2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing

Transport Optimization in Peer-to-Peer Networks

Konstantin Miller
Technische Universität Berlin
Berlin, Germany
Email: kmi@ieee.org

Adam Wolisz
Technische Universität Berlin
Berlin, Germany
Email: awo@ieee.org

***Abstract*—The peer-to-peer networking concept has revolutionized the cost structure of Internet data dissemination by making large scale content delivery with low server cost feasible. In a peer-to-peer network, the total upload capacity increases with the number of downloaders instead of staying constant as in a client-server architecture, making it highly scalable. Despite of its importance, the problem of efficient data transport in a peer-to-peer network is still an open issue, mainly due to its complex combinatorial structure. In the presented work, we formulate the problem of optimizing a peer-to-peer download with respect to its makespan (time until all peers finish downloading) as a mixed integer linear program. Other than previous studies, we consider the case of arbitrary heterogeneous uplink and downlink capacities of the peers. Moreover, we do not consider the fluid limit case but allow the file to be subdivided in finitely many chunks. On the one hand, our results allow to infer the capacity of a peer-to-peer network, providing a benchmark for performance analysis of existing peer-to-peer protocols. On the other hand, we believe that our results build a step towards the development of efficient algorithms serving as a base for the design of data transport protocols leveraging the peer-to-peer concept.**

I. Introduction

The amount of data consumed by Internet users has been growing continuously during the last decades, reaching a considerable degree. In the future, when 3D and virtual reality will become ubiquitous, the volumes will rise even much higher. In addition, certain popular content is often requested by many thousands of users in parallel. Whenever the requests are served by a fixed number of servers, the servers might become overloaded. Moreover, providing large data volumes is very expensive and thus still impossible for an individual or a small company.

Meanwhile, the concept of data dissemination based on Peer-to-Peer (P2P) communication principles has become very popular. According to several studies, the majority of Internet traffic is nowadays generated by P2P file sharing applications and this fraction is still growing [1]. The characteristic property of the concept is that the data is downloaded not only from a fixed set of servers but that users who have already downloaded parts of a file start to upload them to other interested users. Proceeding in this way prevents servers' uplinks from becoming capacity bottlenecks of the dissemination process, thus allowing for faster downloads, while at the same time decreasing costs for content providers since they do not have to pay for expensive uplinks. As a result, P2P file sharing has revolutionized the cost structure of Internet data dissemination by making large scale content delivery with low server cost feasible.

Due to their high popularity, there exists a demand for optimization of P2P data transport mechanisms in order to provide the best possible user experience, at the same time consuming as little resources as possible. As we can observe, network operators have already become quite concerned about minimizing costs caused by P2P traffic within their networks and especially on peering links [2].

A lot of research activities have been concentrated on the concept of P2P networking leading to a number of algorithms and protocols that were proposed, analyzed and implemented, see, e.g., [3]–[5] and references therein. Still, despite of its importance, the problem of efficient data transport in a P2P network has only partially been solved, the reason for it being mainly its complex combinatorial structure. Many studies consider only heuristic approaches for download strategies; moreover, performance analysis is typically limited to comparing one system relative to another and is usually realized by means of simulations and measurements. Few researchers have used analytical models. To the best of our knowledge, a rigorous and systematic analytical treatment is still missing.

In our work, we consider the problem of optimizing a P2P download as a special type of a scheduling problem with precedence constraints. We are given a file subdivided into parts, called chunks. A set of peers is interested in downloading the file. Initially, each peer owns a subset of the chunks. Note that this setting also includes the case where initially one peer, playing the role of a server, has the whole file, while all other peers want to download it. Since each peer can retrieve a chunk from many different sources, choosing one particular source for each of the chunks is one part of the download strategy. Another part is the order in which chunks are downloaded. Here, precedence constraints come into play since a peer can upload a certain chunk only after he has downloaded it himself. Finally, each peer has to chose a download rate for each part of the file, which may also vary with the time. The rates have to be chosen such that they satisfy the capacity constraints of the network.

The contribution of the presented work is a formulation

1066-6192/11 $26.00 © 2011 IEEE
DOI 10.1109/PDP.2011.26

of the makespan-optimal P2P download as a Mixed Integer Linear Program (MIP). Makespan is the time until all peers finish their downloads. Different from existing studies, we allow for arbitrary heterogeneous upload and download capacities of the peers. Further, we do not restrict ourselves to the fluid limit case. Instead, we consider a file subdivided in finitely many chunks. Additionally, we consider initial conditions, where each peer owns an arbitrary subset of chunks. We remark that the problem that we study is considered an open issue in [6].

The outline of this paper is as follows. In Section II, we provide an overview of related work. In Section III, we present our system model. Section IV formulates the problem of finding a makespan-optimal P2P download schedule as a MIP. Section V discusses the applicability of results and names some items of ongoing work. Finally, Section VI concludes the paper.

II. Related Work

Mundinger, Weber and Weiss [6] study a setting, where uplink capacities are constrained while downlink capacities are unlimited. Further, they assume that the uplink capacity of a peer is equally shared among his concurrent uploads. Given that, they assume w.l.o.g. that each peer carries out one single upload at a time, which considerably simplifies the analysis. The authors find a makespan-optimal download schedule for integer upload capacities by solving a series of MIPs.

In contrast, we allow both upload and download capacities to be arbitrarily constrained. Although many access technologies, including, e.g., Asymmetric Digital Subscriber Line (ADSL), which is currently the most common access technology in Germany, offer a much higher download than upload rate, the situation is likely to change in the future. Moreover, traffic from other applications, like, e.g., streaming services, might consume a large portion of the downlink bandwidth of a peer. Further, we do not assume the uplink capacity of a peer to be equally shared among his uploads, since this assumptions may lead to a suboptimal makespan. Finally, our solution can be obtained solving a single MIP.

The study in [6] also provides an analytical expression for the makespan in the fluid limit case for a symmetric scenario, where each peer has a file to be shared with all other peers. Fluid limit means here that each file is subdivided into infinitely many chunks. Although fluid limit models have the advantage of better analytical tractability and often provide elegant solutions, in praxis, communication overhead as well as memory and CPU consumption of peers increase with the number of file parts making fluid limit models unrealistic. Further studies analyzing fluid limit models include [7]–[10].

Several studies, including [11], omit precedence constraints, thus assuming that a peer always have enough "new" data to share with other peers, which might not hold in reality. Occasionally, a peer will only have parts of the file that no other peer is currently interested in.

The tradeoff between performance and fairness is studied in [11]. Other related work include [12].

III. System Model

In this section, we present our system model. We first describe our basic setting. Then, we introduce and formally define the notions of a *Peer-to-Peer Download Problem (PDP)*, and of a general solution to it, a *Peer-to-Peer Download Schedule (PDS)*. Since the class of PDSs exhibits a structure that does not allow to apply tools from mathematical programming in order to optimize with respect to a certain metric of interest, we further define the class of *Piecewise Constant Peer-to-Peer Download Schedules (PCPDSs)*, which is a subclass of PDSs. In Section IV we will then show, that a makespan-optimal schedule can already be found within the simpler class of PCPDSs.

A. Basic setting

We are given a file subdivided in $m \geq 1$ equally sized chunks and $n \geq 2$ peers interested in downloading this file. We use index $k \in M := \{1, \ldots, m\}$ to denote chunks and indexes $i, j \in N := \{1, \ldots, n\}$ to denote peers. We assume that at the beginning of the download, each peer i owns a subset $M_i \subseteq M$ of chunks (which can also be empty). We use a boolean variable $m_{ik} \in \{0, 1\}$ to indicate if $k \in M_i$.

Each peer i is connected to the core network by an access link with uplink capacity $c_i^{up} \in \mathbb{R}_+$ and downlink capacity $c_i^{down} \in \mathbb{R}_+$. The unit of capacity is number of chunks per unit time, which is well-defined since we assume that all chunks have equal size.

We assume that each peer is able to perform multiple downloads and uploads simultaneously. We further assume, that a peer does not download parts of one chunk from different sources. For simplicity of the model, we do not forbid preemptive downloads, that is, a peer can pause the download of a particular chunk and resume it later.

Throughout the paper, we make the assumption that (i) the core network provides connectivity between all pairs of peers, and that (ii) the bottlenecks of all paths between peers are the access links. With these assumptions, the core network can be reduced to one single node resulting in a star-like topology as in Figure 1. Recently, due to the increase of capacity of access networks, capacity shortage in the backbone networks has become a genuine possibility. In our future work, we intend to extend our results by explicitely modeling the core network. This will also allow us to study effects of in-network caches and the problem of their optimal placement.

B. Peer-to-peer download problem

A download problem in a peer-to-peer network is, in our setting, fully described by a set of peers, a set of chunks, the

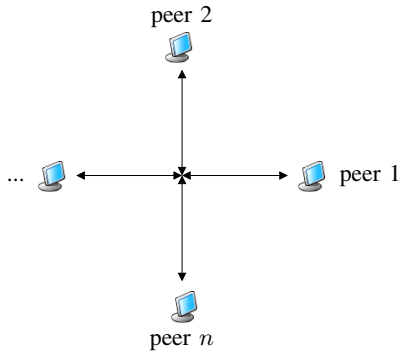

Figure 1. Assuming that the core network provides connectivity between all pairs of peers and that the bottlenecks of all paths are access links results in a simplified network graph.

initial ownership of chunks, and the access link capacities. The following definition formalizes this notion.

Definition 1 (Peer-to-Peer Download Problem (PDP)): A PDP P is a tuple

$$P = \left(M,\, N,\, \{M_i\}_{i\in N},\, \{c_i^{up}\}_{i\in N},\, \{c_i^{down}\}_{i\in N}\right). \quad (1)$$

Remark 1: Without loss of generality, we assume in the following that (i) each chunk is initially available at at least one peer, whose upload capacity is non-zero, and (ii) that each peer, who does not initially own the whole file, has non-zero download capacity. Problems, where one of these assumptions is violated, represent no challenge since they are obviously not solvable.

C. Peer-to-peer download schedule

We call a valid solution to a PDP a *Peer-to-Peer Download Schedule (PDS)*. A download schedule must contain information about (i) who downloads which chunk from whom, (ii) when, and (iii) at which rate. In order to formally define a PDS, we introduce the following three sets of variables.

Source selection variables $b_{ikj} \in \{0,1\},\ i,j \in N,\ k \in M$, indicate if peer i downloads chunk k from peer j. We define $b := (b_{ikj},\ i,j \in N,\ k \in M)$.

Rate functions $x_{ikj} : \mathbb{R}_+ \to \mathbb{R}_+,\ t \mapsto x_{ikj}(t),\ i,j \in N,\ k \in M$, indicate for a destination peer i and a source peer j the download rate of chunk k at time t. We define $x(t) := (x_{ikj}(t),\ i,j \in N,\ k \in M)$. We assume that rate functions are piecewise continuous with a finite number of discontinuities. This is a feasible assumption since the rates are controlled by a technical process, which is not capable of generating an infinite number of discontinuities.

Availability functions $a_{kj} : \mathbb{R}_+ \to \{0,1\},\ t \mapsto a_{kj}(t),\ i,j \in N,\ k \in M$, indicate if chunk k is available at peer j at time t. We define $a(t) := (a_{kj}(t),\ k \in M,\ j \in N)$. W.l.o.g., we assume that availability functions have the shape

$$a_{kj}(t) = \begin{cases} 0 \text{ for } t < \tau_{kj} \\ 1 \text{ for } t \geq \tau_{kj} \end{cases}$$

for a $\tau_{kj} \in \mathbb{R}_+$. This assumption is clearly justified since if a peer owns a chunk at a certain point in time t_0, it still owns it at all points in time $t > t_0$.

These three sets of variables are sufficient to completely describe a PDS. In order to ensure feasibility, the following definition provides additional constraints.

Definition 2 (Peer-to-Peer Download Schedule (PDS)): A PDS for a PDP P, as defined in (1), is a triple (b, x, a) satisfying the following constraints.

$$\sum_{j\in N} b_{ikj} = 1 - m_{ik},\ \forall i \in N,\ \forall k \in M\,. \quad (C1)$$

$$\int_0^\infty x_{ikj}(t)\,dt = b_{ikj},\ \forall i,j \in N,\ \forall k \in M\,. \quad (C2)$$

$$x_{ikj}(t) \leq a_{kj}(t) \leq m_{jk} + \sum_{\eta\in N} \int_0^t x_{jk\eta}(\tau)\,d\tau\,, \quad \forall i,j \in N,\ \forall k \in M,\ \forall t \in \mathbb{R}_+\,. \quad (C3)$$

$$\sum_{k\in M}\sum_{j\in N} x_{ikj}(t) \leq c_i^{down},\ \forall i \in N,\ \forall t \in \mathbb{R}_+\,. \quad (C4)$$

$$\sum_{k\in M}\sum_{i\in N} x_{ikj}(t) \leq c_j^{up},\ \forall j \in N,\ \forall t \in \mathbb{R}_+\,. \quad (C5)$$

Constraint (C1) ensures twofold. First, if a peer does not own a chunk initially, he has to download it. Second, a peer must not download parts of one chunk from different peers. Constraint (C2) ensures that each peer completely downloads each chunk that he does not own initially. Constraint (C3) ensures that a peer i downloads chunk k from a peer j only after chunk k is available at peer j. Finally, uplink and downlink capacity constraints of the peers are expressed by (C4) and (C5).

Remark 2: Note that the set of PDSs for a PDP P is never empty. Indeed, we can immediately provide the following simple PDS. For each chunk k, find any peer i who owns that chunk, and let all other peers that do not own it, download from peer i in an arbitrary order, at rates that satisfy the capacity constraints.

D. Piecewise constant download schedule

In practice, we would like to select PDSs that are efficient with respect to certain metrics, like, e.g., average download time, makespan (time required until all peers finish downloading), fairness of download times, link utilization, etc. Due to the general shape of rate functions in a PDS, the set of PDSs for a given PDP cannot be described as a subset of a finite-dimensional vector space. Thus, techniques from mathematical programming cannot be applied to select efficient PDSs.

In this section, we will define a subclass of PDSs, the *Piecewise Constant Peer-to-Peer Download Schedules (PCPDSs)*. In Section IV, we will then show at the example

of a particular metric, the makespan, that it is sufficient to optimize over schedules from this class. Finally, we will formulate the problem of finding a makespan-optimal PCPDS as a MIP.

The only difference between a PDS and a PCPDS is that in a PCPDS peers are allowed to adapt their download rates only a finite number of times $T \in \mathbb{N}$. In between, the rates remain constant. In order to model this scenario, we introduce a set of new variables $\Delta_t \in \mathbb{R}_+$, $t \in \{1,\dots,T\}$, defining the duration of the t-th time interval, during which no rate adaptation takes place. We define $\Delta := \big(\Delta_t, t \in \{1,\dots,T\}\big)$. The t-th time interval is thus given by $T_t := \Big[\sum_{\tau=1}^{t-1} \Delta_\tau,\ \sum_{\tau=1}^{t} \Delta_\tau\Big]$.

Since rate functions $x_{ikj}(t)$ are now restricted to the set of piecewise constant functions with at most T discontinuities, we can replace them by rate vectors $\big(\tilde{x}_{ikjt},\ t \in \{1,\dots,T\}\big)$. For convenience, instead of defining a rate, each variable $\tilde{x}_{ikjt}$ defines the fraction of chunk k downloaded by peer i from peer j during the time T_t. We define $\tilde{x} := \big(\tilde{x}_{ikjt}, i,j \in N, k \in M, t \in \{1,\dots,T\}\big)$.

Finally, we replace availability functions $a_{kj}(t)$ by binary availability variables, $\tilde{a}_{kjt} \in \{0,1\}$, $t \in T$, indicating if chunk k is available at peer j during the time T_t. Again, we define $\tilde{a} := \big(\tilde{a}_{kjt}, k \in M, j \in N, t \in \{1,\dots,T\}\big)$.

The following definition formalizes the notion of a PCPDS. Similar to the definition of a PDS, it provides constraints ensuring feasibility.

Definition 3 (PCPDS): A PCPDS for a PDP P, as defined in (1), is a tuple $(b,\tilde{x},\tilde{a},\Delta,T)$ satisfying constraint (C1) together with the following constraints.

$$\sum_{t=1}^{T} x_{ikjt} = b_{ikj},\ \forall i,j \in N,\ \forall k \in M\,. \tag{C2.1}$$

$$\begin{aligned} x_{ikjt} \le a_{kjt} \le \sum_{\eta \in N} \sum_{\tau=1}^{t-1} x_{jk\eta\tau} + m_{jk}\,, \\ \forall i,j \in N,\ \forall k \in M,\ \forall t \in \{1,\dots,T\}\,. \end{aligned} \tag{C3.1}$$

$$\sum_{k \in M} \sum_{j \in N} x_{ikjt} \le c_i^d\,\Delta_t,\ \forall i \in N,\ \forall t \in \{1,\dots,T\}\,. \tag{C4.1}$$

$$\sum_{k \in M} \sum_{i \in N} x_{ikjt} \le c_j^u\,\Delta_t,\ \forall j \in N,\ \forall t \in \{1,\dots,T\}\,. \tag{C5.1}$$

For the description of the constraints, see Section III-C.

Since the constraints (C1), (C2.1), (C3.1), (C4.1), (C5.1) describe a convex and compact subset of a finite-dimensional vector space, we have now paved the way to formulating the selection of an optimal download schedule as a MIP, provided that we can prove that it is sufficient to optimize over the class of PCPDSs instead of the more general class of PDSs, and provided that we can formulate the objective function as a linear function of the introduced variables. For a particular metric, the makespan, we present how to achieve both in the next section.

IV. Makespan Minimization

In Section III we formalized the notion of a PDP and defined two classses of solutions to it – PDSs and PCPDSs. Due to the structure of the class of PDSs, it is not possible to apply techniques from mathematical programming in order to optimize with respect to a certain metric of interest.

In this section, we first show at the example of a particular metric, the makespan, that it is possible to find an optimum solution among the much smaller class of PCPDSs. Afterwards, using this result, we will formulate the problem of finding a makespan-optimal download schedule as a MIP.

To begin, we define the makespan of a PDS (b,x,a) for a PDP P in a straightforward manner as

$$\begin{aligned} &T_{makespan}(b,x,a;P) := \\ &\min\{t \in \mathbb{R}_+ \mid x_{ikj}(\tau) = 0,\ \forall \tau > t,\ \forall i,j \in N,\ \forall k \in M\}\,. \end{aligned}$$

Note that the minimum always exists since $x_{ikj}(t)$ are piecewise continuous and have a finite number of discontinuities.

Similarly, for a PCPDS $(b,\tilde{x},\tilde{a},\Delta,T)$, we define the makespan as

$$T_{makespan}(b,\tilde{x},\tilde{a},\Delta,T;P) := \sum_{t=1}^{T} \Delta_t\,.$$

In the following, we restrict ourselves to schedules with a finite makespan. We can do it w.l.o.g. since for each PDP P such a schedule exists (see Remark 2), and since our goal is to find the schedule with a minimum makespan.

We are now ready to define a makespan-optimal PDS.

Definition 4 (Makespan-optimal PDS): A makespan-optimal PDS for a PDP P is a PDS solving the following optimization problem

$$\begin{aligned} &\min\ T_{makespan}(b,x,a) \\ &\text{s.t.}\quad (b,x,a) \text{ is a PDS for } P\,. \end{aligned} \tag{P1}$$

In a similar way, we define a makespan-optimal PCPDS.

Definition 5 (Makespan-optimal PCPDS): A makespan-optimal PCPDS for a PDP P is a PCPDS solving the following optimization problem

$$\begin{aligned} &\min\ T_{makespan}(b,\tilde{x},\tilde{a},\Delta,T;P) \\ &\text{s.t.}\quad (b,\tilde{x},\tilde{a},\Delta,T;P) \text{ is a PCPDS for } P\,. \end{aligned} \tag{P2}$$

As we already argued, problem (P1) cannot be solved directly by means of mathematical programming. The constraints of the problem (P2), however, that is, constraints (C1), (C2.1), (C3.1), (C4.1), and (C5.1) already conform to a MIP formulation. It remains (i) to show that the solution of (P2) is not larger than that of (P1), and (ii) to reformulate the objective function of (P2) such that it is a linear function of the defined variables. In particular,

to reformulate the objective function, we need an upper bound for the number of time intervals T, during which the rates of all peers remain constant. Both items are accounted for by the following Theorem 1. It shows that each PDS can be transformed into a PCPDS with $T = nm$, without altering the makespan. This implies that in order to find a makespan-optimal PDS, it is sufficient to consider PCPDSs with $T = nm$.

Theorem 1: Each PDS (b, x, a) can be transformed into a PCPDS $(b, \tilde{x}, \tilde{a}, \Delta, T)$, with $T = nm$, such that $T_{makespan}(b, \tilde{x}, \tilde{a}, \Delta, T) = T_{makespan}(b, x, a)$.

See Appendix A for the proof.

This observation gives rise to the following MIP formulation of the makespan-optimal download problem.

$$\min_{(b,\tilde{x},\tilde{a},\Delta,T)} \sum_{t=1}^{T} \Delta_t \tag{P3}$$
$$\text{s.t.} \quad (b, \tilde{x}, \tilde{a}, \Delta, T) \text{ is a PCPDS for the PDP } P\,,$$
$$T = nm\,.$$

Again, $(b, \tilde{x}, \tilde{a}, \Delta, T)$ is a PCPDS if constraints (C1), (C2.1), (C3.1), (C4.1), and (C5.1) are fulfilled, see Definition 3.

Finally, the following theorem shows the existence of a makespan-optimal PDS.

Theorem 2 (Existence of a makespan-optimal PDS): For a PDP, there always exists a makespan-optimal PDS.

See Appendix B for the proof.

Remark 3: Note that the formulation of a makespan-optimal PDS as a solution to a MIP still does not give us insight about the complexity of the problem. There might or might not exist a polynomial time algorithm to solve it. Complexity analysis and design of efficient algorithms is subject of ongoing work.

V. Discussion

Important differences between our work and other studies include (i) that we do not consider the fluid limit approximation (that is, we assume a finite number of chunks), and (ii) that we allow the downlink capacities to be constrained arbitrarily. Clearly, it is important to evaluate the difference between our results and existing results in realistic network topologies with constrained downlink bandwidth. This is currently subject of ongoing work.

As for applicability, one important result of our work is an upper limit on the makespan performance of a P2P system. It can serve as a benchmark to evaluate the performance of existing protocols. Further, having found an approach to calculate a makespan-optimal PDS, we are naturally interested in the possibility to design new protocols that can achieve a close-to-optimal performance.

We remark that a straightforward implementation of a protocol following our approach requires information about uplink and downlink capacities of access links of the peers, as well as chunk availability at the begin of the download, to be collected in one place. Further, after the calculation, the resulting schedule has to be communicated to all participating peers. This might be feasible in a small system, however, its scalability to large systems is limited. In order to deploy our approach in a large P2P network, it needs to be implemented in a distributed way, which is currently subject of ongoing work.

Secondly, in order to implement a PDS, as calculated by the MIP, peers have to fulfill certain requirements. E.g., they need a certain degree of sychronization. This synchronization, however, is not related to absolute time. Instead, the order of chunk downloads has to be maintained, which is easier to achieve. It can be done by exchanging messages between pairs of peers indicating that a chunk download is finished, which in turn triggers the begin of a new download.

Thirdly, since a PDS incorporates a certain rate allocation, peers need to be able to control the allocation of their available bandwidth to their flows, which is clearly not achievable with Transmission Control Protocol (TCP).

Finally, the optimization calculates a schedule under the assumption that the peers remain online until all peers complete the download and that all uplink and downlink capacities remain constant. In reality, peers might join and leave the network at random, and also, capacities might change due to background traffic or dynamic characteristics of the communication channel, e.g., a wireless link. This means that the peers require certain mechanisms to adapt to such changes. Up to a certain degree, the adaptation can take place locally. At a certain point, however, a recalculation of the whole strategy might become necessary, including a recollection of information about available bandwidth and availability of the chunks, as well as a redistribution of the new schedule.

VI. Conclusion

In the presented study, we formulated the problem of finding a makespan-optimal peer-to-peer download schedule as a mixed integer linear program. This formulation allows to infer the capacity of peer-to-peer networks, providing a benchmark for existing protocols. Further, it provides important insights serving as a base for the design of new data transport protocols leveraging the peer-to-peer concept.

As for the modeling part, our work opens up several questions. Can the presented formulation be extended to incorporate other metrics like average download time and fairness? What is the complexity of finding an optimum schedule? If it cannot be found within polynomial time, can we design a good approximation algorithm? Naturally, peer-to-peer networks expose a high level of churn, that is, joining and leaving peers. Can this dynamic be accounted for in the model? And finally, is it possible to account for

bottlenecks within the core network? This would allow to study effects of in-network caches and the problem of their optimal placement. Further items include the possibility to jointly optimize the parallel download of multiple files.

Considering optimization of existing protocols and the design of new ones, an item for future work is an analysis of how close to the optimum are the existing protocols like BitTorrent. And, of course, the design of close-to-optimal peer-to-peer transport protocols.

ACKNOWLEDGMENT

The authors would like to thank Martin Skutella and Tobias Harks for valuable discussions.

APPENDIX

A. Proof of Theorem 1

Proof: We are given a download schedule (b, x, a) for a download problem P. To formulate the proof, we define the set of time instants $\mathcal{T}_E$ when a peer finishes a chunk download, that is, $\mathcal{T}_E$ contains all time instants t such that there exists at least one rate function x_{ikj} with $t = \sup\{\tau \geq 0 \,|\, x_{ikj}(\tau) > 0\}$. In a similar way, we define $\mathcal{T}_S$ to be the set of time instants when a download starts.

Consider the set $\mathcal{T} := \mathcal{T}_S \cup \mathcal{T}_E$. W.l.o.g., we can assume $0 \in \mathcal{T}$. Let us sort all $t \in \mathcal{T}$ in an ascending order $t_1 < t_2 < \cdots < t_{|\mathcal{T}|}$. Let us now define new rate functions

$$x'_{ikj}(t) := \frac{1}{t_{l+1} - t_l} \int_{t_l}^{t_{l+1}} x_{ikj}(\xi)\, d\xi\,, \quad \forall\, t \in [t_l,\, t_{l+1}),\ 1 \leq l < |\mathcal{T}|\,.$$

We argue that the tuple (b, x', a) is again a PDS. Constraints (C1) and (C3) are still valid since we did not alter the selection of the sources for chunk downloads nor the availability of chunks. Constraint (C2) is valid since

$$\int_{t_l}^{t_{l+1}} x'_{ikj}(t)\, dt = \int_{t_l}^{t_{l+1}} \frac{1}{t_{l+1} - t_l} \int_{t_l}^{t_{l+1}} x_{ikj}(\xi)\, d\xi\, dt$$
$$= \int_{t_l}^{t_{l+1}} x_{ikj}(t)\, dt\,,\ 1 \leq l < |\mathcal{T}|,\ \forall i, j \in N,\ \forall k \in M\,.$$

To see the validity of constraint (C4), consider an interval $[t_l,\, t_{l+1}]$ for an $1 \leq l < |\mathcal{T}|$. We know that

$$\sum_{k \in M} \sum_{j \in N} x_{ikj}(t) \leq c_i^{down}, \quad \forall i \in N,\ \forall t \in [t_l,\, t_{l+1}]\,.$$

This, however, implies

$$\sum_{k \in M} \sum_{j \in N} \int_{t_l}^{t_{l+1}} x_{ikj}(t)\, dt \;\leq\; c_i^{down}\,(t_{l+1} - t_l)$$

and so

$$c_i^{down} \geq \sum_{k \in M} \sum_{j \in N} \frac{1}{t_{l+1} - t_l} \int_{t_l}^{t_{l+1}} x_{ikj}(t)\, dt$$
$$= \sum_{k \in M} \sum_{j \in N} x'_{ikj}(t)\,.$$

In a similar way, we can prove the validity of constraint (C5).

Note that we have constructed a new PDS with piecewise constant rate functions, having the same makespan as the original PDS. Now we will show that it is possible to shift the starting points of the downloads so that a download might begin only at a time instant when another download ends. Consider a $t_l \in (\mathcal{T}_S \setminus \mathcal{T}_E)$, $t_l \neq 0$, such that $t_{l-1} \in \mathcal{T}_E$. Since $t_l \in \mathcal{S}$, we have $l < |\mathcal{T}|$. Consider the interval $I := [t_{l-1},\, t_{l+1}]$. Similar as above, we define new rate functions

$$\tilde{x}_{ikj}(t) := \frac{1}{t_{l+1} - t_{l-1}} \int_{t_{l-1}}^{t_{l+1}} x'_{ikj}(\xi)\, d\xi,\ \forall\, t \in [t_{l-1},\, t_{l+1})\,.$$

This definition implicitely provides us with new chunk availability functions $\tilde{a}$. Note, however, that the chunk availability constraint is not violated since no download had finished during $[t_{l-1},\, t_l]$ and thus all chunks available at time t_l are also available at t_{l-1}. Applying this procedure iteratively, we obtain a new PDS $(b, \tilde{x}, \tilde{a})$ with piecewise constant rate functions, whose total number of discontinuities does not exceed nm. Defining Δ to contain the time intervals between the elements of $\mathcal{T}_E$, we obtain a PCPDS $(b, \tilde{x}, \tilde{a}, \Delta, T)$, which proves the claim. ∎

B. Proof of Theorem 2

Proof: As proved by Theorem 1, a makespan-optimal PDS can already be found within the class of PCPDSs by solving (P3). However, each PCPDS is obviously also a PDS. Since the set of feasible solutions for (P3) is compact and the objective function continuous, (P3) admits a solution. ∎

REFERENCES

[1] K. Cho, K. Fukuda, H. Esaki, and A. Kato, "The Impact and Implications of the Growth in Residential User-to-User Traffic," in *Proc. ACM SIGCOMM*, 2006.

[2] D. R. Choffnes and F. E. Bustamante, "Taming the Torrent: A Practical Approach to Reducing Cross-ISP Traffic in Peer-to-Peer Systems," *ACM Computer Communication Review*, vol. 38, pp. 363–374, 2008.

[3] E. 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 and Tutorials*, vol. 7, pp. 72–93, 2005.

[4] S. Androutsellis-Theotokis and D. Spinellis, "A Survey of Peer-to-Peer Content Distribution Technologies," *ACM Computing Surveys*, vol. 36, pp. 335–371, 2004.

[5] J. Li, "On Peer-to-Peer (P2P) Content Delivery," *Peer-to-Peer Networking and Applications*, vol. 1, pp. 45–63, 2008.

[6] J. Mundinger, R. R. Weber, and G. Weiss, "Optimal Scheduling of Peer-to-Peer File Dissemination," *Journal of Scheduling*, vol. 11, pp. 105–120, 2008.

[7] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks," in *Proc. ACM SIGCOMM*, 2004.

[8] R. Kumar and K. W. Ross, "Peer Assisted File Distribution: The Minimum Distribution Time," in *Proc. IEEE HOTWEB*, 2006.

[9] M. Mehyar, W. Gu, S. H. Low, M. Effros, and T. Ho, "Optimal Strategies for Efficient Peer-to-Peer File Sharing," in *Proc. ICASSP*, 2007.

[10] G. M. Ezovski, A. Tang, and L. L. H. Andrew, "Minimizing Average Finish Time in P2P Networks," in *Proc. IEEE INFOCOM*, 2009.

[11] B. Fan, J. C. S. Lui, and D.-M. Chiu, "The Design Trade-Offs of BitTorrent-Like File Sharing Protocols," *IEEE/ACM Transactions on Networking*, vol. 17, pp. 365–376, 2009.

[12] X. Yang and G. de Veciana, "Service Capacity of Peer to Peer Networks," in *Proc. IEEE INFOCOM*, 2004.