figure 5.2(c). However, as depicted in Figure 5.2(d), for which the delays between the peers are chosen as random variables between 0 and T, rates might converge to an equilibrium point also with $\gamma_s = \Gamma_s$.



(a) $W = 100, C = 100$ (b) $W = (10,30,60), C = (75,25)$

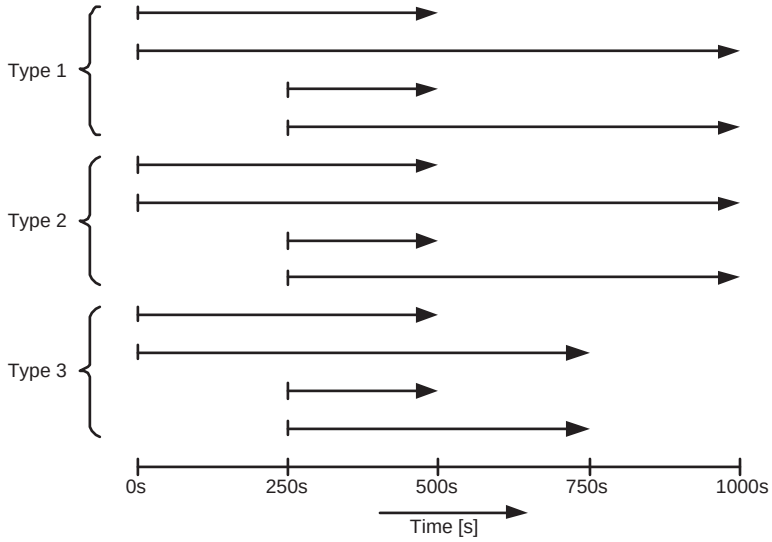
Figure 5.3: Resource Pricing with step size $\gamma = 0.1$

Further simulation results are depicted in Figure 5.3 for $\gamma = 0.1$. Also with this step size rates converge to the optimal values. For a homogeneous willingness-to-pay in Figure 5.3(a) all clients received the same total rate in equilibrium. In Figure 5.3(b) peers have different parameters. Client 1, 2 and 3 have a willingness-to-pay of 10, 30 and 60, respectively, and server 1 and 2 have a capacity of 25 and 75, respectively. Since the total capacity and the total willingness-to-pay in the network each sums up to 100 the rate for each client in Figure 5.3(b) is equal to its willingness-to-pay and agrees with (5.31). A step size of $\gamma = 0.1$ proved to be a good compromise between convergence speed and stability in all conducted simulations and is used in the following.

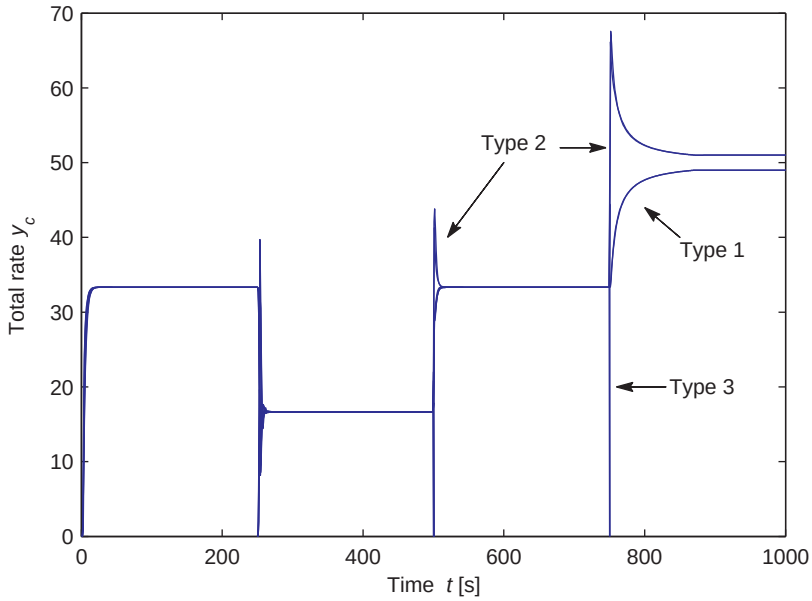
The resource pricing algorithm is studied for dynamics in the peer population in the following. Assume two servers with $C = 100$ and a varying number of clients are present in the network. Similar to the example in Figure 5.1 three types of clients are differentiated: Type 1 is connected to server s_1, Type 2 is connected to s_1 and s_2, and Type 3 is connected to s_2. Peers run the update algorithm every 1 s.

The client population is varied according Figure 5.4(a), where the start and end times of a total 12 peers are specified. Additionally, to avoid synchronous rate control a random offset between 0 and 1 seconds is added to the start time of each peer. The results of the simulation are depicted in Figure 5.4(b).

The plots of the rates for the serviced peers match well between 0 s and 750 s with each other. Although Type 2 clients make use of the available resources faster than other peers, the rates converge fast to a fair share of the total capacity for all peers in each interval. Between 0 s and 250 s six customers are active getting a rate of 33.3. When another six



(a) Start and end time of clients



(b) Rate allocation

Figure 5.4: Example with 2 servers and 12 clients

peers enter the network the rate converges fast to the fair share of 16.7 and it increases again to 33.3 when six peers leave the network at 500 s.

At 750 s all clients of Type 3 leave the network. Now, only Type 2 clients are connected to s_2 and their rates increase in the following. Since the price offers of the Type 2 clients decrease, server s_1 increases the rates for Type 1 clients. The total rates converge in the following to a fair share of 50 for all active clients. The convergence is slower in this case as compared to the previous intervals, because the rate adjustment slows down when it is near zero as it is the case for server s_1 and peers of Type 2.

5.3.2 Large-scale Simulations

For the simulations with a large and varying number of peers we model the peer behaviour as a Poisson process, where the interarrival times between peers and the session times of peers are negative exponentially distributed. Denote the mean interarrival time and mean session time as T^{arr} and T^{ses}, respectively. The average number of peers in the network can be computed with $P^{\text{ave}} = T^{ses}/T^{arr}$.

When a new peer enters the network it connects to a random subset of N_R peers. But the number of connections to other peers does not remain constant since remote peers leave the network and new peers connect to this peer. Furthermore, a peer can be disconnected from the network when all its neighbours go offline. In this case the peer obtains a new random subset in our simulation. Additionally, we assume all peers are interested in each other, except in itself.

We measure the Weighted Fairness Index (WFI) in (3.8) over the simulation time of 10000 s. A peer acts as a service provider and customer at the same time. Therefore, it adjusts its service rates as well as its price offer at the same time. The willingness-to-pay is set to the service capacity of a peer since it reflects its contribution to the network. Hence, the resource allocation is fair when the total rate of each peer is equal to its own capacity.

Two cases are considered. In the first case (denoted as *eq. C*) all peers have a capacity of $C = 100$. In the second case (*diff. C*) the capacity of a peer is an integer determined uniform randomly between 1 and 100.

The simulation results are depicted in Figure 5.5 for a mean interarrival time and mean session time of $T^{arr} = 10\,\text{s}$ and $T^{ses} = 1000\,\text{s}$, respectively, and $N_R = 10$. With this peer behaviour the average number of peers in the network is 100.

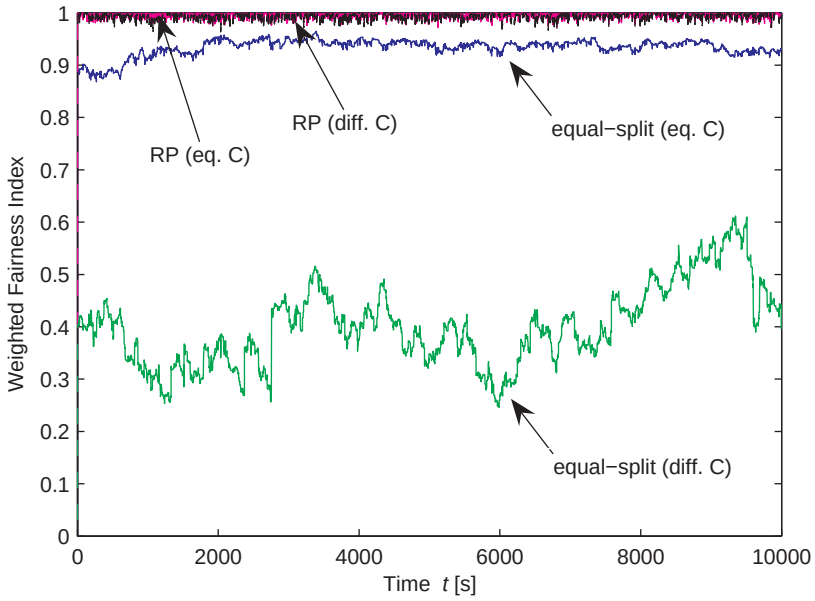


Figure 5.5: WFI for varying P2P network ($T^{arr} = 10\,\text{s}$, $T^{ses} = 1000\,\text{s}$, $N_R = 10$)

In Figure 5.5 RP denotes the resource pricing algorithm. For comparison, the results for the algorithm *equal-split* are shown, where a peer divides its capacity equally over its connections. As it can be seen from Figure 5.5 resource pricing outperforms the equal-split algorithm and is very close to the optimal fairness value of 1. The differences between both algorithms are larger when peers have different capacities. Hence, with the equal-split algorithm the allocation of resources is far from being fair contrary to resource pricing where the fairness index is over 0.95 all the time.

Also when the capacities of all the peers are equal the differences between both algorithms are clearly visible. An equal split is not as fair as resource pricing because peers maintain different numbers of connections to other peers.

Further results with resource pricing for different interarrival times and session times are summarised in Table 5.1 for peers with different capacities. Simulations were run for networks with an average population of 100 and 1000 peers each with two different pairs of interarrival and session times. Each simulation lasts for 100000 s and was replicated 10 times per parameter set. The Weighted Fairness Index is computed every second. In Table 5.1 $\overline{f_w}$ denotes the average Weighted Fairness Index over simulation time and the different runs. Furthermore, σ_{f_w} denotes the standard deviation of the measured WFI values.

In all simulations the WFI is above 0.9 indicating a fair resource allocation in the P2P network. The index decreases with a decreasing number of connections between peers

P^{ave}	T^{arr}	T^{ses}	N_R	$\overline{f_w}$	σ_{f_w}
100	1	100	5	0.90	0.028
			10	0.96	0.014
100	10	1000	5	0.96	0.037
			10	0.99	0.007
1000	1	1000	5	0.96	0.013
			10	0.99	0.004
1000	10	10000	5	0.96	0.027
			10	0.99	0.004

Table 5.1: WFI for Resource Pricing with different P2P populations

(compare the results for $N_R = 5$ with $N_R = 10$). This may be reasoned as follows. Some degree of unfairness can occur when all providers of a specific customer cannot raise the service rate, which is entitled to this customer. This case is more probable for smaller values of N_R.

Furthermore, the decrease of the WFI for highly dynamic networks is small as it can be seen from the results for an interarrival time of 1 s and session time of 100 s. Since the proposed algorithm shows a good performance in different scenarios, it is applied to a specific P2P application in the following

5.4 Bandwidth Trading in P2P CDNs

The proposed resource pricing algorithm is applicable for the allocation of resources in P2P networks, where a service is offered and requested by several peers at the same time. It is only assumed that the service is divisible and any rate allocation has a value of benefit for the corresponding peer. Hence, it can be used to allocate the upload bandwidth in file-sharing applications or the processor cycles in Grid computing architectures.

In this section we apply the algorithm to a P2P content distribution network with multi-source download (swarming). As discussed in Section 2.3.1 with multi-source download the file of interest is fragmented into pieces. Peers exchange different pieces with each other and have to complete only a single piece to contribute upload bandwidths to the network. Therefore, the resources in the P2P network are used more efficiently and the network depends not solely on the altruistic or cooperative behaviour of the peers. The swarming principle rather exploits the two-way interest of peers in different pieces, which the other one provides. Since most of the peers behave selfishly and are interested in maximising their own download rates, the mutual interest results in peers, which bargain for

bandwidth with each other. Hence, the swarming principle supports also content distribution in a non-cooperative environment and reduces the problem of free-riding [AH00].

As an example of P2P content distribution we look at the BitTorrent protocol [Coh03] where a peer uploads to others from which it receives the highest download rates. This strategy is inspired by the tit-for-tat principle that is well known from game theory. Here, a player adopts the strategy, which his opponent used in the previous round. By cooperating in the first step the tit-for-tat strategy proved very effective in the repeated prisoner's dilemma [Osb04]. Unfortunately, simulation-based studies for BitTorrent reveal a high variability in the download rates [IUKB+04] and unfairness in terms of the ratio of uploaded to downloaded data [BHP05]. This means that some peers are better off by using the tit-for-tat strategy than the others.

These results raise two questions. Firstly, from the user perspective: Does another strategy exist which outperforms BitTorrent's tit-for-tat strategy? This means with such kind of strategy a user can increase his download performance. Since the total capacity in the network is finite this happens at the expense of others. Secondly, from the angle of a protocol designer: Does a strategy exist which ensures fairness between peers although peers behave selfishly? In this section we will show that the proposed resource pricing algorithm is the answer to the second question. Additionally, we present a modification, denoted as reciprocal rate control, which uses implicit prices and compare both to BitTorrent's tit-for-tat mechanism.

Frequently, prices (virtual or monetary) are used as an incentive in non-cooperative environments. Approaches like [ACM04] and [GLBML01] study payment mechanisms for P2P file-sharing. In these works a peer gets paid for providing a specific file to a remote peer and with the so earned credits it can request a file at another peer. The discussed pricing mechanisms in this work are different from the aforementioned. With resource pricing each peer receives a virtual payment for uploading data to others. These payments control the sending rates in the future. Peers with higher price offers will receive higher rates whereas the rate will decrease for peers with lower price offers. Furthermore, the total payment a peer can afford is limited to its own upload capacity. In the modified pricing algorithm peers use their download rates from other peers as the payment information. Thus, no explicit pricing must be implemented in the protocol.

It should be noted that a trading scheme is only one of the building blocks of a P2P protocol. Since every peer starts with no piece at the beginning a good implementation has to ensure that new peers have a decent chance to finish their first piece fast. Therefore, altruistic behaviour of some peers is essential or further incentive mechanisms are needed

in the network (e.g. based on the reputation of peers).

Furthermore, the selection of pieces to download is important for the future exchange with other peers. The BitTorrent protocol implements a rarest-first strategy where each peer determines the rarest piece of its neighbours and requests this piece if available. Different papers show the efficiency of this model analytically [QS04] or by simulation [BHP05]. Furthermore, new coding schemes resolve the piece selection problem [GR05]. By exchanging linear combinations of all the pieces a peer stores, the probability that two peers have nothing of interest for the other one is reduced significantly. This section concentrates on the discussion and the detailed comparison of different trading schemes. The piece selection is incorporated into the approach in Section 5.5.

5.4.1 Trading Schemes

Trading schemes give peers an incentive to contribute bandwidth to the P2P content distribution network. Hence, peers can maximise their own download rates by appropriately allocating their upload bandwidths and free-riding is decreased and/or avoided in the network. An example is the game theoretical approach in BitTorrent. It is compared to the resource pricing algorithm in (5.60)-(5.61) as well as to a modified pricing scheme that uses implicit price information. Both pricing algorithms use rate control at the application layer. In a real implementation this can be realised by the application putting different amounts of data into the socket buffers of the TCP connections. A more sophisticated approach is to schedule the traffic directly at the queue of the network interface card, e.g. with Linux Traffic Control [HGM+]. Furthermore, a combined approach of rate control at the application and the transport layer is discussed in Chapter 6.

BitTorrent's Tit-for-Tat

The incentive mechanism in BitTorrent is a tit-for-tat strategy. It is outlined in detail in Section 2.3.1. Briefly, the important points are repeated here. In BitTorrent each peer controls to whom it uploads data. To select a peer for an upload is called unchoking. A peer uploads by default to the four remote peers from which it receives the highest download rates. This peer selection is revised every 10 seconds. When a peer has completed the download of the file, unchoking is based on the upload rate to a remote peer rather than the download rate. An exception is run every 30 seconds, where a peer is unchoked independently of its rate. This is called an optimistic unchoke.

The unchoking algorithm of BitTorrent gives peers an incentive to upload data to the network since it increases the chance to be unchoked by others with increasing upload capacity. Analytical models of BitTorrent's incentive mechanism are presented in [QS04, TC05]. In their simplified models global information is assumed, i.e. the peers know the upload bandwidth of all other peers. In this case an optimistic unchoke is needless and the peer selection achieves an equilibrium [TC05]. Peers form groups with a group size equal to the number of unchokes plus one. Furthermore, groups consist of peers with similar upload bandwidth depending on the descending order of the bandwidth (see [TC05] for details). Hence, the download rate of a peer depends not only on the upload bandwidth but also on its ratio to the bandwidth of other peers in the network. Thus, peers with the same upload bandwidth can be members in different groups and achieve different download rates. Further on, the peers with the lowest upload bandwidths form a group which can be smaller than the other groups.

Resource Pricing

The resource pricing algorithm in (5.60)-(5.61) can be used in P2P content distribution networks, where the willingness-to-pay is set to the upload bandwidth of a peer. Hence, $W_q = C_q$ for peer q and the total download rate in steady-state in (5.31) becomes

$$y_q^* = \frac{C_q \sum_{p \in \mathcal{P}} C_p}{\sum_{c \in \mathcal{C}} C_c}, \tag{5.62}$$

where $\mathcal{P}$ is the set of peers in the network and $\mathcal{C}$ is the set of peers which currently are downloading the file. This model gives peers an incentive to contribute to the network since a peer with a high upload capacity can offer higher prices than others and therefore receives a higher download rate.

Reciprocal Rate Control

The resource pricing algorithm assumes the correct signalling of the price offers. Thus, an implementation must ensure that malicious users cannot forge their price offers to obtain higher download rates. When this is not possible in a specific application, an upload rate control algorithm can only be based on the download rates of the connected peers (like in BitTorrent). This means a peer is only willing to provide upload bandwidth to another peer when it receives in return bandwidth from it.

In the context of pricing mechanisms the download rates can be interpreted as the price a remote peer is willing to pay for the upload bandwidth allocated to it. In the resource pricing algorithm the total price paid to peer p by a remote peer q for the allocation of the rate x_{pq} is the product $x_{pq}\lambda_q$. Hence, by replacing $x_{pq}\lambda_q$ with the rate x_{qp} in (5.61) we obtain the Algorithm 2[‡]. Here, the set $\mathcal{N}(p)$ denotes the set of neighbours of peer p.

Algorithm 2 Reciprocal Rate Control (RRC)

Peer p :

$$x_{pq}(t+1) = \max\left(0, x_{pq}(t) + \gamma\left(x_{qp}(t) - x_{pq}(t)\frac{\sum_{r\in\mathcal{N}(p)} x_{rp}(t)}{C_p}\right)\right) \tag{5.63}$$

Since it uses the rate as implicit price we denote it as Reciprocal Rate Control (RRC).

Similar to the results in Section 5.2.1 and 5.2.2 for resource pricing we present proofs for the efficiency and fairness of the equilibrium point of reciprocal rate control.

Theorem 7 *The equilibrium point x^* of the distributed algorithm in (5.63) finds an efficient and weighted fair resource allocation, if it is feasible.*

Proof: The equilibrium point of (5.63) is obtained if $x_{pq}(t+T) - x_{pq}(t) = 0$. Hence,

$$0 = \gamma\left(x^*_{qp} - x^*_{pq}\frac{\sum_{r\in\mathcal{N}(p)} x^*_{rp}}{C_p}\right) \tag{5.64}$$

$$\Rightarrow x^*_{pq} = \frac{C_p}{\sum_{r\in\mathcal{N}(p)} x^*_{rp}} x^*_{qp} \tag{5.65}$$

and by summing up over all upload rates of peer p

$$\sum_{q\in\mathcal{N}(p)} x^*_{pq} = C_p. \tag{5.66}$$

Equation (5.66) implies that in equilibrium the total upload rate of a peer is equal its upload capacity. With (5.65) we can compute the total download rate y^*_p of peer p in

[‡]For the sake of simplicity we omit to state delays in the algorithm. Similar to Algorithm 1 delays can be taken into account. The presented results in Theorem 7 and Theorem 8 hold also for the delayed version of Algorithm 2.

equilibrium by

$$y_p^* = \sum_{q \in \mathcal{N}(p)} x_{qp}^* = C_p \frac{x_{qp}^*}{x_{pq}^*}, \quad \forall q \in \mathcal{N}(p). \tag{5.67}$$

Equation (5.67) holds for all neighbours of peer p. Hence, the fraction of total download rate to upload capacity is equal for two peers, if they have a common neighbour. Hence, $y_p^*/C_p = y_q^*/C_q$ holds for peers p and q, if a peer r exists with $r \in \mathcal{N}(p) \wedge r \in \mathcal{N}(q)$. Furthermore, for $r \in \mathcal{N}(p)$ also $\frac{y_p^*}{C_p} = \frac{C_r}{y_r^*}$ holds.
Similar to the derivation of the optimum for the parallel bottleneck model in Section 3.4 a bipartite graph can be composed of the sets of uploading and downloading peers, which are identical in this case. An edge denotes a bandwidth allocation greater zero. If the bipartite graph is connected, it holds

$$x_{qp}^* = x_{pq}^*, \quad \forall q \in \mathcal{N}(p) \tag{5.68}$$

and

$$y_p^* = \sum_{q \in \mathcal{N}(p)} x_{qp}^* = C_p. \tag{5.69}$$

Thus, in equilibrium a peer uses its full upload bandwidth (see (5.66)) and downloads from a specific peer with the same rate as it is uploading (see (5.68)). Furthermore, the total download rate over all connections is equal to the upload capacity of the peer (see (5.69)). ■

Additionally, we show in Theorem 8 that the reciprocal rate control algorithm fully utilises the upload capacity under the assumption that in the previous step all resources are used.

Theorem 8 *If $\sum_{q \in \mathcal{N}(p)} x_{pq}(t) = C_p$ holds, the distributed algorithm in (5.63) is efficient in all succeeding update steps.*

Proof: By summing up over all upload rates of peer p

$$\sum_{q \in \mathcal{N}(p)} x_{qp}(t+1) = \sum_{q \in \mathcal{N}(p)} \left(x_{pq}(t) + \gamma \left(x_{qp}(t) - x_{pq}(t) \frac{\sum_{r \in \mathcal{N}(p)} x_{rp}(t)}{C_p} \right) \right) \tag{5.70}$$

$$= C_p + \gamma \left(\sum_{q \in \mathcal{N}(p)} x_{qp}(t) - C_p \frac{\sum_{r \in \mathcal{N}(p)} x_{rp}(t)}{C_p} \right) \tag{5.71}$$

$$= C_p \tag{5.72}$$

This concludes the proof of efficiency. ■

Thus, when we start with an initial rate allocation which is efficient all following rate allocations are efficient as well. The reciprocal rate control algorithm is restricted to peers that did not complete the download already. If feasible, it ensures that a peer receives at least the rate, which it provides to the network. Seeds cannot run the algorithm and have to use other rules for their upload. (E.g. these peers can upload to peers, which have nothing at all.) Therefore, an advantage of explicit prices used by resource pricing as compared to implicit prices is that fairness is preserved also over the altruistically contributed resources in the network.

The authors of [ALTA06] describe in their paper an algorithm that is similar to reciprocal rate control. Both algorithms converge to a pair-wise fair bandwidth allocation, where rates between peers are symmetric. However, [ALTA06] describes a P2P network, where users access remotely data from their home computer as well as from other peers. Thus, self allocations are possible in their system, but not in BitTorrent-like networks.

Furthermore, [YdV06] discusses pair-wise allocations for BitTorrent-like networks. They prove analytically that networks need not to be fully-connected to achieve symmetric allocations and therefore fairness. However, no algorithm is presented in [YdV06], which finds these allocations. Since reciprocal rate control ensures fairness and provides a symmetric bandwidth allocation in equilibrium, it represents an algorithm for this purpose.

5.4.2 Nash Equilibrium

Most peers in a P2P network behave selfishly and try to maximise their performance only. Therefore, we can interpret the trading of bandwidth as a non-cooperative game and study the Nash equilibrium. In the Nash equilibrium no peer gains by only changing its strategy while all other players keep their strategy unchanged. This means a Nash equilibrium corresponds to a set of strategies where every player has found the best response to the actions of the other players [Osb04].

The Nash equilibrium for the BitTorrent protocol is studied in [QS04]. Here, each peer can choose its upload bandwidth between zero and its physical upload capacity. Since a higher upload bandwidth is associated with a higher cost, a peer is interested primarily in maximising its download rate while also minimising its cost at the same time. [QS04] shows that a Nash equilibrium exists for networks consisting of groups (with a group size larger than the number of unchokes plus one) of peers where peers belonging to the same group have identical upload bandwidths. In this situation the best case for a

peer is to upload to other peers in its group. The threshold to be unchoked is exactly the upload bandwidth divided by the number of unchokes (which is assumed to be equal for all peers). Therefore, a peer will be choked by the peers of its group when it reduces its upload bandwidth. This results in inferior performance.

On the other hand a Nash equilibrium does not exist when peers have different upload capacities as it is demonstrated by an example in [QS04]. In this case some peers can decrease their upload bandwidths and are still unchoked by the same remote peers. Additionally, these peers can increase the number of peers to unchoke and are possibly unchoked by further peers in return. Hence, the strategies of peers are diverse and not only restricted by choosing an upload bandwidth. This complicates the discussion of the Nash equilibrium for BitTorrent.

In the following we study the Nash equilibria for the two proposed pricing schemes in steady-state. Assume for resource pricing that a peer cannot forge its price offer λ. It can be seen directly from (5.62) that any reduction of upload capacity will result in a decrease of the download rate. When maximising the download rate has priority over minimising the upload cost every peer will upload with its physical upload capacity. Thus, the steady-state of the system (5.60)-(5.61) is the Nash equilibrium when the willingness-to-pay is set to the upload capacity.

Similarly, the existence of the Nash equilibrium for reciprocal rate control can be shown for the steady-state with (5.68) and (5.69). Since a peer receives exactly the rate, which it provides to the other peer, it gains nothing by reducing its upload bandwidth. Hence, in steady-state its total download rate is its own upload capacity.

5.4.3 Performance Evaluation

The discussed trading schemes for P2P content distribution are evaluated by simulations in the following. We build on the discrete event simulator from Section 5.3 and restrict our analysis to the application layer. It is widely assumed that the bottlenecks in P2P networks are the access links of the users [BHP05, QS04]. This seems reasonable because most peers are home users who are connected e.g. with DSL or cable modems to the Internet. Furthermore, the core of the Internet shows a low utilisation of the available bandwidths [Odl03] and small packet loss rates due to congestion [Flo07].

Additionally, we omit the TCP behaviour in this section and discuss in general the interaction between P2P and the transport protocol in Chapter 6. Furthermore, we present in [EHBK07] a simulation analysis of the differences between packet-level and flow-level

simulations for BitTorrent. Since home users often have asymmetric access links (e.g. ADSL) and/or set the upload bandwidth in the P2P application lower than their physical upload capacity, we assume the uplink is the bottleneck in the whole network.

The bandwidth trading schemes depend on the number of neighbours and especially on the number of connections where both sides are interested in data of the other side. Thus, the topology generated by the protocol and the implemented piece selection algorithm influence the results of bandwidth trading. To concentrate on the trading schemes in the simulations we omit the piece selection algorithm for now and assume connected peers are always interested in pieces of each other. Furthermore, for BitTorrent anti-snubbing is neglected, because it can result in situations where a peer does not contribute its upload bandwidth although it can transfer data to other peers. This can cause inefficiency in the network.

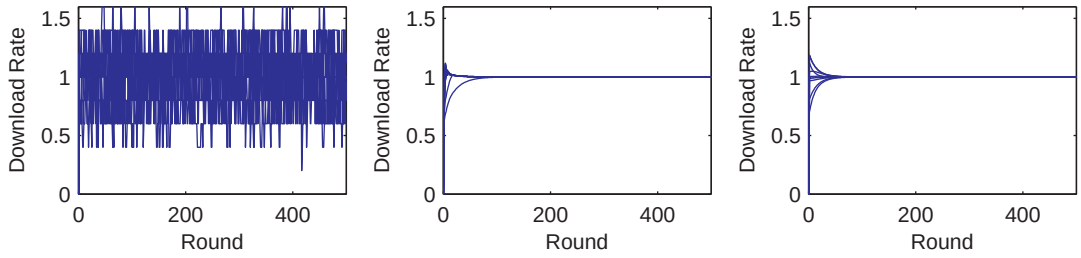
The overlay topology is constructed in the simulations according to the original BitTorrent implementation [Coh03, BTa]. Here, a peer asks a so-called tracker, a centralised component that stores information about all peers, for a list of other peers in the network. The tracker returns a random subset of length N_R to the requesting peer. Hence, a peer opens a new connection to a remote peer based on the list returned by the tracker or when a remote peer asks for it.

We will discuss static and dynamic P2P networks. As outlined in Appendix A the expected number of connections of a peer is $2N_R$ and N_R in the static and the dynamic simulations, respectively. This difference has to be taken into account when results are compared with each other.

Static networks

One major difference of the two pricing schemes to BitTorrent's tit-for-tat strategy is their convergence to a steady-state. To demonstrate the basic functionality of all algorithms we run a simulation for a small network of 10 peers with homogeneous upload capacities of $C = 1$. The superposition of the download rates of the peers is depicted in Figure 5.6. Whereas the download rates of the peers oscillate between 0.4 and 1.6 for BitTorrent, the rates converge fast to a fair allocation for the Resource Pricing (RP) and the Reciprocal Rate Control (RRC) algorithm where each peer receives exactly its upload capacity.

The oscillations with BitTorrent are caused by the optimistic unchoke. For the ideal case where peers have global information about the upload bandwidth of other peers analytical studies [QS04, TC05] show that the tit-for-tat strategy has an equilibrium point. Since the



(a) BitTorrent (b) Resource Pricing (c) RRC

Figure 5.6: Superposition of download rates for 10 peers

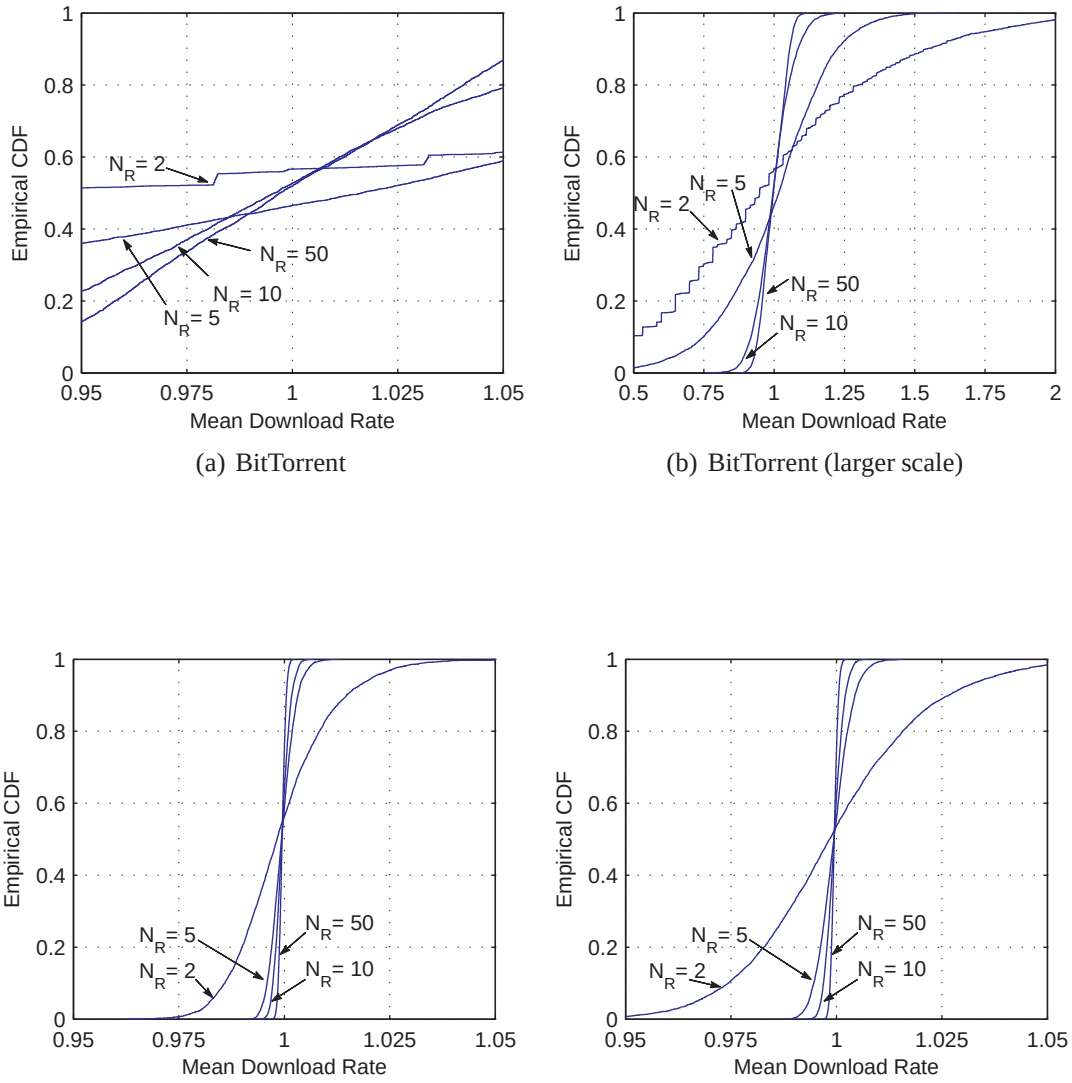
optimistic unchoke is a random process the mean values of the download rates are closer together. In this example the mean download rates of the 10 peers lie between 0.77 and 1.23.

Especially, the performance of BitTorrent depends on the number of neighbours. To study the influence of the topology on the download performance simulations are run with varying parameter N_R. The network population was 1000 peers, each peer starting with a small random offset to avoid synchronisation effects. The simulation consisted of 1000 iterations of the respective algorithm and was repeated 5 times. We set $\gamma = 0.1$ for RP and RRC and $\varepsilon = 0.1C/N_C$, whereby N_C denotes the number of connections of a peer. The capacity of all peers is $C = 1$.

The empirical Cumulative Distribution Function (CDF) of the mean download rate over the 1000 iterations is depicted over all runs in Figure 5.7. To compare the three algorithms Figure 5.7(a), 5.7(c) and 5.7(d) show the results for download rates between 0.95 and 1.05. With BitTorrent's tit-for-tat algorithm peers receive highly different download rates while providing the same upload capacity. To see its performance on a larger scale the CDF for download rates between 0.5 to 2 is depicted in Figure 5.7(b).

Tit-for-tat depends on the number of neighbours. With $N_R = 2$ a peer has on average four connections only and therefore no choice in selecting peers. This results in a poor performance where 10% of the peers receive half or less of what they provide. Download rates are distributed more equitably with increasing N_R. For $N_R = 10$ all peers receive a mean download rate of $\pm 20\%$ of their upload capacity, whereas this percentage decreases to $\pm 10\%$ for $N_R = 50$.

Nonetheless, the distribution of download rates with BitTorrent is not as uniformly distributed as with the two proposed pricing algorithms. Nearly all peers receive download rates between 0.98 and 1.02 with both algorithms and values of $N_R > 2$. Furthermore, the download rates at the end of each simulation were equal to one for every peer. This means



(a) BitTorrent

(b) BitTorrent (larger scale)

(c) Resource Pricing

(d) Reciprocal Rate Control

Figure 5.7: Empirical CDF for varying parameter N_R

the pricing algorithms converged. Thus, the differences of the CDFs for different values of N_R in Figure 5.7(c) and 5.7(d) are due to differences in the rate of convergence. Because of the slight differences observed in the results we analyse the convergence properties for the case of heterogeneous peers.

For the next simulation we consider heterogeneous peers with upload capacities of integers between 1 and 10. The capacity is determined randomly at the start of the simulation. With a total of 1000 peers there are on average 100 peers with the same capacity in the network. The download performance of each peer is measured by the ratio of mean download rate to upload capacity. The CDFs of the download performances are depicted in Figure 5.8(a), 5.9(a) and 5.10(a) for the different algorithms and a download performance between 0.9 and 1.1.

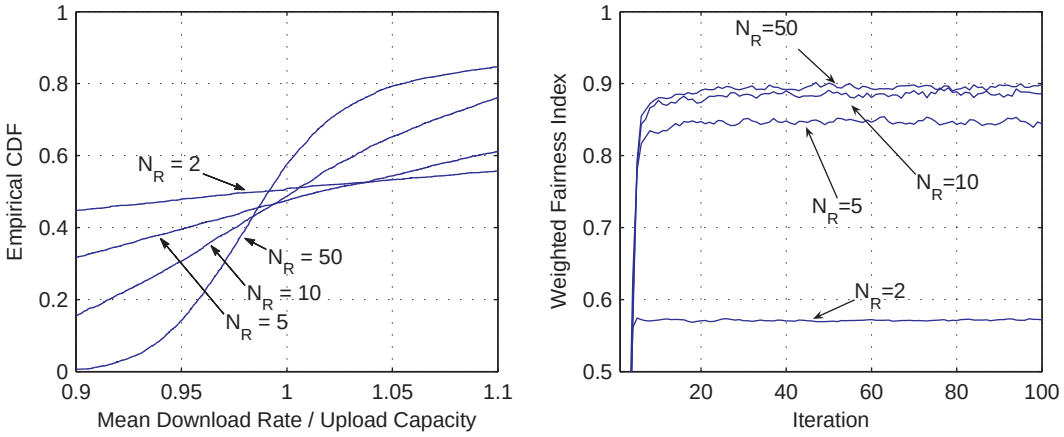
As expected the download performances of the peers vary more as compared to the simulation results with homogeneous peers. The CDF for BitTorrent in Figure 5.8(a) shows only with $N_R = 50$ a good performance, because only around 1% of the peers receive a download to upload ratio of less than 0.9. But this proportion rises to about 15% and 31% for $N_R = 10$ and for $N_R = 5$, respectively.

The CDFs for the two pricing schemes in Figure 5.9(a) and 5.10(a) outperform BitTorrent with respect to fairness. The ratio of the mean download rate to the upload capacity is over 0.98 for all peers for $N_R > 2$. Only for $N_R = 2$ fairness deteriorates and around 12% and 20% of the peers have a ratio of less than 0.95 using resource pricing and reciprocal rate control, respectively.

To study the convergence of the algorithms we measured the Weighted Fairness Index defined in (3.8) over simulation time. Furthermore, we associate the upload capacity of a peer with its weight. The results in Figure 5.9(b) and 5.10(b) show for $N_R > 2$ the convergence to a fair resource allocation for both proposed algorithms in less than 70 iterations. The convergence is better with increasing N_R and better for resource pricing than with reciprocal rate control. When $N_R = 2$ we observed no convergence even for a larger number of iterations. In this case some peers receive an unfair download rate because its small neighbourhood cannot raise the adequate rate with their upload capacity.

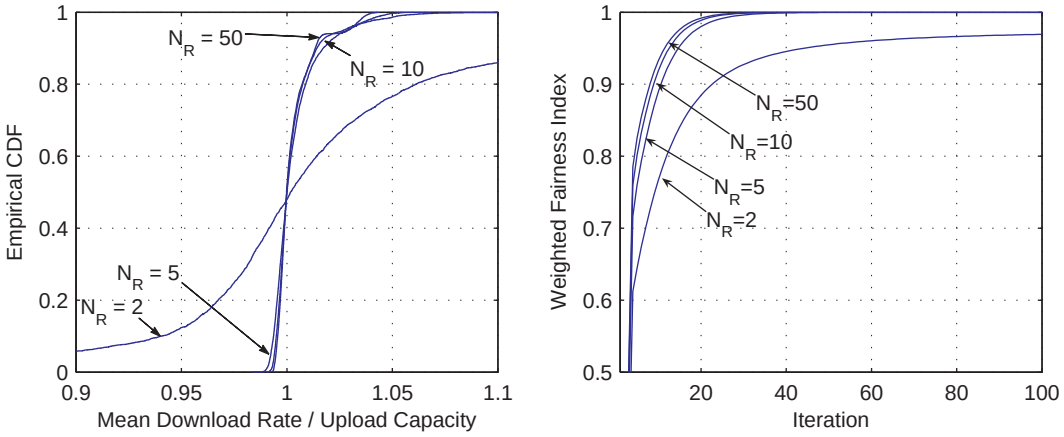
The weighted fairness index of BitTorrent in Figure 5.8(b) is below 0.9 for all studied values of N_R. (It should be noted that a fairness index of 0.9 corresponds for example to an allocation where 10% of the peers receive nothing at all and the remaining 90% receive equal rates.)

We also measured the download performance of the peers depending on their upload capacity. Figure 5.11 shows the median and the 2.5 to 97.5 percentile range of the mean



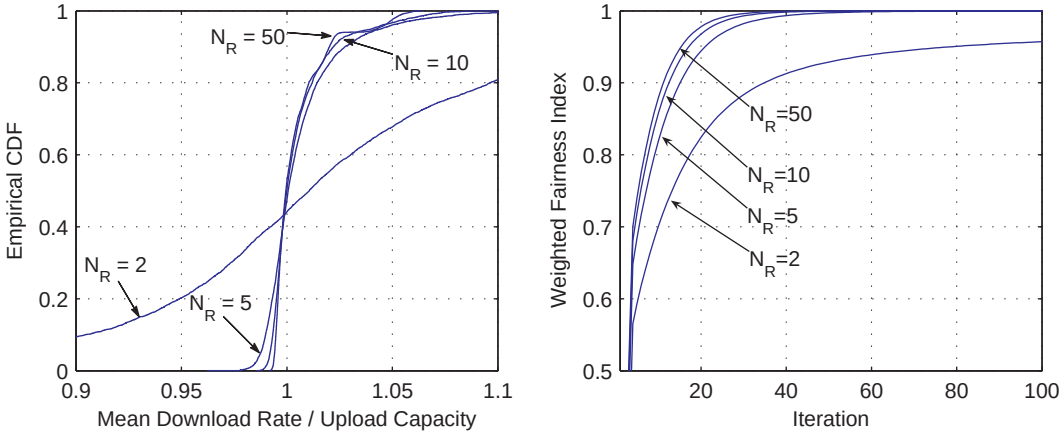
(a) CDF (b) WFI

Figure 5.8: BitTorrent (heterogeneous peers)



(a) CDF (b) WFI

Figure 5.9: Resource Pricing (heterogeneous peers)



(a) CDF (b) WFI

Figure 5.10: Reciprocal Rate Control (heterogeneous peers)

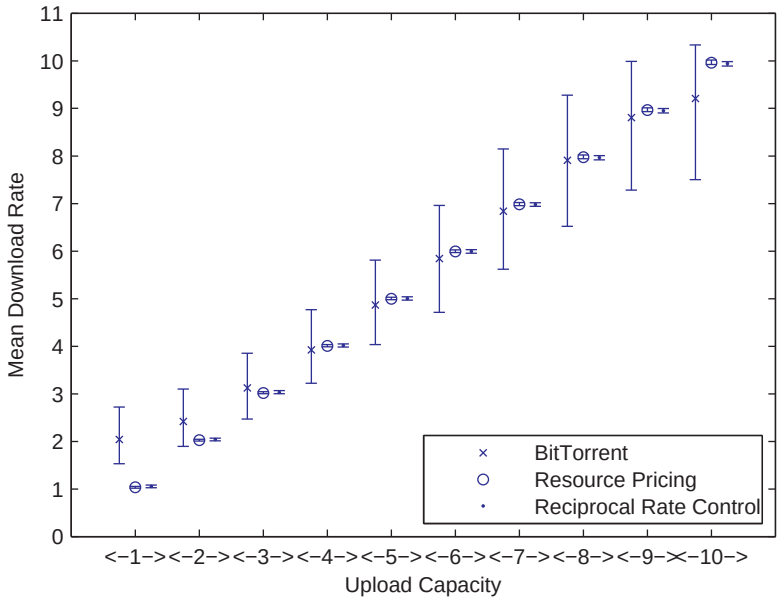


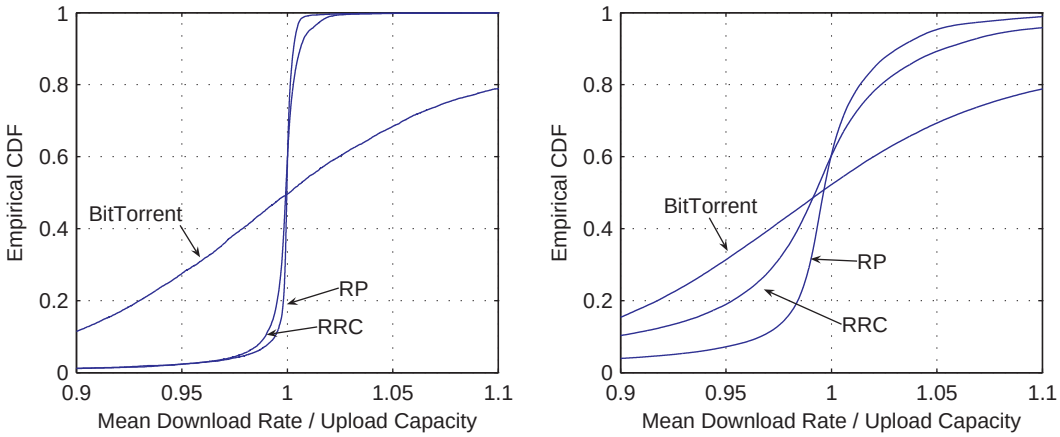
Figure 5.11: Median and the 2.5-97.5 percentile range of the mean download rates for heterogeneous peers ($N_R = 10$)

download rates for peers with the same upload capacity. By using BitTorrent the median for peers with an upload capacity of $C = 1$ is much larger as the fair share of one whereas the peers with $C = 10$ receive mostly a rate lower than their contribution. One reason for this is the optimistic unchoke since a low capacity peer gains more than it invests compared to a high capacity peer. Another reason is the incapability to find the right match which also explains the different performances between peers with the same capacity.

For resource pricing and reciprocal rate control the median of the download performance is very close to the upload capacities of the peers. Additionally, the small percentile range indicates that peers with the same upload capacity receive nearly the same download performance.

Dynamic networks

We conducted simulations also for dynamic networks where interarrival times between peers and session times of peers are negative exponentially distributed random variables. We set $N_R = 10$ and compare the performance for two cases with the same average peer population of 1000. We assume the respective algorithm is run per unit time. The resulting CDFs are depicted in Figure 5.12(a) and 5.12(b). With a mean interarrival time between peers of $T^{arr} = 10$, i.e. on average 10 iterations of the algorithm before a new peer enters,



(a) $T^{arr} = 10, T^{ses} = 10000$ (b) $T^{arr} = 1, T^{ses} = 1000$

Figure 5.12: CDFs for dynamic networks ($N_R = 10$)

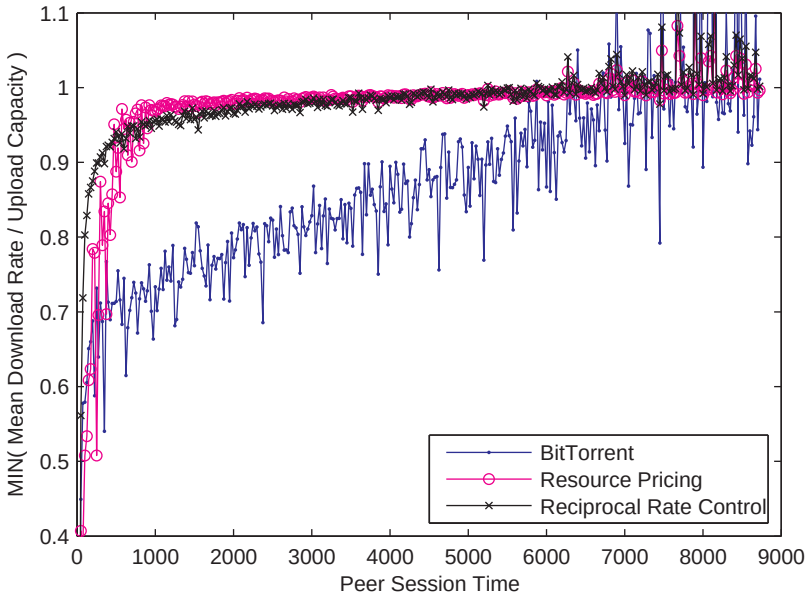


Figure 5.13: Minimal download performance ($T^{arr} = 1, T^{ses} = 1000$)

N_R	T^{arr}	T^{ses}	BitTorrent	Resource Pricing	Reciprocal Rate Control
4	1	1000	0.5644	0.9272	0.9021
	10	10000	0.5774	0.9571	0.9526
10	1	1000	0.8415	0.9946	0.9872
	10	10000	0.85	0.999	0.9985
20	1	1000	0.8872	0.9971	0.9916
	10	10000	0.8924	0.9996	0.9992
100	1	1000	0.8959	0.9981	0.9935
	10	10000	0.9008	0.9998	0.9993

Table 5.2: Mean of WFI in dynamic networks

the two pricing schemes achieve nearly uniformly distributed download performances for heterogeneous peers.

For $T^{arr} = 1$ in Figure 5.12(b) the CDF shows higher variability in the achieved download performance for the two proposed algorithms. Furthermore, the results for reciprocal rate control deteriorate more than for resource pricing comparing it with the results in Figure 5.12(a).

Especially for $T^{arr} = 1, T^{ses} = 1000$ some peers stay only a very short time in the network and merely run the trading algorithm a few times. This results in a poor download performance as it can be seen from Figure 5.13. Here, the minima of the download performances of peers with similar session times are measured for time intervals of 25 time units. For all schemes the minimal download performance increases with the session time of the peers, but the increase is more pronounced for the two trading schemes. After around 500 time units the minimal download performance in the network is above 0.9.

As discussed in Appendix A the average number of connections for the same value of N_R is different in static and dynamic simulations. To ensure a fair comparison between both cases we run simulations for dynamic networks with $N_R = 4, 10, 20, 100$. Thus, the average number of connections is equal to the static simulations with $N_R = 2, 5, 10, 50$, respectively. Table 5.2 summarises the results for the mean value of the Weighted Fairness Index. We measured the Weighted Fairness Index in the P2P network per time unit and averaged it over a simulation time of 100000 s. By comparing the results of BitTorrent in Table 5.2 with the values of the static simulations in Figure 5.8(b) we see that the results with the same average number of connections agree with each other. Thus, for the conducted simulations the dynamics of the peers hardly influence the fairness in the network.

Also the results for the two proposed algorithms in the dynamic case are similar to the static case. With an average of 10 or more connections the Weighted Fairness Index is close to the optimal value of one. The differences between resource pricing and reciprocal rate control are small, especially for larger values of N_R, but resource pricing produces slightly better results throughout all simulations. With higher churn rates (i.e. for $T^{arr} = 1$, $T^{ses} = 1000$) the Weighted Fairness Index decreases for all three algorithms with $N_R = 4$. For larger values of N_R the differences to the simulations with $T^{arr} = 10$, $T^{ses} = 10000$ are small. Also the results for reciprocal rate control with $N_R = 10$ are close together for both runs. This does not disagree with the results in Figure 5.12(b), because peers with short session times contribute less to the Weighted Fairness Index as compared to peers that stay longer in the network.

5.4.4 Conclusion

BitTorrent's tit-for-tat strategy gives peers an incentive to contribute upload bandwidth to the network. But it cannot avoid unfairness between peers with respect to the experienced download performance. Especially, when peers have a small number of connections, download rates vary considerably. Furthermore, peers with small upload capacities compared to others receive considerably more than what they contribute to the network.

The two proposed alternatives improve fairness. Both use price information to control the upload rates at the application layer. We derive that the proposed distributed algorithms achieve in equilibrium a fair and efficient allocation of the total upload bandwidth contributed by the peers in the network. Furthermore, the steady-states of the two schemes provide Nash equilibria. In static and dynamic simulations both algorithms show good convergence and better fairness as the tit-for-tat strategy used in BitTorrent.

5.5 Resource Pricing and Piece Selection

In Section 5.4 we assume peers have always parts of the file which are of interest for the remote peers. This simplification enables to study the incentive schemes without discussing a specific piece selection algorithm. It is also applicable to systems with network coding [GR05] where linear combinations of all pieces are exchanged and no piece selection is run. When network coding is not used, piece and peer selection depend on each other. Besides finding the peers, which provide good download rates, a peer should download rare pieces to ensure that it has something of interest for others in the future.

Furthermore, requesting rare pieces increases the availability of a full copy of the file in the network.

The optimisation problem in (5.1)-(5.4) can be extended to a multi-objective optimisation problem, where beside rates also the content availability is maximised. An optimisation problem for the content availability for BitTorrent-like networks is presented in [CXH06]. However, the authors of [CXH06] prove that the optimisation problem is NP-complete. Hence, no algorithm finds the optimum of the problem in polynomial time unless the well-known $P = NP$ problem in complexity theory is resolved [FH03].

On the other hand analytical and simulation-based studies [QS04, BHP05] show that BitTorrent's piece selection is efficient. Thus, we discuss in the following how resource pricing (Algorithm 1, p. 65) can be combined with BitTorrent's piece selection. We choose resource pricing instead of reciprocal rate control, because it has two advantages. It ensures fairness also for the resources contributed by the seeds and has higher flexibility by defining the willingness-to-pay appropriately. For example, when the content distribution is realised by proprietary software of the content providers, they may set the willingness-to-pay for each peer according to the subscription status of the user. Furthermore, when the content owner contributes also resources to the P2P network the owner has influence how these are allocated to the peers. Additionally, proprietary software (e.g. for set-top boxes) and the combination of P2P content distribution with IP multimedia subsystems as discussed in [LL07] have the potential to reduce the influence of malicious peers.

Assume we replace BitTorrent's peer selection by Algorithm 1 but keep the other functionalities of BitTorrent. This means a peer controls the upload rate to all its neighbours with resource pricing and peers request pieces according to the rules described in Section 2.3.1[§]. We evaluate this new approach and compare it to the original BitTorrent implementation by simulations at the application layer.

In the first simulation scenario a constant number P of peers is present in the network. Further on, S and L denote the number of seeds and leechers in the network, respectively. Hence, $P = S + L$. To keep the number of seeds and leechers constant in our simulation we assume a leecher leaves the network when its download is completed and a new leecher with no pieces enters the network at this point in time. At the beginning of the simulation the leechers have a random set of pieces. This simulation setup studies the performance of peers with different download progress. Furthermore, it can be easily studied analytically.

[§]For BitTorrent as well as the revised version with resource pricing the endgame-mode is neglected. We omitted the endgame mode in our implementation because it is not clearly specified when a peer switches to the endgame mode. Hence, different implementations realise it differently.

The total capacity in the network is $\sum_{s=1}^{S} C_s + \sum_{l=1}^{L} C_l$, where C denotes the upload capacity of a peer. To compare resource pricing with BitTorrent we set the willingness-to-pay to the upload capacity for each leecher. Hence, the objective of resource pricing as well as BitTorrent's tit-for-tat is a proportional relationship between upload capacity and download rate. This objective gives peers an incentive to contribute bandwidth to the network. In the optimal case the download rate achieved by trading bandwidth with other leechers is equal to the peer's own upload capacity. Another portion of the download rate is contributed by the seeds in the network. If the resources of the seeds are allocated proportional fair with respect to the leecher's upload bandwidth a leecher l has a total download rate

$$y_l = C_l + C_l \frac{\sum_{s \in S} C_s}{\sum_{m \in L} C_m} = C_l \frac{\sum_{p \in P} C_p}{\sum_{m \in L} C_m}. \tag{5.73}$$

Equation (5.73) overestimates the download performance of a peer because it neglects that a peer needs a piece which is of interest for other peers. This is not always the case. Particularly, at the start a peer has no data and cannot contribute its resources to the network. Furthermore, (5.73) is identical to the steady-state download rate of resource pricing in (5.62) because resource pricing is designed to ensure fairness.

As we have seen in the previous sections the number of neighbours has an influence on the download performance. In BitTorrent several parameters control the number of neighbours. So far we have taken only N_R, the number of peers returned by the tracker, into account. Now, we will introduce also the parameters *max initiate* and *min peers*, which are used in BitTorrent [BTa]. Denote *max initiate* with N_{MI} and define it as the number of connections at which the peer stops initiating new connections. Furthermore, denote *min peers* with N_{MP} and define it as the lower limit of connections at which a peer does not re-request new peer information from the tracker. By default, a BitTorrent peer requests $N_R = 50$ addresses of peers at the tracker and opens if possible up to $N_{MI} = 40$ connections to other peers. Due to churn the number of connections changes over time. It can increase above $N_{MI} = 40$ connections when remote peers ask for a connection because remote requests are not denied in the implementation. When many peers leave the network and the number of connections is N_{MP} or below the peer concerned asks the tracker for additional peer information. To avoid frequent re-requests a peer has to wait at least 300 s between two requests to the tracker.

In the following we will not present a detailed parameter study but use the default values for BitTorrent. For resource pricing we choose the parameters such that peers in the network have approximately the same number of neighbours and do not run out of neighbours. Hence, N_R is set to a large value and $N_{MI} = N_{MP}$. Additionally, we run simulations

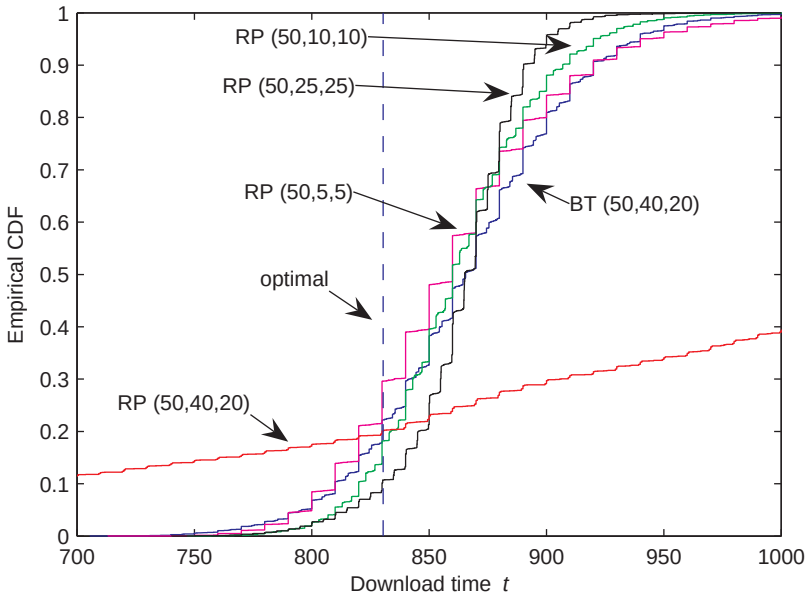
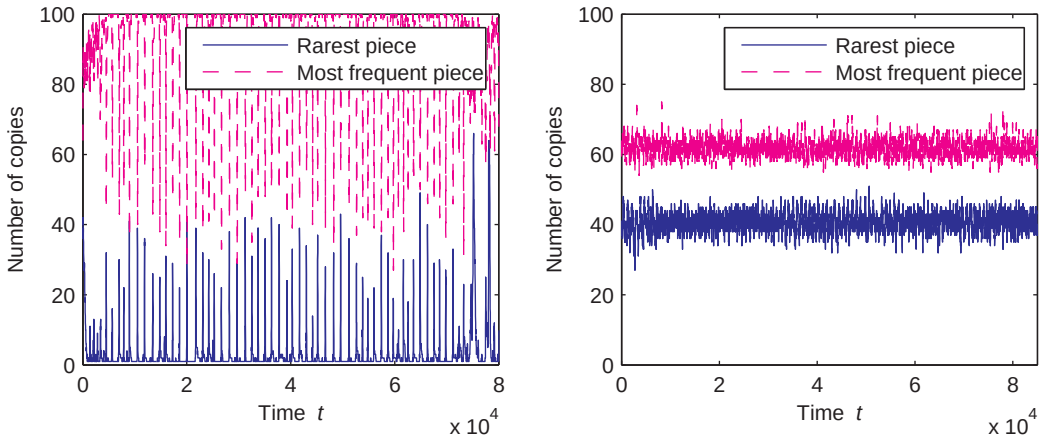


Figure 5.14: CDF of the download time for Resource Pricing (RP) and BitTorrent (BT) with different parameters (N_R,N_{MI},N_{MP})

with the default parameters of BitTorrent also for resource pricing.

We simulate a network of $P = 100$ peers, which download a file of size $S_F = 100\,\text{MB}$. Pieces have a size of 256 KB. With BitTorrent each peer unchokes 4 other peers plus the optimistic unchoke. Additionally, all other parameters are set to the default values of BitTorrent [BTa]. For $S = 1$ and $L = 99$ and a homogeneous upload capacity of $C = 1\,\text{Mbps}$ the optimal download time t is $t = S_F/y_l \approx 830\,\text{s}$ using (5.73). Figure 5.14 compares the optimal download time with BitTorrent using default parameters and resource pricing with varying parameters N_{MI} and N_{MP}.

The most striking result is the poor performance with resource pricing and parameters $N_R = 50$, $N_{MI} = 40$ and $N_{MP} = 20$, which will be discussed in detail shortly. The other simulation results are close to the optimal value. The mean download time with BitTorrent is 866 s and for resource pricing with $N_{MI} = N_{MP} = 5$, $N_{MI} = N_{MP} = 10$ and $N_{MI} = N_{MP} = 25$ around 861 s, 863 s and 864 s, respectively. Looking at the distribution 90% of the peers have a download time between 800-940 s, 810-920 s and 815-900 s with resource pricing and $N_{MI} = N_{MP} = 5$, $N_{MI} = N_{MP} = 10$ and $N_{MI} = N_{MP} = 25$, respectively. BitTorrent shows a range of 796-940 s. Hence, fairness between peers increases for resource pricing with an increasing value for the parameters N_{MI} and N_{MP} up to a limit of 25. Although differences are small for homogeneous peers fairness is better for these cases as with BitTorrent.



(a) Resource Pricing (50,40,20) (b) Resource Pricing (50,10,10)

Figure 5.15: Number of copies of rarest and most frequent piece in the network for parameters (N_R, N_{MI}, N_{MP})

As opposed to BitTorrent, where peers upload by default to five other peers, resource pricing uploads data over all the connections of a peer. This can cause inefficiency as seen in Figure 5.14 for $N_R = 50$, $N_{MI} = 40$ and $N_{MP} = 20$. By requesting data from a large number of other peers the piece selection is not efficient and rare pieces emerge. This can be seen from Figure 5.15, where the number of copies for the rarest and the most frequent piece is depicted for the inefficient case ($N_R = 50, N_{MI} = 40, N_{MP} = 20$) in comparison to an efficient case ($N_R = 50, N_{MI} = 10, N_{MP} = 10$).

In Figure 5.15(a) the rarest piece is sometimes only available at the seed. Although the number of copies increases fast in the following another piece becomes rare. The piece availability in the network is unstable and peers have highly different download times. This is different in Figure 5.15(b). Here, the piece selection works fine and the network is time invariant. This problem is not specific to resource pricing. Also BitTorrent shows similar behaviour when peers unchoke a larger number of remote peers (e.g. with 20 unchokes for the discussed simulation scenario). The reason for inefficiency lies in the time it takes until a full chunk is uploaded. With many upload connections the rate per connection decreases and it takes longer to complete a specific chunk. Hence, when a chunk becomes rare it takes longer with many connections until further peers offer copies of it. Furthermore, the piece availability depends on the number of peers in the network. For this simulation setup all pieces are distributed randomly at the beginning. Hence, on average each piece is available at half of the peers in the network. With an increasing number of peers also the piece availability increases in absolute numbers. And, simulations with 1000 peers show that resource pricing is efficient also with $N_R = 50$, $N_{MI} = 40$

and $N_{MP} = 20$.

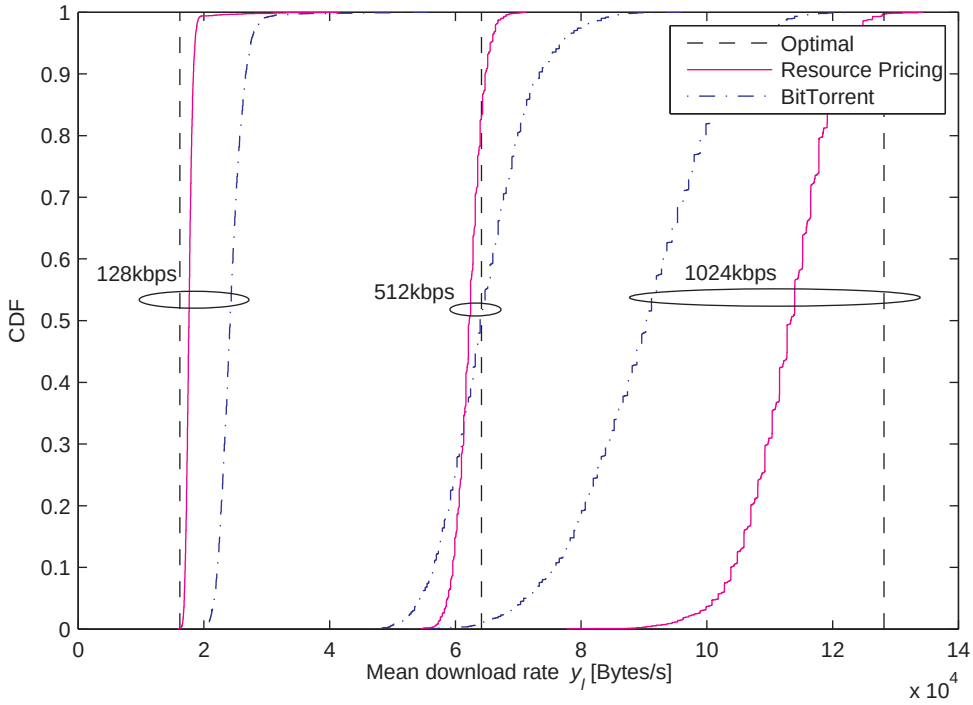
The differences between resource pricing and BitTorrent are small for networks where most of the peers are leechers and have homogeneous upload capacities. This changes when peers have different upload capacities and altruistic behaviour is common. Figure 5.16 shows the CDFs for the mean download rate of peers with different upload capacities of 128 kbps, 512 kbps and 1024 kbps, which are assigned randomly to the peers. In Figure 5.16(a) 1 seed and 99 leechers are in the network. The optimal download rates are computed with (5.73). With resource pricing the download rate is closer to the optimal values than with BitTorrent. Identical to the results from Section 5.3 the download performance with BitTorrent is better for peers with lower capacities than with higher capacities. Also the range for the same capacity is larger for BitTorrent than with resource pricing.

When a large number of seeds is present in the network BitTorrent fails to provide a strong incentive. This is shown in Figure 5.16(b), where half of the peers are seeds. As in Figure 5.16(a) the curve to the left, middle and right for each method corresponds to an upload capacity of 128 kbps, 512 kbps and 1024 kbps, respectively. Since the three curves of BitTorrent are close together, peers do not gain much by contributing higher upload bandwidth to the network. With resource pricing the incentive for peers is much stronger and the curves are closer to the computed optimal values.

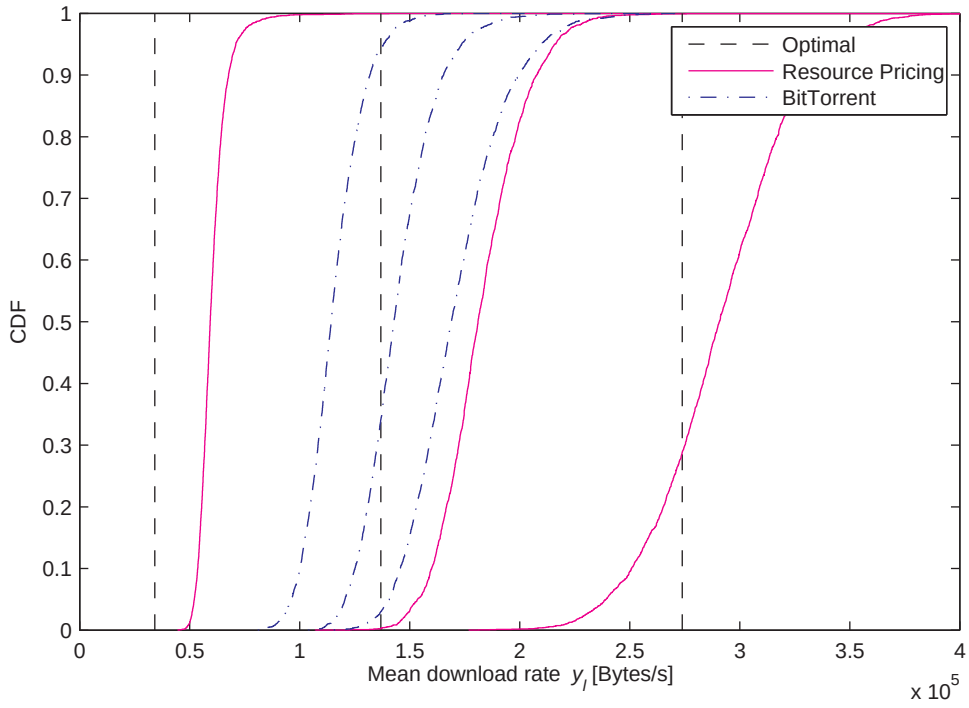
Up to this point we restricted our attention to a scenario where pieces are frequently available at different peers in the network. This is a desirable state of the network since download times are near to the optimal values. In the following we study a scenario where one chunk is rare and observe if the network gets out of this state and attains a state with balanced piece availability. In Section 2.3.1 we derived the download time for a flash-crowd scenario analytically. Additionally, we computed the increase of the rarest chunks. A similar result can be derived for the simulation scenario with one rare chunk.

Assume the rarest chunk is uploaded by the seed to U other peers. Furthermore, when a peer completes the rarest chunk it uploads it to others. Hence, for homogeneous peers with capacity C the number of copies of the rarest chunk increases by $(U+1)^i$, where i is the number of time intervals it takes to upload a full chunk to U peers. With a chunk size of S_C this time interval can be computed with $U \cdot S_C/C$. Hence, the increase of the rarest chunk is exponentially and at fastest rate with $U = 1$. For example, with the default chunk size and homogeneous peers with a capacity of 1 Mbps it takes around 15 s and 31 s to disseminate a chunk to over hundred peers with one and five current uploads, respectively.

In simulations we assume the chunk with id 0 is only available at the seed and all other



(a) 1 seed and 99 leechers



(b) 50 seeds and 50 leechers

Figure 5.16: Network with heterogeneous capacities for Resource Pricing (50,10,10) and BitTorrent (50,40,20) with parameters (N_R, N_{MI}, N_{MP})

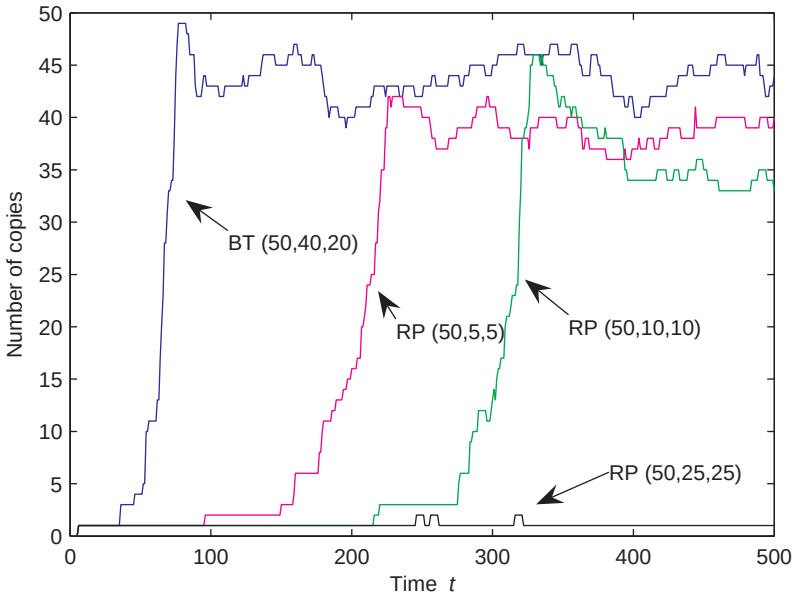


Figure 5.17: Dissemination of one rare chunk

BT (50,40,20)	RP (50,5,5)	RP (50,10,10)	RP (50,25,25)	RP (50,40,20)
85 ± 11	206 ± 37	358 ± 46	1393 ± 423	2120 ± 579

Table 5.3: Time until Chunk 0 is not the rarest chunk (and 95% confidence intervals)

peers have a random set of chunks except chunk 0. Results for BitTorrent and resource pricing are depicted in Figure 5.17 for a specific run and are summarised for 10 runs in Table 5.3. The average time until chunk 0 is no longer the rarest chunk in the network is given in Table 5.3 along with the 95% confidence intervals.

Although the simulation results are much larger than the analytical numbers also the measured dissemination time for the rare chunk increases with the number of upload connections. With BitTorrent a peer uploads to five other peers at a time and after around 85 s the chunk 0 is available frequently in the network. With resource pricing and $N_{MI} = 5$ each peer has more than five connections because it accepts remote connection requests. Hence, it takes around 206 s to disseminate the chunk in the network. For larger values of N_{MI} the time for dissemination increases considerably.

In summary, the results for resource pricing show that fairness is increased with the number of connections up to a point where the network becomes inefficient. Furthermore, dissemination of a rare chunk is better for a small number of upload connections. Hence, there is a trade-off between fairness and efficiency. To adapt dynamically to the network conditions we extend resource pricing in the following and make the price offer dependent

PDRP (50,5,5)	PDRP (50,10,10)	PDRP (50,25,25)	PDRP (50,40,20)
153 ± 15	205 ± 25	238 ± 18	208 ± 15

Table 5.4: Time until Chunk 0 is not the rarest chunk for PDRP

on the piece availability.

5.5.1 Piece-dependent Resource Pricing

We use a linear approach to weigh the price offer λ_c on every connection depending on the requested chunk. If the chunk is rare then higher prices are offered. Thus, the server will increase its rate on this connection. However, this works only as long as a server uploads and a client downloads different chunks. When all peers request the same piece also the price offers are similar and roughly the same upload rate is allocated to them. This extension of resource pricing is denoted as Piece-Dependent Resource Pricing (PDRP) in the following.

Each client estimates the availability of chunks based on the chunk information of its neighbours. Denote the number of neighbours of client c that completed chunk i as f_c^i. The price offer of client c for chunk i is

$$\lambda_c^i = \alpha_c \frac{\lambda_c}{f_c^i}. \tag{5.74}$$

Here, α_c normalises the price offers such that each peer does not offer higher prices than allowed with its willingness-to-pay. To ensure that $W_c = \sum_{s \in \mathcal{S}(c)} x_{sc} \lambda_c^i$ holds we set

$$\alpha_c = \frac{W_c}{\sum_{s \in \mathcal{S}(c)} \frac{x_{sc} \lambda_c^i}{f_c^i}}. \tag{5.75}$$

With (5.74) the weighting factor increases linearly with the rareness of a chunk. For example, a client offers the double price for a piece that is half as frequent as another piece.

We repeat the simulation with the rare chunk form the end of Section 5.5 for piece-dependent resource pricing. The results are given in Figure 5.18 and Table 5.4. For resource pricing with piece-dependent price offers the rare chunk is disseminated faster for all studied parameter sets. The differences for different parameters with $N_{MI} \geq 10$ are small. However, BitTorrent provides also in this case the best results because peers upload to a small number of other peers. This means, weighting the price offers by the

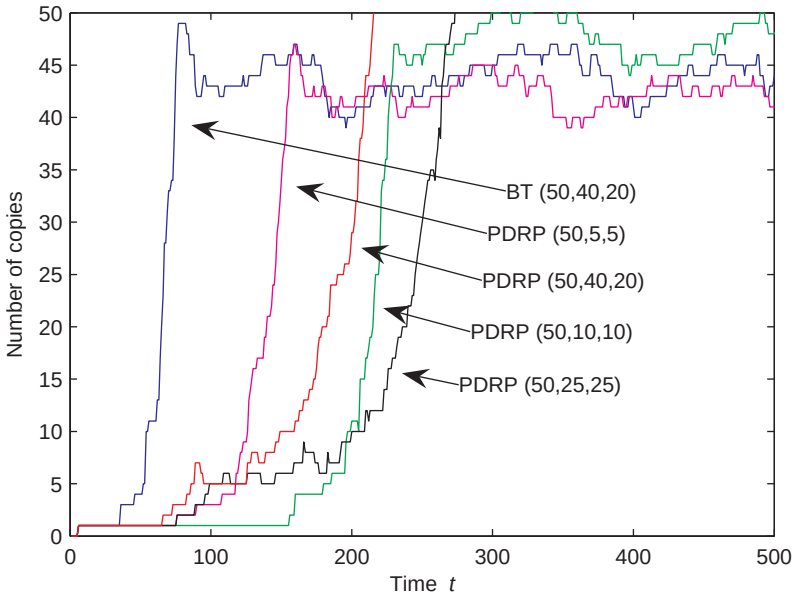


Figure 5.18: Dissemination of one rare chunk for PDRP

piece availability improves the dissemination time to some extent but also the number of uploads should be decreased when some pieces are rare.

We also re-run the other simulations for PDRP. Here, all pieces are frequently available in the network. Hence, PDRP produces very similar results as compared to the original implementation. Also for many connections ($N_R = 50$, $N_{MI} = 40$ and $N_{MP} = 20$) the weighting of the price offer is too slow and the performance is as poor as with resource pricing.

In conclusion, BitTorrent networks are a very efficient way for the fast dissemination of large content in the Internet. Although it provides an incentive to contribute upload bandwidth to the network it does not avoid unfairness between the peers. Especially, for peers with different upload capacities and for altruistically contributed resources the performance becomes unfair. Resource pricing and its extension piece-dependent resource pricing increases fairness. For a medium number of uploads (e.g. for the parameter set (N_R,N_{MI},N_{MP}))=(50,10,10)) the proposed algorithms outperform BitTorrent and give peers a strong incentive to contribute resources. Only for parameter sets where the upload bandwidth is divided among many peers it takes long until a full piece is provided to another peer. Hence, rare chunks might emerge. To some extent this can be avoided by higher price offers for rarer chunks. However, as in BitTorrent limiting the number of concurrent uploads is a more efficient way. To trade off between fairness and dissemination time of rare chunks piece-dependent resource pricing is a good compromise.

5.6 Service Degradation

Up to this point we assumed that the bottleneck in the P2P network is the capacity of the service providers. For the P2P content distribution network this is the upload capacity of the access link of a peer and packet losses in the core network are neglected. When we take service degradation like packet losses in the underlying connection between two peers into account, a service customer sees a smaller service rate than allocated by the service provider. Hence, its price offer λ_c is different than without service degradation, which will influence the rate allocation at the provider in the next update step.

To study the rate allocation with service degradation we repeat the simple simulation from Section 5.3.1, where three client peers and two server peers are present in the network. They are connected according to Figure 5.1. However, in this section we assume an error rate of r^{err} between peers such that the service rate x^c_{sc} at client c is less than the allocated rate x^s_{sc} at server s and

$$x^c_{sc} = (1 - r^{err}_{sc}) x^s_{sc}. \tag{5.76}$$

The influence of the service degradation $e = r^{err}_{s_1c_2}$ on the connection between peer s_1 and c_2 (see Figure 5.1) for different error rates is shown in Figure 5.19 denoted as *price per unit feedback*. The rate allocation for the three customers is equal for a specific error rate and decreases with an increasing error rate. Although only c_2 faces the service degradation it receives the same rate as the other customers, because the price offer per unit bandwidth of c_2 is higher as long as its service rate is smaller compared to the other two customers. Hence, the providers will allocate more service rate to c_2 until the offered prices per unit bandwidth are in equilibrium. With price per unit bandwidth as feedback signal, absolute priority is given to customers with lower rates and the allocation is max-min fair as it can be seen from Figure 5.19.

We computed the max-min fair rates in Figure 5.19 by the water-filling algorithm described in [BG87]. It follows directly from the definition of max-min fairness that resources are fully utilised. Furthermore, the service rates of the three customers are equal as long as these rates do not violate any given constraint.

Besides the max-min fair rate allocation also the proportional fair rate allocation is depicted in Figure 5.19. It can be shown with (3.5) that a proportional fair rate allocation fully utilises the available resources. Furthermore, $(1 - r^{err}_{s_1c_2})y_1 = y_2 = y_3$ holds for $r^{err}_{s_1c_2} \leq 0.5$ and for higher error rates all resources of s_1 are allocated to c_1. With proportional fairness c_1 achieves a higher rate than c_2 and c_3 since priority is given less emphatically to the small rates. The difference between the rates is proportional to $(1 - r^{err}_{s_1c_2})$ and

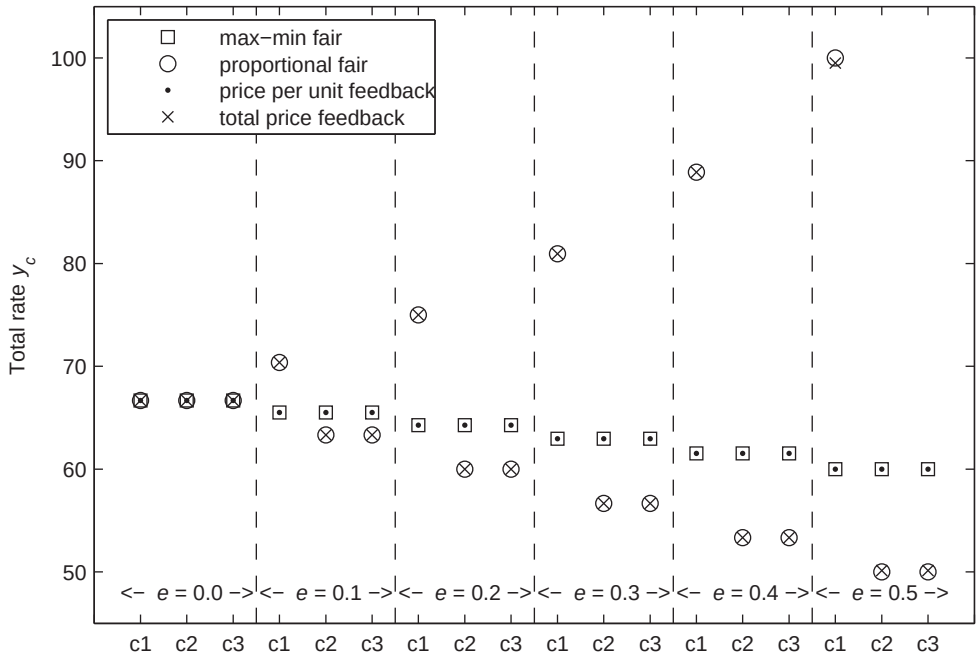


Figure 5.19: Rate allocation for different error rates between s_1 and c_2

is more severe for higher error rates because c_1 benefits from an additional increase in rate to a greater extent as c_2.

We achieve proportional fairness with the proposed resource pricing algorithm if a serviced peer does not signal the feedback λ_c, the offered price per unit bandwidth, but the total offered price $x^c_{sc}\lambda_c$. We distinguish between the allocated and received rate x^s_{sc} and x^c_{sc}, respectively, and (5.17) becomes

$$x^s_{sc}(t+1) = \max\left(\varepsilon, x^s_{sc}(t) + \gamma\left(x^c_{sc}(t)\,\lambda_c(t) - x^s_{sc}(t)\,\frac{\sum_{d\in\mathcal{C}(s)} x^d_{sd}(t)\,\lambda_d(t)}{C_s}\right)\right). \quad (5.77)$$

The results for the simulation with (5.77) are denoted as *total price feedback* in Figure 5.19 and agree with the computed proportional fair rates. A small difference can be seen at $r^{err}_{s_1c_2} = 0.5$ since convergence is slow when a service rate of a provider to a customer tends to zero as it is the case for x_{12}, the rate between server s_1 and client c_2, in this example.

As discussed in Chapter 4 congestion in the network can be controlled by adapting the sending rates of sources based on link prices. Hence, for P2P content distribution the capacity constraints of core links can be incorporated into the optimisation problem in (5.1)-(5.4). Thus, service degradation is included into the model. Such an approach is presented

in the next chapter.

5.7 Conclusion

We studied resource pricing for P2P networks to ensure fair service rates between peers. Based on the global optimisation problem a distributed algorithm is derived where service providing peers adjust their service rates and service requesting peers adjust their price offers. It is shown analytically by a Lyapunov function and for a model with delays that the solution of the global optimisation problem is a globally asymptotically stable equilibrium point for the distributed algorithm. Simulation results reveal that resource pricing is suitable for large and varying peer populations. The measured Weighted Fairness Index is close to the optimal value of one.

The pricing idea is applied to P2P content distribution networks and compared to BitTorrent. It is shown that the resource pricing algorithm and its derivative reciprocal rate control prevent unfairness as opposed to the tit-for-tat scheme of BitTorrent. Furthermore, the distributed algorithm can be extended by the piece availability. Here, the price offer does not solely depend on the total download rate but also on the rareness of the data. To improve availability of each piece a peer offers higher prices for rare chunks. With a linear weighting of the price offer the time to disseminate rare chunks is decreased. However, to ensure efficiency the number of uploads should also be decreased in situations with poor chunk availability. In simulation scenarios, where pieces are available at different peers, resource pricing achieves fairness with a small and moderate number of concurrent uploads and outperforms BitTorrent.

The pricing approach is affected by service degradation on lower layers, e.g. P2P file-sharing applications by packet loss in the IP network. These effects can be included into the model. This is described in the following chapter.

Chapter 6

Rate Control for P2P over IP Networks

The prevalent mechanism to avoid congestion in IP networks is the control of the sending rate with TCP. As discussed in Chapter 4 dynamic routing strategies at the IP layer are not deployed because of problems like route oscillations and out-of-order packet deliveries.

In this chapter we study the interaction between the P2P overlay and the underlying IP network. With the adoption of P2P technology, routing is done also in these overlay networks. With multi-source download protocols peers upload and download to/from other peers in parallel. Hence, by controlling the sending rates between the peers we may achieve two desirable properties: Load balancing at the peers, which upload data, and the avoidance of congested links in the IP network.

Considering load balancing, P2P technology has found already its way to content distribution networks [TP02]. But most of the work assumes that a file request is routed to a single server (in a sophisticated way). Multi-source download protocols adapt faster to changes at the servers or in the network because data is requested at multiple peers in parallel. Hence, the total bandwidth of all serving peers can be allocated in a fair manner to the client peers. Furthermore, also congestion can be avoided. When a connection between two peers becomes overloaded, the uploading peer can favour another peer on an uncongested route. Thus, it uses its upload bandwidth more efficiently. On the other hand the downloading peer could ask for a higher download rate at other providers of the file. Thus, it could receive the same download rate as in an uncongested network.

Based on the resource pricing algorithm for P2P networks in Chapter 5 we propose in this chapter a rate control algorithm for P2P over IP networks. Here, pieces of the file are not taken into account and interest between peers is assumed. We denote the approach as *TCPeer*, because a peer adopts the functionality of TCP and extends it with information

from the overlay network. The service requesting peers make price offers depending on their download rates. The upload bandwidths of the serving peers are allocated based on the price offers of the remote peers minus the path price charged by the links along the routes of these flows. Flows with high price offers and low path prices receive higher rates than others. Thus, a sending peer is able to shift traffic from a congested route to an uncongested one. This change in the rate allocation will be balanced by other peers in the overlay. Hence, the receiving peers experience no degradation of their total download rate. This chapter is based on [EK07c].

6.1 P2P over IP Network Model

We extend the P2P network model of Section 5.1 by the underlying IP network. Hence, the resource allocation is modelled as a cross-layer optimisation problem and functionalities of the transport and application layer are studied concurrently. In general, we say the network consists of three different entities. Besides the set of service providers or servers and the set of service customers or clients, which are also used in Chapter 5, we add the set of links or service carriers known from Chapter 4 to the new model. We associate a client with a user, which requests a file for download. To differentiate between the different entities we introduce the set of servers $\mathcal{S}$, the set of clients $\mathcal{C}$, and the set of links $\mathcal{L}$. The capacity of the servers and the links is finite and is denoted by C.

In a P2P application that supports a multi-source download a client uses several flows to different servers to download a file. We denote the set of servers that can be used by a client c as $\mathcal{S}(c)$. The other way around $\mathcal{C}(s)$ is the set of clients which can be served by server s. Furthermore, we define a flow as a rate allocation from a server $s \in \mathcal{S}$ to a client $c \in \mathcal{C}$ over a route, which is a non-empty subset of all links, denoted as $\mathcal{L}(s,c)$. To identify a flow that uses the link l we introduce δ_{sc}^{l}, where $\delta_{sc}^{l} = 1$ if $l \in \mathcal{L}(s,c)$ and $\delta_{sc}^{l} = 0$ otherwise.

Suppose the utility of a client c is defined by a utility function U_c, which depends on the total download rate y_c, where y_c is the sum of the rates x_{sc} from all servers which are known to c and have parts of the file in which c is interested in. We model the resource

allocation for an IP network with a P2P overlay as a global optimisation problem

$$\text{P2P} + \text{IP SYSTEM}:$$

$$\text{maximise} \quad \sum_{c \in \mathcal{C}} U_c(y_c) \tag{6.1}$$

$$\text{subject to} \quad \sum_{s \in \mathcal{S}(c)} x_{sc} = y_c, \quad \forall c \in \mathcal{C} \tag{6.2}$$

$$\sum_{c \in \mathcal{C}(s)} x_{sc} \leq C_s, \quad \forall s \in \mathcal{S} \tag{6.3}$$

$$\sum_{s \in \mathcal{S}} \sum_{c \in \mathcal{C}(s)} \delta_{sc}^{l} x_{sc} \leq C_l, \quad \forall l \in \mathcal{L} \tag{6.4}$$

$$\text{over} \quad x_{sc} \geq 0. \tag{6.5}$$

Hereby, maximising the aggregated utility of the service rate y_c over all service customers is the objective of the whole system. The problem is constrained by the capacity at the servers and the links. Although servers and links face the same constraint, we differentiate between both in the model, because of their different functionalities: A server can freely allocate its bandwidth over its clients, whereas a link only forwards or discards packets. As in Section 5.1 with a concave, strictly increasing utility function this optimisation problem has a unique optimum with respect to y_c. The rates x_{sc} are not necessarily unique at the optimum, because different rate allocations may sum up to the same total download rate y_c. Thus, many possible rate allocations may exist with respect to x_{sc}.

The Lagrangian of (6.1)-(6.5) is

$$L_{\text{IP}}^{\text{P2P}}(x, y, \lambda, \mu, \nu, m, n) = \sum_{c \in \mathcal{C}} (U_c(y_c) - \lambda_c y_c) + \sum_{s \in \mathcal{S}} \sum_{c \in \mathcal{C}(s)} x_{sc} \left(\lambda_c - \nu_s - \sum_{l \in \mathcal{L}(s,c)} \mu_l \right) + \sum_{s \in \mathcal{S}} \nu_s (C_s - n_s) + \sum_{l \in \mathcal{L}} \mu_l (C_l - m_l), \tag{6.6}$$

where λ_c, ν_s and μ_l are Lagrange multipliers and $n_s \geq 0$ and $m_l \geq 0$ are slack variables. By interpreting the Lagrange multipliers as prices we say λ_c is the price per unit bandwidth offered by the client c and ν_s and μ_l are prices per unit bandwidth charged by the server s and the link l, respectively.

By looking at the Lagrangian in (6.6) we see that the global optimisation problem in (6.1)-(6.5) is separable into sub-problems for the clients, the servers and the links. Furthermore, maximising the total of a sum is equivalent to maximising each summand. Hence, we can

decompose the Lagrangian into the sub-problems

$$\begin{aligned}
&\text{CLIENT } c: \\
&\qquad \text{maximise} \quad U_c(y_c) - \lambda_c y_c && (6.7)\\
&\qquad \text{over} \quad y_c \geq 0 && (6.8)\\
&\text{SERVER } s: \\
&\qquad \text{maximise} \quad \sum_{c\in\mathcal{C}(s)} \left(\lambda_c - \sum_{l\in\mathcal{L}(s,c)} \mu_l\right) x_{sc} && (6.9)\\
&\qquad \text{subject to} \quad \sum_{c\in\mathcal{C}(s)} x_{sc} \leq C_s && (6.10)\\
&\qquad \text{over} \quad x_{sc} \geq 0 && (6.11)\\
&\text{LINK } l: \\
&\qquad \text{maximise} \quad \mu_l\,(C_l - m_l) && (6.12)\\
&\qquad \text{over} \quad \mu_l \geq 0. && (6.13)
\end{aligned}$$

Here, for each sub-problem only locally available information is needed. Only a server needs to be informed about the price offers λ_c of all connected clients minus the prices charged by the links along the used route.

The economical interpretation of the sub-problems is as follows. A client is selfish and tries to maximise its own utility, which depends on the rate y_c. However, the client has to pay a price for using bandwidth. Since λ_c is a price per unit bandwidth, the product $\lambda_c y_c$ reflects the total price that client c pays. On the other hand a server gets paid for the allocation of bandwidth. It is interested in maximising its total revenue. The total revenue is the sum of the revenue for each client, which is computed with price multiplied by quantity. The price the server earns for a unit bandwidth is the price paid by the client minus the cost charged by the links which are needed for forwarding the traffic to the client. Thus, also the server behaves selfishly since it is interested in its own revenue only. Therefore, it will allocate bandwidth preferentially to flows, where clients pay a high price and costs for using the routes to the clients are low. Also the links maximise their revenue. As outlined in Section 3.3 the slack variable m_l is interpreted as the spare capacity. Hence, subtracting m_l from the capacity C_l reflects the total rate of forwarded traffic of this link. By multiplying it with the price per unit bandwidth charged by the link we obtain the revenue of that link.

As already well studied in the previous chapters we set the utility for all clients to the logarithmic function in (3.6). This utility function ensures a weighted proportional fair

resource allocation. Furthermore, in the context of P2P networks the willingness-to-pay can be interpreted as the contribution of a peer to the network. Thus, we could set it to the upload bandwidth, which peer c allocates to the P2P application.

Using (3.6) the optimum (y^*, λ^*) of (6.1)-(6.5) can be computed by differentiating the Lagrange function in (6.6) with respect to y_c and x_{sc}. Hence,

$$y_c^* = \sum_{s \in \mathcal{S}(c)} x_{sc} = \frac{w_c}{\lambda_c^*} \quad \text{if} \quad \mathcal{S}(c) \neq \{\} \tag{6.14}$$

$$\lambda_c^* = \nu_s^* + \sum_{l \in \mathcal{L}(s,c)} \mu_l^* \quad \text{if} \quad x_{sc} > 0 \tag{6.15}$$

$$\leq \nu_s^* + \sum_{l \in \mathcal{L}(s,c)} \mu_l^* \quad \text{if} \quad x_{sc} = 0. \tag{6.16}$$

Furthermore, we deduce from $\partial L / \partial n_s = -\nu_s$ and $\partial L / \partial m_l = -\mu_l$ that the price at a server or at a link is only greater zero, if the corresponding slack variable is zero. Since the slack variable is interpreted as the spare capacity, in equilibrium a price is charged only when the resources are fully used and competition for a resource is present.

In case the bottlenecks of the network are the uplink connections of the servers and all link prices are zero, the model reduces to the one discussed in Chapter 5 and results of the parallel bottleneck model in Section 3.4 apply.

6.2 Distributed Algorithm

An implementation in a decentralised architecture can only use locally available information. Thus, the problems CLIENT, SERVER and LINK from Section 6.1 provide a good starting point for the development of a distributed algorithm. CLIENT is an optimisation problem with respect to y_c, but in a real implementation a client has no direct influence on the total download rate. It can only vary its price offer λ_c. Similar, a link has only influence over its load by varying the charged price. This price depends on the level of congestion. We assume that a server receives the difference of the price offered by a client minus the price charged by the links along the used route and adapts its sending rate to the client. Based on Algorithm 1 we propose the Algorithm 3 for combined rate control in P2P over IP networks. Like in Chapter 5 the parameters ε and η are small positive constants for probing the network conditions. In contrast to Algorithm 1 a server adapts its service rate not only on the price λ_c offered by client c, but on the difference of it to the prices charged by the links of the route. Hence, the remaining price offer from client c

Algorithm 3 Resource Pricing for P2P over IP

CLIENT c :

$$\lambda_c(t) = \frac{W_c}{\max(\eta, y_c(t))} = \frac{W_c}{\max\left(\eta, \sum_{s\in\mathcal{S}(c)} x_{sc}(t)\right)} \tag{6.17}$$

SERVER s :

$$x_{sc}(t+1) = \max\left(\varepsilon, x_{sc}(t) + \gamma x_{sc}(t)\left(p_{cs}(t) - \frac{\sum_{d\in\mathcal{C}(s)} x_{sd}(t)p_{ds}(t)}{C_s}\right)\right) \tag{6.18}$$

LINK l :

$$\mu_l(t+1) = \max(0, \mu_l(t) + \kappa(z_l(t) - C_l)) \tag{6.19}$$

received by server s is

$$p_{cs}(t) = \lambda_c(t) - \sum_{l\in\mathcal{L}(s,c)} \mu_l(t). \tag{6.20}$$

The price of link l in (6.19) depends on the aggregated input rate

$$z_l(t) \leq \sum_{s\in\mathcal{S}} \sum_{c\in\mathcal{C}(s)} \delta_{sc}^l x_{sc}(t). \tag{6.21}$$

The link price is greater zero when the input rate exceeds the capacity and thus the link is overloaded. (6.19) corresponds to the link price rule PC1 in [AL00], which is also used in Chapter 4. Thus, although we extend the congestion pricing model to P2P networks, we do not change the router implementation of previous work for IP networks. Furthermore, we show in the following that the rate adaptation of a server in (6.18) reduces to a differential equation very similar to the primal algorithm (Equation 5) in [KMT98] for the case of a single client-server pair. A client-server pair is a server handling a single client, which is served only by this server. Hence, $\mathcal{C}(s) = \{c\} \wedge \mathcal{S}(c) = \{s\}$ holds for a server s and a client c and the differential equation corresponding to (6.18) reduces to

$$\frac{d}{dt}x_{sc}(t) = \gamma\left(\left(W_c - x_{sc}(t)\sum_{l\in\mathcal{L}(s,c)} \mu_l(t)\right)\left(1 - \frac{x_{sc}(t)}{C_s}\right)\right). \tag{6.22}$$

When the uplink of the server is not the bottleneck the term $(1 - x_{sc}/C_s)$ is just a scaling of the step size γ and (6.22) corresponds to Equation 5 in [KMT98].

Packet-level implementation

The distributed algorithm discussed above considers the problem at flow-level. Now, we transform the algorithm to packet-level. Like in any other TCP implementation the sending rate is specified by the congestion window *cwnd*. *cwnd* is the number of packets being sent per round-trip time (RTT). Hence, the sending rate can be computed with $x = cwnd/RTT$. The congestion window is adapted on each acknowledgement. These are received with rate $cwnd/RTT$ by assuming that the RTT is constant for small time intervals. Therefore, updating the congestion window with

$$\Delta cwnd_{sc}(t) = \gamma RTT_{sc}(t) \left(p_{cs}(t) - \frac{\sum_{d \in C(s)} \frac{cwnd_{sd}(t)}{RTT_{sd}(t)} p_{ds}(t)}{C_s} \right) \tag{6.23}$$

at packet-level corresponds to (6.18) at flow-level.

The link rule in (6.19) computes the price based on the current utilisation only and does not take the queue size into account. To penalise large queues (6.19) can be extended to the link rule in (4.17) from Chapter 4.3, where prices increase when the current queue size is larger than a target queue size. The time between two updates of the link price is T_l. Similarly, we assume that (6.17) is updated every T_c. Both update rules depend on estimated rates, which are computed identically to Section 4.3.

In the following evaluation we assume explicit prices of the links and of the clients are communicated to the server (e.g. in the header of the packets). Although this does not conform to the IP and TCP standards, it demonstrates the functionality of the proposed approach in general. For IP networks a large body of research [GK99b, AL00, ZK02] studies marking strategies where the ECN bit conveys a single bit pricing information. We believe this approach is applicable also for our model, i.e. the ECN bit is used to transport the price information of the links and the P2P protocol is used to communicate the price offers from the client peers to the serving peers.

6.3 Evaluation

To demonstrate the functionality of the distributed algorithm we study two simple examples in ns-2. The network of interest is depicted in Figure 6.1. The first example consists of the two servers $s1$ and $s2$ and the two clients $c1$ and $c2$. The bottlenecks in the net-

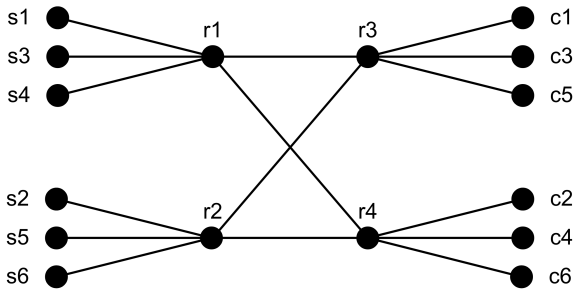
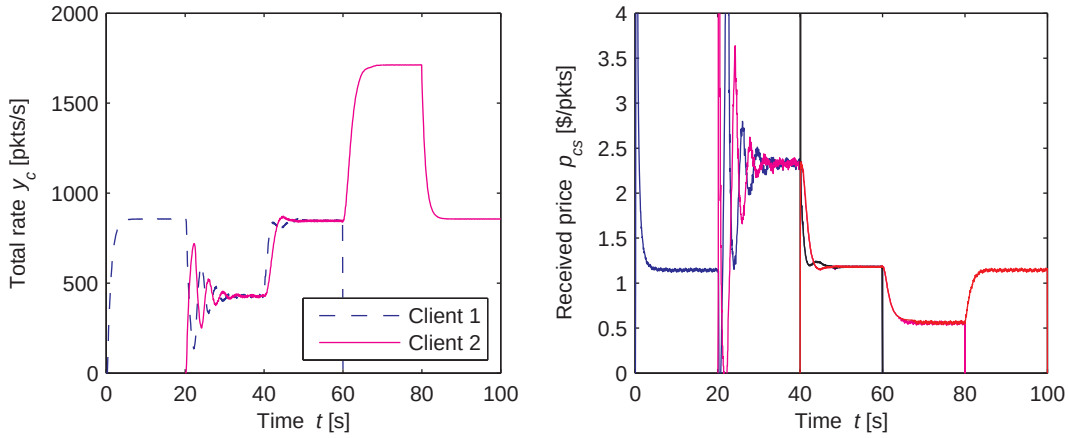


Figure 6.1: Network



(a) Total download rate

(b) Received price

Figure 6.2: Example 1 ($W_c = 1000 \;\; \forall c \in \mathcal{C}$, $\gamma = 0.1RTT$, PC1: $\kappa = 0.1$, $T_l = T_c = 0.1\,\text{s}$)

work are the uplink capacities of the servers and $C_{s1} = C_{s2} = 10\,\text{Mbps}$. The capacity of the links in the core (i.e. in Figure 6.1 the links connecting routers denoted by r) and of the downlinks to the clients is much larger, e.g. $C = 100\,\text{Mbps}$. The delay of all links is 10 ms.

Assume both clients can download from both servers in parallel, if they are online. This represents a small P2P overlay. At $t = 0\,\text{s}$ only server $s1$ and client $c1$ are online. $c2$ and $s2$ are switched on at $t = 20\,\text{s}$ and $t = 40\,\text{s}$, respectively. At $t = 60\,\text{s}$ client $c1$ goes offline, followed by server $s1$ at $t = 80\,\text{s}$.

The total download rate of the two clients and the price information received by the servers (see (6.20)) are depicted in Figure 6.2. For the first 20 seconds only $c1$ and $s1$ are active and the download rate of $c1$ converges to the capacity of $s1$ (which is around 874 packets per second when using a packet size of 1500 B). With the link rule in (6.19) the queue size at the uplink of $s1$ converges to 12 packets. Also the price p_{c1s1} converges to a steady-state value of around 1.15, which can be computed according (3.34) by dividing

the willingness-to-pay of $c1$ with the server capacity in packets.

When $c2$ gets active at $t = 20\,\text{s}$ the server $s1$ receives higher payments because the total willingness-to-pay in the network doubles. The download rates of both clients converge to a fair share of the bottleneck capacity in the following. Hereby, oscillations occur since the estimated download rate (and consequently its price offer) of the new client $c2$ is inaccurate at the beginning. As depicted in Figure 6.2(b) the price received by the server can fall below zero because the server allocates too much bandwidth and the uplink bandwidth is exceeded such that the link price of the server's uplink increases.

When a new server gets online at $t = 40\,\text{s}$ and one client leaves at $t = 60\,\text{s}$ the spare capacity is used efficiently by the online client(s). Thus, this example shows that the proposed algorithm can be used for load balancing because it realises a fair allocation of the servers' capacities.

In the following we investigate the influence of congestion in the core of the network. Therefore, we reduce the capacity for the links $r1-r3$, $r1-r4$, $r2-r3$ and $r2-r4$ in Figure 6.1 to $C_{\text{core}} = 12\,\text{Mbps}$. To avoid large queuing delays we compute link prices with (4.17) in this example. We start with the two servers and the two clients from the previous example, but new client-server pairs get active during the simulation. Assume $s3-c3$, $s4-c4$, $s5-c5$ and $s6-c6$ start at 20 s, 40 s, 60 s and 80 s, respectively. Furthermore, only $c1$ and $c2$ are served in parallel by $s1$ and $s2$. The other clients are served by a single server only (indicated by the same number).

The total download rate for each client is depicted in Figure 6.3. At the beginning the clients $c1$ and $c2$ share the available bandwidth equally between each other as in the previous example. Looking at the congestion window of the four connections in Figure 6.4 we see that not all connections are used equally. It suffices that the total download rate of the two clients is the same.

When $s3$ and $c3$ start at $t = 20\,\text{s}$ the link $r1-r3$ gets congested. This can be seen by an increasing link price in Figure 6.5. Thus, the congestion window between $s1-c1$ as well as the total download rate of client 1 decrease. However, this increases the price offer of client 1 and therefore the congestion window between $s2-c1$. The servers $s1$ and $s2$ shift traffic from one connection to the other one (see Fig. 6.4) and the total download rates of the three active clients converge to a fair and efficient allocation (see Fig. 6.3). The link price of $r1-r3$ falls to zero again, which indicates that it is not a bottleneck in the network.

When the next client-server pair starts at $t = 40\,\text{s}$ prices for the links $r1-r3$ and $r1-r4$ increase and stagnate. The servers $s1$ and $s2$ restart to shift the traffic but cannot avoid

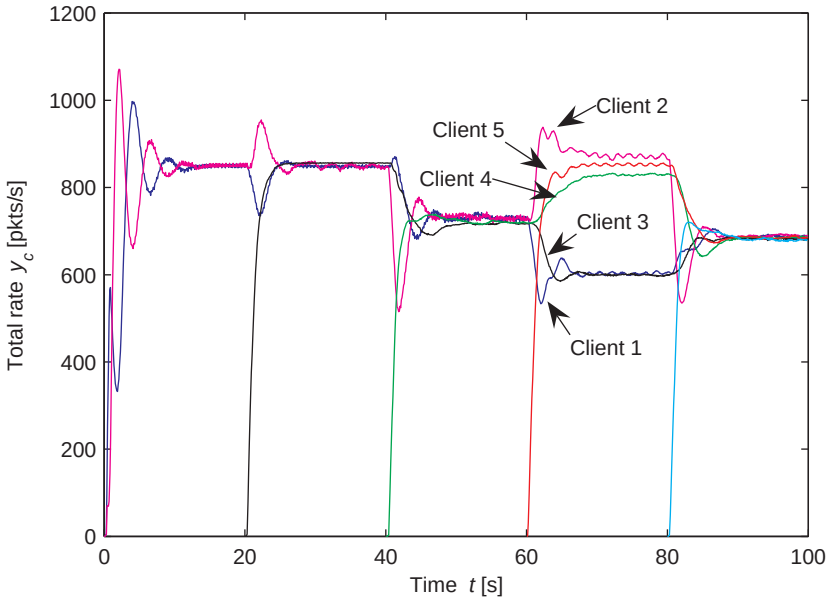


Figure 6.3: Example 2: Total download rate ($w_c = 1000$, $\gamma = 0.1RTT$, PC3: $\kappa = 0.1$, $\alpha = 0.1$, $b_0 = 5$, $T_l = T_c = 0.1\,\mathrm{s}$)

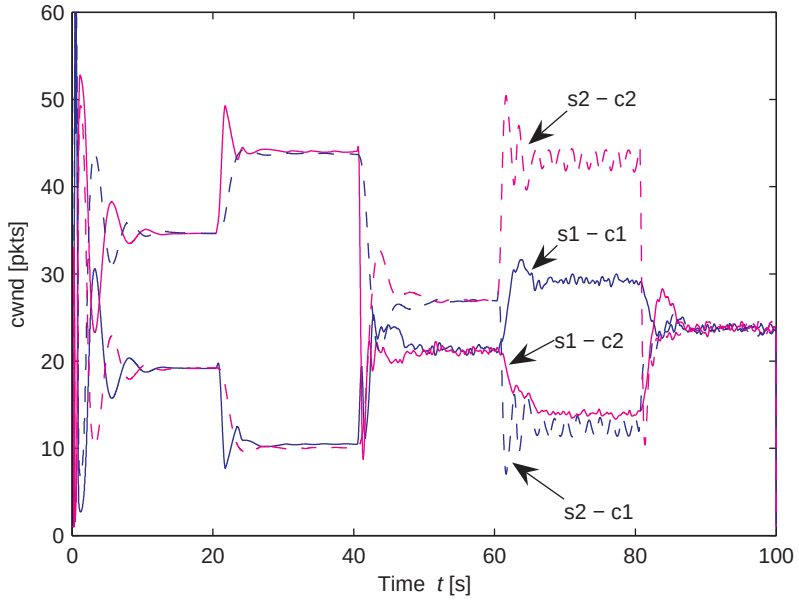


Figure 6.4: Example 2: Congestion window

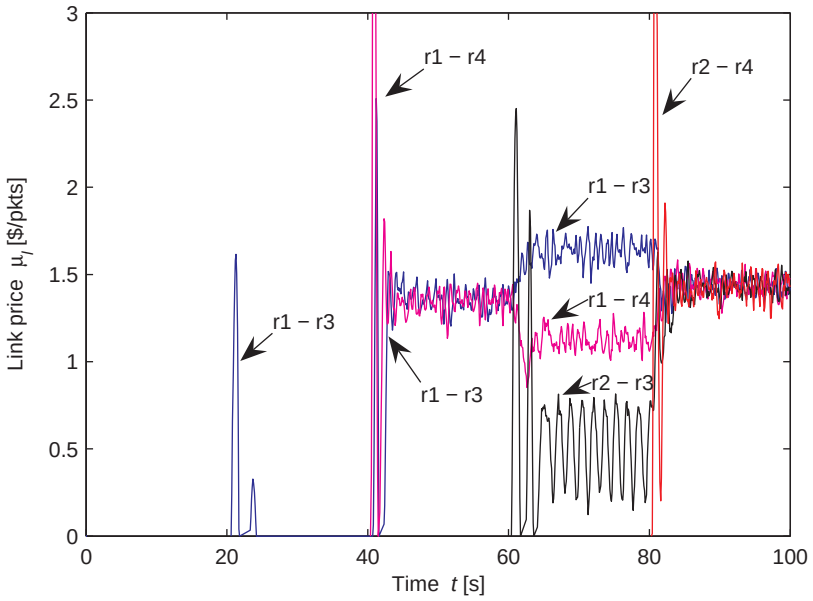


Figure 6.5: Example 2: Link price

congestion in the core network. However, by changing the rate they still ensure a fair rate allocation since all four clients receive roughly the same total download rate.

At $t = 60\,\text{s}$ the connection $s5 - c5$ starts. Thereon, the link $r2 - r3$ becomes overloaded and its link price increases. Now, the network is in a congested state, where a trade-off between fairness and efficiency has to be done. Only connection $s2 - c2$ uses the uncongested core link $r2 - r4$. Thus, $c2$ has the largest download rate. Since most of the server capacity is allocated to the connection $s2 - c2$, client 5 and client 4 benefit from small link prices at $r2 - r3$ and $r1 - r4$. The link price on $r1 - r3$ is the highest one. Hence, client 1 and client 3 get smaller download rates as compared to the other clients.

Finally, with $s6 - c6$ starting at $t = 80\,\text{s}$ all core links get congested and the servers $s1$ and $s2$ control the rate to $c1$ and $c2$ such that bandwidth is allocated in a fair manner. The price oscillates around 1.43, which is the equilibrium price.

This example demonstrates that the proposed algorithm is able to shift traffic from congested to uncongested routes. Thus, resources of the network are used efficiently while preserving fairness between clients.

6.4 Conclusion

In this chapter we model the allocation of resources in an IP network when it is used by a P2P overlay. It extends the well-known congestion pricing approach for IP networks to multi-source download P2P protocols. Furthermore, a distributed algorithm is proposed, where sending peers control the rate depending on the price offers by the remote peers and the path prices charged by the routers in the IP network. Since sending rates depend on the price offers of the remote peers the algorithm ensures fairness between peers. Furthermore, indicated by the path price a serving peer is aware of the level of congestion in the underlying IP network. Thus, it can shift upload bandwidth from congested to uncongested flows. Because of the change in price offers at the receiving peers other senders will change their sending rates accordingly. The rate allocation of all peers converges to an efficient and fair allocation of bandwidth in the network. Furthermore, P2P traffic avoids congested links when uncongested routes between some peers exist. We validate the functionality of the new approach analytically and by simulations for a small network, where explicit price information is assumed.

Chapter 7

Conclusions

In this thesis resource pricing is applied at different layers of the ISO/OSI reference model. We study (i) cross-layer optimisation of rate control and routing in IP networks, (ii) optimisation of the resource allocation in P2P overlay networks and (iii) cross-layer optimisation of rate control for P2P over IP networks.

In general, resource pricing is a mathematical framework for the resource allocation and initially used for rate control at the transport layer. The approach is based on a global optimisation problem, where the resources of the network are modelled as the constraints of the problem. By choosing an appropriate objective function the social welfare is maximised and a specific fairness criterion is realised. Frequently in the literature and throughout this work a logarithmic utility function weighted by the willingness-to-pay is chosen. Hence, the objective is to maximise the sum of the utilities over all users in the network. With this function weighted proportional fairness is achieved. Further on, with linear capacity constraints the optimisation belongs to the set of convex optimisation problems. These problems are fairly tractable to solve in a central way. However, only distributed algorithms ensure scalability for large networks like the Internet.

Distributed algorithms can be derived from the Lagrange dual function, where the constraints of the optimisation problem are internalised with Lagrange multipliers. These multipliers or dual variables are interpreted frequently as prices, which gives the framework its name. By rearranging and decomposing the Lagrange function local optimisation problems are deduced. Further on, distributed algorithms are derived from the local subproblems. In the easiest way gradient projection algorithms can be used. One advantage of dual theory for convex optimisation problems is that the duality gap is zero. Hence, optimising the primal or dual variables yields the same result. Furthermore, in distributed

algorithms primal as well as dual variables are optimised at the same time. In the discussed algorithms dual variables are used as control information and are passed to another entity, where the primal variable is adapted.

The basic model for the transport layer assumes a fixed routing. However, in extensions of this approach and also in this work routing is incorporated into the optimisation problem. Two different attempts are investigated: multi-path and single-path routing. In the case of multi-path routing the utility of the total rate, the sum of the rates over all available routes, is maximised. Although this problem is not unique with respect to the rates on the routes distributed algorithms are derived that converge to the solution of the cross-layer optimisation problem.

With single-path routing, the de facto standard in the Internet, the optimisation becomes an integer problem. This means, the duality gap might be nonzero and the optimal dual variables will not result in the optimum of the primal problem. In a distributed implementation oscillations occur and the algorithms are not stable. Furthermore, we show by simulations that also with equal-cost multi-path oscillations are not avoided. Although multi-path routing makes use of all available resources it is unlikely that IP networks will deviate from the single-path concept. Hence, overlay networks offer an alternative to the rigid design of the underlying IP network.

P2P networks are a good example for overlay networks that implement similar functions as in the IP network. In multi-source download protocols for content dissemination the file is fragmented into pieces and peers download different pieces from different peers at the same time. Furthermore, to realise incentive mechanisms peers control the upload rate to others. Hence, routing and rate control are implemented also in the P2P network.

In the first step we apply resource pricing to P2P networks and propose a mathematical model for the resource allocation. Although the optimisation problem for the P2P model is very similar to the one for IP networks the distributed algorithms are different: In IP networks the TCP sender controls the sending rate. The corresponding TCP flow consumes bandwidth at each hop along its path to the sink. But the routers charge a price for the usage of bandwidth, which is summed up to a path price for each TCP flow and controls the future sending rate of the flow.

In the P2P network a peer chooses freely the peers it provides service to and allocates resources for the service. Therefore, resources can be used efficiently at a peer when at least one other peer requests a service from it. However, peers act frequently on their own interest. Hence, an incentive to contribute resources to the P2P network should be given to a peer. Furthermore, a requesting peer is normally serviced by several providers in

parallel. Hence, its total service rate is accumulated over its providers. To ensure fairness the service requesting peers offer prices for the service and each providing peer sets the service rate according to the price offers of its customers.

A distributed algorithm is derived and proofs are presented, which show the efficiency, fairness and stability of this approach. By applying the algorithm to P2P content distribution networks fairness between peers can be increased as compared to the popular BitTorrent protocol. Fairness is of great importance in P2P networks, because by setting the willingness-to-pay of a peer to its upload bandwidth it gives peers an incentive to contribute resources to the network. When the piece availability is taken into account a trade-off between efficiency and fairness has to be done. With an increasing number of uploads fairness is improved. However, rare pieces might emerge because the upload bandwidth is divided between many peers. One alternative is to set higher prices for rare pieces. This will fasten the dissemination of rare pieces. The proposed distributed algorithm shows good fairness for small and moderate number of concurrent uploads. The approach outperforms BitTorrent especially when peers have different upload capacities and many seeds are in the network.

In the second step the P2P and IP network is modelled in a single optimisation problem. Since combined rate control and routing in IP networks is possible with multi-path but not with single-path routing we overcome this by using the routing function in the P2P overlay. For multi-source download we propose a TCP variant that uses information from the P2P network. Here, load balancing at the serving peers and fairness between the served peers is achieved in the overlay. At the same time this approach reacts on congestion in the IP network. Hence, traffic is shifted from congested to uncongested routes by ensuring fairness between peers.

In summary, this thesis presents a novel decentralised approach to the resource allocation in P2P networks. It builds on an optimisation model for TCP/IP. Both approaches are combined in a cross-layer optimisation problem to derive in addition a rate control algorithm for P2P over IP networks.

Appendix A

Overlay Topology of BitTorrent

According to the original BitTorrent specification [Coh03, BTa] the overlay topology is constructed with information returned by the tracker. This centralised component stores information about all peers. A new peer, which enters the network, asks the tracker for a list of active peers in the overlay. The tracker returns a random subset of peer addresses to the requesting peer. Furthermore, an active peer contacts the tracker from time to time to obtain information about new peers in the network.

In the simulations discussed in Section 5.4.3 we consider static and dynamic P2P networks. In the static case all P peers are registered at the tracker. Then each peer obtains a peer list of length N_R from the tracker. If we assume that all peers accept every incoming connection request, a peer p iterates through its peer list and opens N_R connections. Furthermore, one more connection is established when a remote peer which is unknown to p, i.e. not on the peer list of p, contacts it. The probability that x peers contact p is $B(x)$ and follows the binomial distribution

$$B(x) = \binom{P-1}{x} Y^x (1-Y)^{P-1-x}, \tag{A.1}$$

where Y is the probability that peer p is on the peer list of a peer q under the assumption that q is not on the list of p.

The probability that a specific peer address is returned by the tracker is based on the hypergeometric distribution and is

$$\frac{1 \cdot \binom{P-2}{N_R-1}}{\binom{P-1}{N_R}} = \frac{N_R}{P-1}, \tag{A.2}$$

when we assume a tracker does not return the address of the requesting peer. Thus, we can compute Y with

$$Y = \frac{N_R}{P-1}\left(1 - \frac{N_R}{P-1}\right). \tag{A.3}$$

The expectation value $E[N_C]$ of the number of connections N_C of a peer is the sum of N_R and the expectation value $E[X]$ of the random variable X which is distributed according to (A.1). Therefore,

$$\begin{aligned} E[N_C] &= N_R + E[X] && \text{(A.4)}\\ &= N_R + (P-1)\frac{N_R}{P-1}\left(1 - \frac{N_R}{P-1}\right) && \text{(A.5)}\\ &= N_R + N_R\left(1 - \frac{N_R}{P-1}\right) && \text{(A.6)}\\ &\approx 2N_R && \text{(A.7)} \end{aligned}$$

The approximation in (A.7) holds if $N_R \ll P$.

For the simulations with dynamics we model the peer behaviour as a Poisson process, where the interarrival times between peers and the session times of peers are exponentially distributed. The expectation value of the number of peers in the network is $E[P] = \lambda^{arr}/\mu^{ses}$ which depends on the mean arrival rate $\lambda^{arr} = 1/T^{arr}$ and the mean session time $T^{ses} = 1/\mu^{ses}$. We assume the tracker returns valid information of online peers only. Since the Poisson process is memoryless a peer which is asked to open a new connection by a remote peer has a remaining session time distributed with mean $1/\mu^{ses}$. Therefore, the rate of closing connections for a peer is $N_R \cdot \mu^{ses}$.

On the other hand new peers will open a connection to peer p. The probability that peer p is chosen by a new peer is given by (A.2) and can be estimated with $E[P]$. New peers enter with the rate of λ^{arr}. Therefore, the rate of opening new connections is $\lambda^{arr}\frac{N_R}{E[P]-1}$. Hence, the expectation value of the number of connections is

$$\begin{aligned} E[N_C] &= \frac{\lambda^{arr} \cdot N_R}{(E[P]-1)N_R \cdot \mu^{ses}} N_R && \text{(A.8)}\\ &\approx \frac{\lambda^{arr} \cdot N_R}{\frac{\lambda^{arr}}{\mu^{ses}} \cdot N_R \cdot \mu^{ses}} N_R && \text{(A.9)}\\ &= N_R && \text{(A.10)} \end{aligned}$$

The results of the expected number of connections in the static and the dynamic network

in (A.7) and (A.10), respectively, are different and deviate by a factor of 2. This has to be taken into account for the simulations where fairness is studied for different values of N_R.

Appendix B

List of Symbols

Variables:

C	Capacity
$cwnd$	Congestion window
L	Number of leechers
N	Number of users
N_{MI}	Number of connections at which a peer stops initiating new ones
N_{MP}	Lower limit of connections at which a peer does not re-request the tracker
N_R	Number of peers returned by tracker, size of a random subset of peers
P	Number of peers
S	Number of seeds
S_F	File size
S_C	Chunk size
RTT	Round-trip time
U	Number of concurrent uploads
v_r	Willingness-to-pay for route r
W	Willingness-to-pay
x	Rate of a connection
y	Aggregated download rate
z_l	Aggregated input rate of link l
z_s	Aggregated upload rate of server s

λ	Price offer
μ, ν	Charged prices
m, n	Slack variables
γ, κ	Step sizes
ε, η	Small positive constants

Functions:

$L(\cdot)$	Lagrange dual function
$U(\cdot)$	Utility function
$V(\cdot)$	Lyapunov function

Sets:

$\mathcal{C}$	Set of clients
$\mathcal{L}$	Set of links
$\mathcal{N}$	Set of users
$\mathcal{N}(p)$	Set of neighbours of p
$\mathcal{R}$	Set of routes
$\mathcal{P}$	Set of peers
$\mathcal{S}$	Set of servers

Bibliography

[ABKM01] D. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. 18th ACM Symposium on Operating Systems Principles*, Banff, Canada, Oct. 2001. 10

[ACM04] P. Antoniadis, C. Courcoubetis, and R. Mason, "Comparing economic incentives in peer-to-peer networks," *Computer Networks*, vol. 46, no. 1, pp. 133–146, 2004. 72

[AH00] E. Adar and B. A. Huberman, "Free riding on Gnutella," *First Monday*, vol. 5, no. 10, Oct. 2000. 14, 72

[AL00] S. Athuraliya and S. Low, "Optimization flow control, II: Implementation," Melbourne University, Technical Report, 2000. [Online]. Available: http://netlab.caltech.edu/pub/papers/rem2.ps.gz 8, 19, 35, 41, 43, 106, 107

[All03] W. Allcock, *GridFTP: Protocol Extensions to FTP for the Grid*, Open Grid Forum (OGF), 2003. 9

[ALTA06] S. Agarwal, M. Laifenfeld, A. Trachtenberg, and M. Alanyali, "Fast data access over asymmetric channels using fair and secure bandwidth sharing," in *Proc. of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS '06)*, Washington D.C., USA, 2006, pp. 58–62. 77

[ASK+02] A. Akella, S. Seshan, R. Karp, S. Shenker, and C. Papadimitriou, "Selfish behavior and stability of the Internet:: a game-theoretic analysis of TCP," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 117–130, 2002. 10

[BD92] W. Boyce and R. DiPrima, *Elementary Differential Equations and Boundary Value Problems*, 5th ed. John Wiley & Sons, 1992. 57

[BG87] D. Bertsekas and R. Gallager, *Data networks*. Prentice-Hall. Inc., 1987. 24, 97

[BHP05] A. R. Bharambe, C. Herley, and V. N. Padmanabhan, "Analyzing and improving BitTorrent performance," Microsoft Research, Tech. Rep. MSR-TR-2005-03, 2005. [Online]. Available: http://www.research.microsoft.com/~padmanab/papers/msr-tr-2005-03.pdf 72, 73, 78, 88

[BP95] L. Brakmo and L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," *IEEE Journal on Selected Areas in Communication*, vol. 13, no. 8, pp. 1465–1480, Oct. 1995. 7

[Bri07] B. Briscoe, "Flow rate fairness: dismantling a religion," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 2, pp. 63–74, 2007. 10, 24

[BT89] D. Bertsekas and J. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice-Hall. Inc., 1989. 35, 53, 61, 62

[BTa] "Bittorrent protocol specification v1.0." [Online]. Available: http://wiki.theory.org/BitTorrentSpecification 14, 15, 79, 89, 90, 117

[BTb] "Bittorrent enhancement proposal: DHT protocol." [Online]. Available: http://www.bittorrent.org/beps/bep_0005.html 13, 14

[BV06] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2006. 23, 26, 27, 28, 36, 62

[CCS96] I. Castineyra, N. Chiappa, and M. Steenstrup, "The Nimrod routing architecture," *IETF RFC 1992*, 1996. 9

[CDKR02] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "Scribe: a large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1489–1499, Oct. 2002. 13

[Chi07] M. Chiang, "Nonconvex optimization of communication networks," *Advances in Mechanics and Mathematics, Springer*, Oct. 2007. 19

[CLCD07] M. Chiang, S. Low, A. Calderbank, and J. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 255–312, Jan. 2007. 19, 20

[Coh03] B. Cohen, “Incentives build robustness in BitTorrent,” in *Proc. 1st Workshop on Economics of Peer-to-Peer Systems*, Berkeley, Jun. 2003. 11, 13, 14, 72, 79, 117

[CW03] C. Courcoubetis and R. Weber, *Pricing Communication Networks: Economics, Technology and Modelling (Wiley Interscience Series in Systems and Optimization)*. John Wiley & Sons, 2003. 48

[CXH06] W. Chehai, L. Xianliang, and D. Hancong, “Analysis of content availability optimization in BitTorrent,” in *Proc. 2006 International Conference on Hybrid Information Technology (ICHIT’06)*, Los Alamitos, CA, USA, Nov. 2006, pp. 525–529. 88

[DBEB06] M. Duke, R. Braden, W. Eddy, and E. Blanton, “A roadmap for transmission control protocol (TCP),” *IETF RFC 4614*, Sep. 2006. 7

[Dem90] H. Demmler, *Einführung in die Volkswirtschaftslehre - Elementare Preistheorie*. R. Oldenbourg Verlag, 1990. 47

[EHBK07] K. Eger, T. Hoßfeld, A. Binzenhöfer, and G. Kunzmann, “Efficient simulation of large-scale P2P networks: Packet-level vs. flow-level simulations,” in *2nd Workshop on the Use of P2P, GRID and Agents for the Development of Content Networks (UPGRADE-CN’07)*, Monterey Bay, CA, USA, Jun. 2007, pp. 9–16. 3, 16, 78

[EK05] K. Eger and U. Killat, “Revenue maximisation in peer-to-peer networks,” in *Workshop Peer-to-Peer-Systeme und -Anwendungen, 14.GI/ITG-Fachtagung Kommunikation in Verteilten Systemen (KiVS)*, Kaiserslautern, Germany, Mar. 2005, pp. 221–224. 2, 50

[EK06a] K. Eger and U. Killat, “Bandwidth trading in unstructured P2P content distribution networks,” in *Proc. IEEE International Conference on Peer-to-Peer Computing (P2P2006)*, Cambridge, UK, Sep. 2006, pp. 39–46. 3, 50

[EK06b] K. Eger and U. Killat, “Die überlastpreisgestaltung als ein einheitlicher Ansatz zur Ratenregelung und Wegfindung,” in *Proc. 13th GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems (MMB 2006)*, Nürnberg, Germany, Mar. 2006, pp. 363–380. 2, 33, 43

[EK06c] K. Eger and U. Killat, "Fair resource allocation in peer-to-peer networks," in *Proc. 2006 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'06)*, Calgary, Canada, Jul. 2006, pp. 39–45. 2, 50

[EK07a] K. Eger and U. Killat, "Fair resource allocation in peer-to-peer networks (extended version)," *Computer Communications, Special Issue: Advances in Communication Networking, Elsevier*, vol. 30, no. 16, pp. 3046–3054, Nov. 2007. 2, 50

[EK07b] K. Eger and U. Killat, "Resource pricing in peer-to-peer networks," *IEEE Communications Letters*, vol. 11, no. 1, pp. 82–84, Jan. 2007. 3, 29, 50

[EK07c] K. Eger and U. Killat, "TCPeer: Rate control in P2P over IP networks," in *LNCS 4516: 20th International Teletraffic Congress (ITC20)*, Ottawa, Canada, Jun. 2007, pp. 618–629. 3, 102

[EK08] K. Eger and U. Killat, "Bandwidth trading in BitTorrent-like P2P networks for content distribution," *Computer Communications, Special Issue: Foundation of Peer-to-Peer Computing*, vol. 31, no. 2, pp. 201–211, Feb. 2008. 3, 50

[Ela05] S. N. Elaydi, *An introduction to difference equations*, 3rd ed. Springer-Verlag New York, Inc., 2005. 56

[EMW06] A. Elwalid, D. Mitra, and Q. Wang, "Distributed nonlinear integer optimization for data-optical internetworking," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1502–1513, Aug. 2006. 19

[FC05] M. Fazel and M. Chiang, "Network utility maximization with nonconcave utilities using sum-of-squares method," in *Proc. 44th IEEE Conference on Decision and Control*, Dec. 2005, pp. 1867–1874. 20

[FF99] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 458–472, 1999. 53

[FH99] S. Floyd and T. Henderson, "The NewReno modification to TCP's fast recovery algorithm," *IETF RFC 2582*, Apr. 1999. 7

[FH03] L. Fortnow and S. Homer, "A short history of computational complexity," *Bulletin of the EATCS*, vol. 80, pp. 95–133, 2003. 88

[FJ93] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993. 7

[Flo03] S. Floyd, "HighSpeed TCP for large congestion windows," *IETF RFC 3649*, Dec. 2003. 7

[Flo07] S. Floyd, "Measurement studies of end-to-end congestion control in the Internet," Last modified in April 2007. [Online]. Available: http://www.icir.org/floyd/ccmeasure.html 78

[GK99a] R. Gibbens and F. Kelly, "Distributed connection acceptance control for a connectionless network," in *Proc. 16th International Teletraffic Congress (ITC)*, Edinburgh, UK, Jun. 1999. 19

[GK99b] R. Gibbens and F. Kelly, "Resource pricing and the evolution of congestion control," *Automatica*, vol. 35, pp. 1969–1985, 1999. 8, 19, 22, 43, 107

[GLBML01] P. Golle, K. Leyton-Brown, I. Mironov, and M. Lillibridge, "Incentives for sharing in peer-to-peer networks," *Lecture Notes in Computer Science*, vol. 2232, pp. 75–86, 2001. 72

[Gnu] *The Annotated Gnutella Protocol Specification v0.4*, http://rfc-gnutella.sourceforge.net/developer/stable/index.html. 11

[GR05] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," in *Proc. IEEE INFOCOM*, Miami, Mar. 2005. 73, 87

[Har68] G. Hardin, "The tragedy of the commons," *Science*, vol. 162, no. 3859, pp. 1243–1248, Dec. 1968. 18

[HGM+] B. Hubert, T. Graf, G. Maxwell, R. van Mook, M. van Oosterhout, P. Schroeder, J. Spaans, and P. Larroy, *Linux Advanced Routing & Traffic Control HOWTO*. [Online]. Available: http://lartc.org/ 73

[HNA04] T. Hacker, B. Noble, and B. Athey, "Improving throughput and maintaining fairness using parallel TCP," in *Proc. IEEE INFOCOM*, Hong Kong, Mar. 2004, pp. 2480–2489. 9

[Hoe95] J. Hoe, "Start-up dynamics of TCP's congestion control and avoidance scheme," Master's thesis, Massachusetts Institute of Technology, Jun. 1995. 7

[HSH+03] H. Han, S. Shakkottai, C. Hollot, R. Srikant, and D. Towsley, "Overlay TCP for multi-path routing and congestion control," in *Proc. ENS-INRIA ARC-TCP Workshop*, Paris, France, 2003. 37

[ISC] *Internet Domain Survey*, Internet Systems Consortium, Inc. (ISC), Online available: http://www.isc.org. 9

[IUKB+04] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. Felber, A. A. Hamra, and L. Garcés-Erice, "Dissecting BitTorrent: Five months in a torrent's lifetime." in *Passive and Active Mesurements*, Apr. 2004, pp. 1–11. 72

[IWS] "World Internet usage and population statistics," Internet World Stats. [Online]. Available: http://www.internetworldstats.com 1, 9

[Jac88] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM '88*, Stanford, CA, Aug. 1988, pp. 314–329. 7, 36, 37, 53

[Jac90] V. Jacobson, "Modified TCP congestion avoidance algorithm," end2end-interest mailing list, April 30, 1990. [Online]. Available: ftp://ftp.isi.edu/end2end/end2end-interest-1990.mail 7

[Jai91] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, 1991. 25

[Joh04] R. Johari, "Efficiency loss in market mechanisms for resource allocation," Ph.D. dissertation, Massachusetts Institute of Technology, 2004. 48

[JT01] R. Johari and D. K. H. Tan, "End-to-end congestion control for the internet: delays and stability," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 818–832, Dec. 2001. 56, 60

[JWL04] C. Jin, D. Wei, and S. Low, "FAST TCP: Motivation, architecture, algorithms, performance," in *Proc. IEEE INFOCOM*, Hong Kong, Mar. 2004, pp. 2490–2501. 7, 8, 19

[Kel97] F. Kelly, "Charging and rate control for elastic traffic," in *European Transactions on Telecommunications*, vol. 8, Jan. 1997, pp. 33–37. 2, 18, 21, 22, 25, 28, 52

[Kel03] T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," *Computer Communication Review*, vol. 32, no. 2, Apr. 2003. 7

[Key01] P. Key, "Resource pricing for differentiated services," in *Kommunikation in Verteilten Systemen (KiVS)*, 2001, pp. 3–18. 8, 19, 22, 56

[Kha00] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Prentice-Hall. Inc., 2000. 57

[KHR02] D. Katabi, M. Handley, and C. Rohrs, "Internet congestion control for future high bandwidth-delay product environments," in *ACM SIGCOMM*, Aug. 2002. 8

[KMBL99] P. Key, D. McAuley, P. Barham, and K. Laevens, "Congestion pricing for congestion avoidance," Microsoft Research, Tech. Rep. MSR-TR-99-15, 1999. 19

[KMT98] F. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, Mar. 1998. 2, 18, 21, 22, 24, 25, 26, 27, 36, 53, 56, 57, 106

[KST01] K. Kar, S. Sarkar, and L. Tassiulas, "Optimization based rate control for multipath sessions," in *Teletraffic Engineering in the Internet Era, Proc. of the International Teletraffic Congress – ITC-17, Salvador da Bahia, Brazil*, Sep. 2001, pp. 805–816. 33

[KV05] F. Kelly and T. Voice, "Stability of end-to-end algorithms for joint routing and rate control," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 2, pp. 5–12, 2005. 33, 54, 56

[KZ89] A. Khanna and J. Zinky, "The revised ARPANET routing metric," *SIGCOMM Comput. Commun. Rev.*, vol. 19, no. 4, pp. 45–56, 1989. 33

[LAPF05] L. Lopez, G. Almansa, S. Paquelet, and A. Fernandez, "A mathematical model for the TCP tragedy of the commons," *Theoretical Computer Science*, vol. 343, pp. 4–26, 2005. 10

[LCP+05] K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Surveys & Tutorials*, pp. 72–93, 2005. 13

[LL99] S. Low and D. Lapsley, "Optimization flow control, I: basic algorithm and convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861–874, 1999. 22

[LL07] A. Liotta and L. Lin, "IP multimedia subsystem - the operator's response to P2P service demand," *IEEE Communications Magazine*, vol. 45, no. 7, pp. 76–83, Jul. 2007. 17, 88

[LMS05] J.-W. Lee, R. Mazumdar, and N. Shroff, "Non-convex optimization and rate control for multi-class services in the internet," *IEEE/ACM Transactions on Networking*, vol. 13, no. 4, pp. 827– 840, Aug. 2005. 20

[LPD02] S. Low, F. Paganini, and J. Doyle, "Internet congestion control," *IEEE Control Systems Magazine*, vol. 22, no. 1, pp. 28–43, 2002. 8, 19, 22

[LSS06] X. Lin, N. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1452–1463, Aug. 2006. 20

[Lue03] D. Luenberger, *Linear and Nonlinear Programming*, 2nd ed. Kluwer Academic Publishers, 2003. 26

[MAF05] A. Medina, M. Allman, and S. Floyd, "Measuring the evolution of transport protocols in the Internet," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 2, pp. 37–52, 2005. 8

[Mal06] S. Malik, "Aggregate Internet traffic: Considerations for the planning of high speed IP networks," Ph.D. dissertation, Hamburg University of Technology, Germany, 2006. 11

[Mas02] L. Massoulie, "Stability of distributed congestion control with heterogeneousfeedback delays," *IEEE Transactions on Automatic Control*, vol. 47, no. 6, pp. 895–902, Jun. 2002. 19

[MMV94] J. MacKie-Mason and H. Varian, "Pricing the Internet," in *Public Access to the Internet*, B. Kahin and J. Keller, Eds. Englewood Cliffs, NJ: Prentice-Hall. Inc., 1994. 18

[Moy98] J. Moy, "OSPF version 2," *IETF RFC 2328*, 1998. 8, 9

[MWW06] J. Mundinger, R. R. Weber, and G. Weiss, "Analysis of peer-to-peer file dissemination," *Performance Evaluation Review, MAMA 2006 Issue*, 2006. 16

[Nap] "Napster messages." [Online]. Available: http://opennap.sourceforge.net/napster.txt 11

[ns2] "The Network Simulator - ns-2." [Online]. Available: http://nsnam.isi.edu/nsnam/index.php/User_Information 41

[Odl03] A. Odlyzko, "Data networks are lightly utilized, and will stay that way," *Review of Network Economics*, vol. 2, no. 3, pp. 210–237, Sep. 2003. 78

[Osb04] M. J. Osborne, *An Introduction to Game Theory*. Oxford University Press, 2004. 72, 77

[Pag06] F. Paganini, "Congestion control with adaptive multipath routing based on optimization," in *40th Annual Conference on Information Sciences and Systems*, Mar. 2006, pp. 333–338. 33

[Par05] A. Parker, "Peer-to-peer in 2005," CacheLogic, Tech. Rep., 2005. [Online]. Available: http://www.cachelogic.com 13

[PD07] L. L. Peterson and B. S. Davie, *Computer Networks: A Systems Approach*, 4th ed. Morgan Kaufmann, 2007. 7, 44

[Pos81] J. Postel, "STD 5: Internet Protocol: DARPA Internet Program Protocol Specification," Sep. 1981. 44

[PR05] R. S. Pindyck and D. L. Rubinfeld, *Microeconomics*, 6th ed. Pearson Prentice Hall, 2005. 23, 26, 35

[QS04] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," *Computer Communication Review*, vol. 34, no. 4, pp. 367–378, 2004. 73, 74, 77, 78, 79, 88

[QYZS06] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker, "On selfish routing in Internet-like environments," *IEEE/ACM Transactions on Networking*, vol. 14, no. 4, pp. 725–738, 2006. 10

[RFB01] K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ECN) to IP," *IETF RFC 3168*, 2001. 7, 8, 43

[RGRK04] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a DHT," in *ATEC'04: Proceedings of the USENIX Annual Technical Conference 2004 on USENIX Annual Technical Conference*, Berkeley, CA, USA, 2004. 65

[RHE00] R. Rejaie, M. Handley, and D. Estrin, "Layered quality adaptation for internet video streaming," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 12, pp. 2530–2543, Dec. 2000. 6

[SE05] R. Steinmetz and K. W. (Eds.), *Peer-to-Peer Systems and Applications*. Series: Lecture Notes in Computer Science, Vol. 3485, Springer-Verlag, 2005. 10, 13

[SFKT06] K. Suh, D. Figueiredo, J. Kurose, and D. Towsley, "Characterizing and detecting Skype-relayed traffic," in *Proc. IEEE INFOCOM 2006*, Barcelona, Spain, Apr. 2006. 10

[She95] S. Shenker, "Fundamental design issues for the future Internet," *IEEE Journal on Selected Areas in Communication*, vol. 13, no. 7, Sep. 1995. 6, 20, 23, 52

[Ste94] W. R. Stevens, *TCP/IP Illustrated Volume 1, The Protocols*. Cambridge: Addison-Wesley, 1994. 6, 7

[Tan02] A. S. Tanenbaum, *Computer networks*, 4th ed. Prentice-Hall. Inc., 2002. 6, 7, 8, 9

[TC05] R. Thommes and M. Coates, "BitTorrent fairness: analysis and improvements," in *Proc. Workshop Internet, Telecom. and Signal Proc.*, Noosa, Australia, Dec. 2005. 74, 79

[TP02] E. Turrini and F. Panzieri, "Using P2P techniques for content distribution internetworking: A research proposal," in *Proc. IEEE P2P 2002*, 2002. 101

[Vin02] G. Vinnicombe, "On the stability of networks operating TCP-like congestion control," in *Proc. IFAC*, 2002. 19

[Voi05] T. Voice, “Stability of congestion control algorithms with multi-path routing and linear stochastic modelling of congestion control,” Ph.D. dissertation, University of Cambridge, 2005. 36, 37

[WLLD05] J. Wang, L. Li, S. Low, and J. Doyle, “Cross-layer optimization in TCP/IP networks,” *IEEE/ACM Transactions on Networking*, vol. 13, no. 3, pp. 582–595, 2005. 33

[WPL03] W. Wang, M. Palaniswami, and S. Low, “Optimal flow control and routing in multi-path networks,” *Perform. Eval.*, vol. 52, no. 2-3, pp. 119–132, 2003. 33

[YdV06] X. Yang and G. de Veciana, “Performance of peer-to-peer networks: Service capacity and role of resource sharing policies.” *Perform. Eval.*, vol. 63, no. 3, pp. 175–194, 2006. 16, 77

[YLE04] C. K. Yeo, B. S. Lee, and M. H. Er, “A survey of application level multicast techniques,” *Computer Communications*, vol. 27, no. 15, pp. 1547–1568, Sep. 2004. 13

[ZGC03] D. Zhu, M. Gritter, and D. Cheriton, “Feedback based routing,” *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 71–76, 2003. 33

[Zim05] S. Zimmermann, “Congestion pricing as scalable, efficient and stable congestion control for future IP networks,” Ph.D. dissertation, Hamburg University of Technology, Germany, 2005. 22, 41, 43, 60

[ZK02] S. Zimmermann and U. Killat, “Resource marking and fair rate allocation,” in *Proc. ICC 2002, New York*, vol. 2, Apr. 2002, pp. 1310–1314. 8, 19, 41, 43, 107

[ZLCS08] X. Zhang, D. Liu, S. Chen, and R. Sandhu, “Towards digital rights protection in BitTorrent-like P2P systems,” in *Multimedia Computing and Networking 2008*, Jan. 2008. 18