Peer-to-Peer Netw. Appl.
DOI 10.1007/s12083-012-0190-2

The performance and locality tradeoff in bittorrent-like file sharing systems

**Wei Huang · Chuan Wu · Zongpeng Li ·
Francis C.M. Lau**

Received: 29 December 2011 / Accepted: 28 November 2012

Abstract In recent years, the surge of large-scale peer-to-peer (P2P) applications has brought huge amounts of P2P traffic, which has significantly changed the Internet traffic pattern and increased the traffic-relay cost at the Internet Service Providers (ISPs). To alleviate the stress on networks, methods of localized peer selection have been proposed that advocate neighbor selection within the same network (AS or ISP) to reduce the cross-ISP traffic. Nevertheless, localized peer selection may potentially lead to the downgrade of download speed at the peers, rendering a non-negligible tradeoff between the download performance and traffic localization in the P2P system. Aiming at effective peer selection strategies that achieve any desired Pareto optimum in face of the tradeoff, our contributions in this paper are three-fold: (1) We characterize the performance and locality tradeoff as a multi-objective *b*-matching optimization problem. In particular, we first present a generic weighted *b*-matching model that characterizes the tit-for-tat in BitTorrent-like peer selection. We then introduce multiple optimization objectives into the model, which effectively characterize the performance and locality tradeoff using simultaneous objectives to optimize. (2) We design fully distributed peer selection algorithms that can effectively approximate any desired Pareto optimum of the global multi-objective optimization problem, which represents a desired tradeoff point between performance and locality in the entire system. (3) Taking network dynamics into consideration, we further propose practical protocols that allow each peer to dynamically adjust its peer selection preference on download performance or traffic locality, in order to adapt to the current quality of peering connections, while guaranteeing that the desired tradeoff is still achieved over its entire download process. To support our models and protocols, we have conducted rigorous analysis, extensive simulations, and prototype experiments under various practical settings extracted from real-world traces.

Keywords Peer-to-peer · File sharing · Traffic locality · Multiple-objective optimization

1 Introduction

In recent years, we have witnessed a surge in the number of peer-to-peer (P2P) applications over the Internet, which enables large-scale content distribution at cheap server cost (e.g., BitTorrent [1], PPLive [2], Skype [3]). The prosperousness of P2P applications has brought about huge amounts of P2P traffic [4] which repeated reaches approximately 70 % of the total broadband traffic in the year of 2007. P2P applications have also significantly changed the traffic pattern in the Internet and dramatically increased traffic-relay cost at the Internet Service Providers (ISPs). Such a cost threat has led to ISPs packet filtering and rate throttling towards P2P traffic [5], while on the other hand

W. Huang (✉)
University of Toronto, Toronto, ON, Canada
e-mail: wh.huang@mail.utoronto.ca

C. Wu · F. C. M. Lau
University of Hong Kong, Hong Kong, China

C. Wu
e-mail: cwu@cs.hku.hk

F. C. M. Lau
e-mail: fcmlau@cs.hku.hk

Z. Li
University of Calgary, Calgary, AB T2N 1N4, Canada
e-mail: zongpeng@ucalgary.ca

Published online: 10 January 2013

P2P application providers react by encryption and dynamic ports to avoid being identified [6]. There have recently emerged hot arguments that such a conflict cannot lead to desirable outcomes for both parties.

We are optimistic to expect and assume the war between P2P users and ISPs will finally end up with an agreement: ISPs will stop filtering traffic of all P2P applications and in return, users will agree to help ISPs reducing inter-ISP traffic cost by using specific P2P protocols. To achieve this goal, researchers have proposed traffic localization designs that connect peers to nearby (local) neighbors in terms of delay, routing hop count, etc., by approaches at either the P2P application side [7–9] or ISP side [10, 11], or based on collaborations between both parties [12, 13]. While such localized peer selection is effective in reducing P2P traffic across network boundaries according to measurement results [14], the selection strategies may not always be aligned with performance optimization, e.g., download speed maximization, which is targeted by BitTorrent-like systems. The localization of peer selection may unfavorably degrade the downloading performance at peers in a BitTorrent-like file-sharing system [15] when the number of cross-ISP links is limited, as local peers may not necessarily be ones that can supply large upload bandwidths. In another word, a non-negligible tradeoff may exist between file download performance and traffic localization in the system, i.e., peer selection biased towards minimizing network cost may meanwhile unfavorably degrade the downloading performance at peers.

Given such a realistic situation, an agreement of a specific peer selection protocol is desired which respects both sides of users and ISPs. A practical solution for the benefits of both the P2P application and the ISPs, is to achieve a desired tradeoff point between performance and locality in the P2P system, that is acceptable and possibly decided by both parties, such that they all will honestly execute the protocol and otherwise, be punished somehow. Intriguing questions thus arise: How can one formally characterize such a tradeoff between performance and locality? How can one design effective and fully decentralized peer selection strategies, that achieve any desired tradeoff in a practical dynamic system?

To address these challenges, we novelly characterize the performance and locality tradeoff in peer selection as a multi-objective b-matching optimization problem, with the two objectives of download speed maximization and network cost minimization simultaneously. We also design effective peer selection strategies that approach any pre-set Pareto optimum (the tradeoff point) of the global multi-objective optimization problem in a fully distributed fashion, in a dynamic P2P network.

We summarize the main contributions of this paper's work as follows:

- We present a generic weighted b-matching model to characterize tit-for-tat (TFT) peer selection in BitTorrent-like P2P systems.
- We novelly introduce multiple optimization objectives into the generic model, so that the performance and locality tradeoff in peer selection can be characterized as a multi-objective b-matching optimization problem. The two objectives of download speed maximization and network cost minimization are addressed simultaneously.
- We design efficient peer selection strategies, which represent fully distributed algorithms to approximate any pre-set Pareto optimum of the global multi-objective optimization problem in the entire system.
- For better adaptation to network dynamics, we extend the peer selection strategies to a practical dynamic peer selection protocol, with which each peer adjusts its peering preferences on download speed and traffic locality over time, while still guaranteeing that the pre-set Pareto optimum is achieved over its entire download period.
- We extensively evaluate our models and protocols using simulations driven by traces from the P4P project [12]. We have also implemented a prototype BitTorrent-like file sharing application and evaluated our protocols in a cluster of servers under realistic settings.

The remainder of the paper is organized as follows. We discuss related work in Section 2, present a generic weighted b-matching peer selection model in Section 3, and extend the model to a multi-objective optimization problem that characterizes the performance and locality tradeoff in Section 4. The distributed peer selection strategies to achieve Pareto optimum are discussed in Section 5, and the dynamic peer selection protocol with preference adjustment is presented in Section 6. We evaluate our models and protocols using trace-driven simulations and prototype-based experiments in Section 7. Finally, we conclude the paper in Section 8.

2 Related work

A number of proposals have emerged in recent years for P2P traffic localization, in order to reduce the ever-increasing inter-ISP P2P traffic.

Karagiannis et al. [7] point out that locality and caching can both reduce egress link usage, based on traces from an edge network. They propose a locality-aware P2P application design, where peers only download pieces of files available in their local ISPs, to save unnecessary traffic across ISP boundaries.

Bindal et al. [8] study the impact of different maximum numbers of connections a peer can make across ISP boundaries in BitTorrent networks. Similarly, Aggarwal et al. [13] study the effects of peer selection strategies based on locality information in the Gnutella network.

Choffnes et al. [16] implement a BitTorrent-like P2P application, ONO, which allows peers to select neighbors that are close to themselves based on network location derived from the CDN redirection information. Ren et al. [17] design and implement a topology-aware BitTorrent system, TopBT, where each client discovers close peers by estimating path proximity based on TCP pings and link/AS hop calculation.

From the perspective of ISPs, Saleh et al. [10] explore the potential of deploying proxy caches in different ASs to alleviate the load on the Internet backbone. Promoting collaborations between both ISPs and P2P application providers, P4P [12] presents a novel architecture by which ISPs provide P2P applications necessary information for them to make peer selection decisions, which honor network policy and the current network status simultaneously.

With respect to experimental work, Le Blond et al. [15] have evaluated the impact of peer selection locality on inter-ISP traffic volume and peer download completion time, using extensive experiments on a controlled environment with 10,000 BitTorrent peers. Large inter-ISP traffic reduction is observed, while a certain level of negative impact on peer download time is also revealed.

Cuevas et al. [18] measure BitTorrent networks from hundreds of ISPs in Europe and Unite States, and use the acquired datasets to evaluate and compare the downloading performance and inter-ISP traffic under different overlay construction mechanisms. They propose a construction mechanism, *locality only if faster* (LOIF), by which a peer switches a connection to a peer in another ISP only when downloading speed of the later is faster. They show that the ISPs can save 10–30 % bandwidth with LOIF, as compared to random peer selection.

Focusing on P2P video-on-demand (VoD) systems, Huang et al. [19] show that the traffic cross ISPs could be significantly reduced if modifications of localized peer selection are applied, based on traces from MSN video streaming.

With respect to theoretical studies, we are only aware of one work by Wang et al. [20], which models the peering and routing tussle between ISPs and P2P applications. The purpose of their game-theoretic model is to evaluate economic efficiency of inter-domain routing in a market place of two competing ISPs, and to analyze the effectiveness of different peering strategies.

Different from all these work, our work aims to mathematically characterize the tradeoff between download performance and traffic localization using matching-based optimization models, and to design fully distributed, effective peer selection algorithms to achieve any desired tradeoff between the two objectives.

For matching-based modeling of P2P systems, Mathieu et al. [21, 22] have investigated a b-matching model [23] for preference-based collaborator selection in BitTorrent. Their study focuses on analysis of the convergence speed of the matching and properties of the stabilized system. Differently, we not only model optimal peer selection into a b-matching-like multi-objective optimization problem, but also design efficient algorithms to solve the problem in a fully decentralized fashion. Moreover, we introduce the concept of weight towards performance or locality and a guideline of how to adjust it.

A preliminary version of our work appeared in [24]. This journal submission extensively extends our previous work, especially on the design and analysis of a dynamic peer selection algorithm, as well as experimental evaluations in realistic environments.

3 Maximum weight B-matching based peer selection: a generic model

A BitTorrent-like P2P file sharing network can be modeled as a directed graph $\mathcal{G} = (V, E)$ with vertices in V representing peers and edges in E connecting mutually selected peers. The tit-for-tat (TFT) mechanism, i.e., peer i uploads to peer j if and only if peer j uploads to peer i, can be represented by a pair of directed edges established between the two nodes, which is referred to as a *matching* between two nodes in the graph, i.e., $e(i, j) \in E$ and $e(j, i) \in E$.

We use binary variable x_{ji} to denote whether peer i wishes to download from peer j ($1 =$ yes, $0 =$ no), i.e., the data flows from peer j to peer i if the directed edge (j, i) is established. When $x_{ij} = x_{ji} = 1$, peer i and j both request to download from each other and there will be a matching between peer i and j in the P2P graph.

In traditional peer selections, the preference of a peer is as simple as selecting the fastest neighbors. In order to take benefits of both ISPs and P2P users into consideration, we model a preference function which combines both the download rate objective and the network cost objective. In general, we use a function $q_{ji}(x_{ji}) : \{0, 1\} \rightarrow [0, +\infty)$ to represent peer i's preference in selecting peer j to download from. A higher preference value can reflect (1) a larger upload bandwidth from peer j to peer i, or (2) lower inter-ISP traffic relay cost (better traffic localization) from peer j to peer i. A concrete preference function will be discussed in Section 4 which characterizes the performance and locality tradeoff. For now, we only need to assume that q_{ji} is non-decreasing and quasi-linear; $q_{ji}(1)$ denotes peer i's preference value in downloading from peer j and $q_{ji}(0) = 0$.

In a typical BitTorrent-like application, a peer can make a certain number of connections, e.g., 4–5 as in the BitTorrent client Vuze [25]. Let b be the maximum number of download connections each peer can establish. Let N_i denote the neighborhood of peer i containing known peers it learns from a tracker server in the BitTorrent-like system. Our peer selection problem at hand is to decide at each peer i the subset of neighbors in N_i to actually request to download from. Such a peer selection problem at peer i can be modeled into the following optimization problem, given the download requests that peer i itself has received from other peers (i.e., $x_{ij}, \forall j \in N_i$):

$$\max \sum_{j \in N_i} q_{ji}(x_{ji}) \tag{1}$$

subject to:

$$\begin{aligned} &\sum_{j \in N_i} x_{ji} \le b, \\ &x_{ji} \le x_{ij}, \forall j \in N_i, \\ &x_{ji} \in \{0, 1\}, \forall j \in N_i. \end{aligned} \tag{2}$$

The constraints in Eq. 2 characterize the TFT mechanism in a BitTorrent-like system: Only when peer i uploads to peer j upon request ($x_{ij} = 1$), would peer j possibly upload to peer i ($x_{ji} = 1$). Given neighbors' current requests $x_{ij}, \forall j \in N_i$, the optimization in Eq. 1 derives the optimal values of x_{ji}'s, $\forall j \in N_i$, at peer i, i.e., the best up-to-b neighbors that peer i will select to download from, in order to maximize its aggregate preference.

Putting all the local optimization at peers together, we obtain the following global optimal peer selection problem in the entire P2P network:

$$\max \sum_{i \in V} \sum_{j \in N_i} q_{ji}(x_{ji}) \tag{3}$$

Subject to:

$$\begin{aligned} &\sum_{j \in N_i} x_{ji} \le b, \forall i \in V, \\ &x_{ji} \le x_{ij}, \forall i \in V, j \in N_i, \\ &x_{ji} \in \{0, 1\}, \forall i \in V, j \in N_i. \end{aligned}$$

Taking $q_{ji}(1)$ as the weight associated with the directed edge (j, i) in the P2P graph, the global optimization problem in Eq. 3 is essentially a *maximum weight b-matching problem* [23]. We will propose a fully decentralized algorithm to solve the problem and achieve stable and pareto-optimal peer selection in the entire P2P network in Section 5.

4 Characterizing the performance and locality tradeoff: the multi-objective model

We now extend the generic matching-based model in the previous section to optimal peer selection that addresses the tradeoff between download performance and neighbor locality. At each peer, the *download performance* refers to its aggregate downloading rate from selected peers, and the *neighbor locality* is reflected by the overall inter-ISP traffic relay cost (referred to as *network cost* hereinafter) incurred by downloading from the selected neighbors.

4.1 Multi-objective peer selection

At each peer i, we use a non-negative constant r_{ji} to denote the average rate that peer i can download from peer j. Let c_{ji} be the non-negative network cost incurred by downloading from peer j to peer i. The cost, essentially incurred by inter-ISP charges among different ISPs, can be set by users' home ISPs. We assume the network cost between any pair of peers could be assigned based on the peering relationship of their corresponding ISPs, or using metrics such as the *p-distance* in P4P [12], which represent the network policy and the current network status.

We use a vector-valued function [26] to represent the preference function $q_{ji}(x_{ji})$ in Eq. 1: $q_{ji}(x_{ji}) = \begin{pmatrix} r_{ji}x_{ji} \\ -c_{ji}x_{ji} \end{pmatrix}$. The new objective function in peer i's neighbor selection becomes

$$\max \sum_{j \in N_i} q_{ji}(x_{ji}) = \begin{cases} \max \sum_{j \in N_i} r_{ji}x_{ji} \\ \min \sum_{j \in N_i} c_{ji}x_{ji} \end{cases},$$

which captures the tradeoff between two goals:

- ◇ *Downloading rate maximization.* Similarly to traditional peer selection strategies, each peer should maximally choose to download from neighbors with large upload bandwidth, in order to achieve the best download performance.
- ◇ *Network cost minimization.* Large P2P traffic between ISPs with high traffic relay cost is undesirable, leading to higher probability of traffic throttling. Therefore, each peer should also maximally choose neighbors

from the same ISP or ISPs with low network costs in between, in order to reduce inter-ISP traffic incurred.

The peer selection problem at peer i in Eq. 1 becomes:

$$\begin{cases} \max \sum_{j\in N_i} r_{ji}x_{ji} \\ \min \sum_{j\in N_i} c_{ji}x_{ji} \end{cases} \tag{4}$$

Subject to:

$$\sum_{j\in N_i} x_{ji} \le b,$$

$$x_{ji} \le x_{ij}, \quad \forall j \in N_i,$$

$$x_{ji} \in \{0,1\}, \quad \forall j \in N_i. \tag{5}$$

The global multi-objective optimal peer selection problem is (an extension from the global optimization problem in Eq. 3):

$$\begin{cases} \max \sum_{i\in V}\sum_{j\in N_i} r_{ji}x_{ji} \\ \min \sum_{i\in V}\sum_{j\in N_i} c_{ji}x_{ji} \end{cases} \tag{6}$$

Subject to:

$$\sum_{j\in N_i} x_{ji} \le b, \quad \forall i \in V, \tag{7}$$

$$x_{ji} \le x_{ij}, \quad \forall i \in V, j \in N_i, \tag{8}$$

$$x_{ji} \in \{0,1\}, \quad \forall i \in V, j \in N_i. \tag{9}$$

This multi-objective optimization aims to derive the best peer selection strategies in the entire P2P network, which maximize the aggregate download rates and minimize the overall network costs incurred. Nevertheless, in multi-objective optimization, *optimal* solutions which achieve all objectives concurrently do not usually exist [26], i.e., there commonly exists a tradeoff among the multiple objectives. In our optimal peer selection problem, there may not exist ideal optimal strategies and a tradeoff has to be compromised between both of our objectives. Although in reality, ISPs always are the side who set up the agreement, even about how much

In what follows, we discuss how a *Pareto optimal* solution can be derived, which achieves any desired tradeoff of the two objectives.

4.2 Pareto optimal solutions

A feasible solution to a multi-objective optimization problem is *Pareto optimal* if there is no other feasible solution that performs better than it, with respect to all objectives [26]. In our optimization in Eq. 6, feasible $\mathbf{x}^*$ is *Pareto optimal* if there does not exist feasible $\mathbf{x}$, such that $\sum_{i\in V}\sum_{j\in N_i} r_{ji}x_{ji} > \sum_{i\in V}\sum_{j\in N_i} r_{ji}x^*_{ji}$ and $\sum_{i\in V}\sum_{j\in N_i} c_{ji}x_{ji} < \sum_{i\in V}\sum_{j\in N_i} c_{ji}x^*_{ji}$.

A typical technique to find a *Pareto optimal* solution is *scalarization*, which converts the multi-objective problem into a regular optimization problem with a scalar objective function, that is the linear weighted combination of the original multiple objectives [26]. Introducing weights α_G and β_G ($\alpha_G + \beta_G = 1, \alpha_G \ge 0, \beta_G \ge 0$) to the bandwidth maximization objective and the cost minimization objective in Eq. 6, respectively, our multi-objective problem can be converted to the following:

$$\max \quad \alpha_G \sum_{i\in V}\sum_{j\in N_i} r_{ji}x_{ji} - \beta_G \sum_{i\in V}\sum_{j\in N_i} c_{ji}x_{ji} \tag{10}$$

Subject to:

$$\sum_{j\in N_i} x_{ji} \le b, \quad \forall i \in V, \tag{11}$$

$$x_{ji} \le x_{ij}, \quad \forall i \in V, j \in N_i, \tag{12}$$

$$x_{ji} \in \{0,1\}, \quad \forall i \in V, j \in N_i. \tag{13}$$

By solving the above linear program using different values of α_G and β_G, we can derive different *Pareto optimal* solutions to the multi-objective problem in Eq. 6. Therefore, given a weight pair (α_G, β_G) that represents the desired tradeoff between the two objectives, the linear program derives the *Pareto optimal* peer selection strategy that achieves the desired tradeoff. We further note that if there does exist an optimal solution to the multi-objective problem which optimizes both objectives concurrently, it can be derived by solving the linear program using any non-negative weights satisfying $\alpha_G + \beta_G = 1$ [26].

In practice, the non-negative weight pair α_G and β_G can be set by the P2P application provider or upon negotiation between the P2P provider and ISPs. We believe that in a free market, if an ISP is too mean to P2P applications which are popular among users, by setting weight toward download rate too low, it would probably lose a portion of unhappy clients. In the other hand, if users do not obey the made rules about how to use P2P applications, ISP can fine or keep blocking their traffics according to the contract anyway. Thus in a long-term view, we have reasons to expect that P2P users and ISPs will come to an agreement on the value of weight α_G and β_G, and both sides will obey the rules.

5 Multi-objective peer selection: a distributed algorithm

We design a fully decentralized algorithm that derives a stable and pareto-optimal peer selection solution to the generic optimization model in Section 3, and then apply it in multi-objective peer selection in Section 4 to achieve any desired tradeoff between performance and locality.

5.1 Peer selection based on the generic preference function

We solve the generic global optimization problem in Eq. 3 with a distributed algorithm, in which each peer i iteratively carries out optimal neighbor selection based on its local optimization in Eq. 1. The algorithm peer i carries out is given in Algorithm 1, with symbols defined in Table 1.

Algorithm 1 Peer Selection Algorithm at Peer i

```
 1: procedure REMOVE(node $i$, list $X$, node $k$)
 2:   Remove i from node k's list X(k)
 3: end procedure
 1: procedure ADDMATCHING(node $i$, node $j$)
 2:   M(i) := M(i) + {j}, Remove(j, R, i), Remove(j, P, i)
 3:   M(j) := M(j) + {i}, Remove(i, P, j), Remove(i, R, j)
 4: end procedure
 1: procedure REPLACE(node $i$, node $j$, node $k$)
 2:   Remove(k, M, i), Remove(i, M, k), AddMatching(i, j)
 3: end procedure
 1: procedure PROC1(node $i$)
 2:   pick peer j in W(i) with the highest rank
 3:   Remove(j, W, i)
 4:   send j a request to download, P(i):=P(i)+{j} (j will do R(j):=R(j)+{i})
 5: end procedure
 1: procedure PROC2(node $i$)
 2:     for each j in P(i) such that $rank(j,i) < rank(k,i)$,
        where k is a peer with the lowest rank in
    M(i) do
 3:         Remove(j, P, i), add j back to W(i)
 4:         inform j to do Remove(i, R, j)
 5:     end for
 6: end procedure
 1: procedure PROC3(node $i$)
 2:    pick peer j from R(i) with the highest rank
 3:    if j∈P(i), i.e., i has sent a download request to j
 4:      if the number of peers in matching list $|M(i)| < b$ then
 5:        AddMatching(j, i)
 6:      else if $\exists k \in M(i), rank(k,i) < rank(j,i)$ then
 7:           Replace(i, j, k), add k back to W(i)
 8:         endif
 9:      endif
10:    else if $|M(i)| < b$ or $\exists k \in M(i), rank(k,i) < rank(j,i)$
11:           send j a request to download, P(i):=P(i)+{j}
12:        endif
13: end procedure
 1: procedure MAIN PROCEDURE
 2: Get W(i) from the tracker server
 3: Repeat Line 4 – 11 until download complete
 4:   if number of peers in matching list $|M(i)| < b$ then
 5:      call Proc1(i)
 6:   else /*$|M(i)| = b$*/
 7:      call Proc2(i)
 8:   endif
 9:   if Receiving list R(i) is not empty then
10:      call Proc3(i)
11:   endif
12: end procedure
```

Table 1 Notations in Algorithm 1

N	Set of nodes in the network.
i, j, k	Nodes $i, j, k \in N$.
$W(i)$	Preference list of node i, which contains node(s) that node i obtains from a tracker server.
rank(j,i)	Rank of neighbor j in i's list (the larger the value, the more i prefers to download from j).
$P(i)$	Proposal list of node i, containing the node(s) to which node i has proposed to set up a download connection.
$M(i)$	Matching list of node i, containing the node(s) with which node i has established a match.
$R(i)$	Connection request receiving list of node i, containing the node(s) which has (have) proposed to set up a download connection with node i.

In the distributed algorithm, peer i ranks all known neighbors according to the preferences $q_{ji}(1), \forall j \in N_i$, into its preference list $W(i)$. It sends download requests to peers with the highest ranks in the preference list and adds those requested peers into its proposal list $P(i)$; at each neighbor j which receives this request, it adds i into its receiving list $R(j)$ (*Proc1*()). Peer i then waits for requests from others, and places those peers to which it has requested a download connection and from which it has received a request too (the matched peers) into its matching list $M(i)$ (*Proc3*()). Matches are also dynamically adjusted in order to achieve the best peer selection that maximizes aggregate preference at each peer: if peer i's matching list is full with b peers when it receives a request from a peer it prefers more than a matched peer, it will replace the least preferred peer in $M(i)$ with the new requesting peer; when peer i's matching list is full, it will also adjust its proposal list by withdrawing requests sent to neighbors whose preferences are lower than those of its already matched peers (*Proc2*()). The algorithm repeats at each peer until no more changes occur to its matching list.

5.2 Analysis of algorithm optimality

We next show that such a distributed iterative algorithm converges to a stable Pareto maximum weight b-matching among peers under a mild assumption, in a network without peer dynamics.

Definition 1 In an undirected graph $\mathcal{G} = (V, E)$, a stable Pareto maximum weight b-matching is a subgraph $\mathcal{M}$ of $\mathcal{G}$ which satisfies: (1) it contains all nodes in V and each node is incident with at most b edges in E; (2) no edges in $\mathcal{M}$ change any more; (3) there does no exist another b-matching $\mathcal{M}'$ in $\mathcal{G}$, in which the sum of weights (preferences) on all incident edges at each node is not smaller than that in $\mathcal{M}$.

Assumption 1 *Given the preference lists at the peers, there does not exist a preference cycle in the network, i.e., there is no such a sequence of peers, $p_0, p_1, \cdots, p_{m-1} (m \geq 3)$, such that p_i prefers $p_{(i+1) \bmod m}$ to $p_{(i-1) \bmod m}$, $i = 0, 1, \cdots, m-1$.*

Such an assumption largely holds when a peer's preferences towards different candidate neighbors are different in the P2P network. The different preference values can be achieved by introducing a small random error into the preferences derived, e.g., based on downloading rate and network cost as discussed in Section 4.1.

Theorem 1 *Under Assumption* 1, *the distributed algorithm in Algorithm* 1 *converges to a stable Pareto maximum weight b-matching in a P2P network without peer dynamics, which represents a Pareto optimal stable peer selection defined by the global optimization problem in Eq.* 3.

Proof According to Algorithm 1, a peer can establish matches with at most b peers, i.e., the number of peers in its matching list, $|M(i)|$, is no larger than b. Therefore, the subgraph with all peers and matched edges in the network is a b-matching at any given time.

We next prove that the algorithm converges to a stable b-matching.

We first prove by contradiction that there must be at least two peers ranking each other at the top of their respective preference lists in one round of the algorithm, such that at least one stable match between two peers is established. Suppose that there do not exist such two peers. We have the following: suppose that p_0 prefers p_1 most, then p_1 should not prefer p_0 most; suppose that p_1 prefers p_2 most, and then neither p_0 nor p_1 should rank top in p_2's preference list (otherwise either Assumption 1 or our above assumption is not satisfied), and p_2 prefers some other peer p_3 most; and so on. Following the similar logic, since the total number of peers in the network is a constant (n), there must still exist a preference circle, i.e., p_{n-1} prefers p_0 most, which contradicts with Assumption 1. Therefore, we have proven that at least one match between two peers is established in one round of the algorithm, which is stable and would not change in the future. Excluding the match from the network and continuing running the algorithm, more and more stable matches will be established, and eventually a stable b-matching results in the network.

Finally, we prove the Pareto optimality of the achieved stable b-matching by contradiction. We assume that there exists another stable b-matching $\mathcal{S}'$, in which the overall weight along incident edges at any node is no smaller than that of the node in the stable b-matching $\mathcal{S}^*$ that our algorithm derives, and $\mathcal{S}' \neq \mathcal{S}^*$. Therefore, there must exist peer i, whose overall weight along its matched edges is larger in $\mathcal{S}'$ than in $\mathcal{S}^*$, i.e., $\sum_{k \in M'(i)} rank(k, i) > \sum_{k \in M^*(i)} rank(k, i)$, where $M'(i)$ and $M^*(i)$ denote the set of neighbors of peer i in the two stable b-matchings, respectively. There must exist peer l in $M'(i)$ but not in $M^*(i)$ and peer j in $M^*(i)$, such that $rank(l, i) > rank(j, i)$; l and j are both on i's preference list, and i should be on both j's and l's proposal lists. In b-matching S^*, i would replace j by l based on Algorithm 1, causing changes to S^*. This contradicts the assumption that b-matching S^* is a *stable* one. Therefore, we can conclude that there exists no alternative stable b-matching that achieves a no-smaller total weight at each of the peers than that in S^*. □

To apply to the multi-objective peer selection process, Algorithm 1 can be directly applied to derive the Pareto optimal peer selection striking a desired performance and locality tradeoff, by using the following combined multi-objective preference function at each peer i:

$$q_{ji}(x_{ji}) = \alpha r_{ji} x_{ji} - \beta c_{ji} x_{ji}, \forall j \in N_i. \tag{14}$$

In particular, each peer i ranks its known neighbors using the above preference function (14) into its preference list $W(i)$, and carries out Algorithm 1 in an iterative fashion. Algorithm 1 at peer i derives its local Pareto optimal peer selection based on the local optimization in Eq. 1; the iterations of the algorithm at all peers converge to the global Pareto optimal peer selection as is the solution to Eq. 10.

6 Dynamic peer selection in practical networks

We focus on static P2P networks in the previous section. In what follows, we will propose a peer selection algorithm featuring dynamic adjustment of weights to the bandwidth maximization and cost minimization objectives in Eq. 6, which achieves Pareto optimal peer selection in practical network with volatile peer dynamics.

The basic idea of the algorithm is to allow each peer to change overtime the weights on download rate and network cost in its local Pareto optimal peer selection, as defined below. Here, α_L and β_L ($\alpha_L + \beta_L = 1, \alpha_L \geq 0, \beta_L \geq 0$) are weights in this local optimization for the bandwidth maximization objective and the cost minimization objective, respectively.

$$\max \quad \alpha_L \sum_{j \in N_i} r_{ji} x_{ji} - \beta_L \sum_{j \in N_i} c_{ji} x_{ji} \tag{15}$$

Subject to:

$$\sum_j x_{ji} \leq b, \tag{16}$$

$$x_{ji} \leq x_{ij}, \quad \forall j \in N_i, \tag{17}$$

$$x_{ji} \in \{0, 1\}, \quad \forall j \in N_i. \tag{18}$$

The purpose of adjusting weights α_L and β_L is for each peer to prioritize downloading rate or network cost in peer selection at different stages of its file download according to different availability of supplying neighbors, in order to achieve the shortest file download time while guaranteeing that a preset global weight pair (α_G, β_G) in Eq. 10 is achieved on average over its entire download period.

We first analyze the relationship between the weights and the aggregate download rate/network cost at each peer in Section 6.1, and then present the algorithm in Section 6.2.

6.1 Relation between weights and aggregate peer downloading rate/network cost

The weights in the objective function in Eq. 15 decide the tradeoff between download rate maximization and network cost minimization in each peer's neighbor selection. We now prove the relationship rigorously in the following theorem.

Theorem 2 *Let x^*_{ji} denote the optimal solution to the local optimization in Eq. 15 when $\alpha_L = \alpha_1$ and $\beta_L = 1 - \alpha_1$. Suppose $X_1 = \sum_{j \in N_i} r_{ji} x^*_{ji}$ and $Y_1 = \sum_{j \in N_i} c_{ji} x^*_{ji}$. Similarly, let x'_{ji} be the optimal solution to Eq. 15 when $\alpha_L = \alpha_2$ and $\beta_L = 1 - \alpha_2$, and suppose $X_2 = \sum_{j \in N_i} r_{ji} x'_{ji}$ and $Y_2 = \sum_{j \in N_i} c_{ji} x'_{ji}$. If $\alpha_2 > \alpha_1$ ($\beta_1 > \beta_2$), then $X_2 \geq X_1$ and $Y_2 \geq Y_1$, i.e., the aggregate download rate and the aggregate network cost at peer i are non-decreasing on α_L, when its neighbors' current requests $x_{ij}, \forall j \in N_i$ remain the same.*

Proof We first prove by contradiction that if $\alpha_2 > \alpha_1$, then $X_2 \geq X_1$. Assume the following:

(1) $\alpha_2 > \alpha_1$, i.e., $\alpha_2 = \alpha_1 + \epsilon_1$, where $\epsilon_1 > 0$,
(2) $X_2 < X_1$, i.e., $X_1 = X_2 + \epsilon_2$, where $\epsilon_2 > 0$,
(3) $Y_1 = Y_2 + \epsilon_3$.

Since X_1 and Y_1 are aggregate download rate and network cost achieved by optimal solution to the problem in Eq. 15 when $\alpha_L = \alpha_1$, we have

$$\begin{aligned} &\alpha_1 X_1 - \beta_1 Y_1 \geq \alpha_1 X_2 - \beta_1 Y_2 \\ \Rightarrow &\alpha_1 (X_2 + \epsilon_2) - \beta_1 (Y_2 + \epsilon_3) \geq \alpha_1 X_2 - \beta_1 Y_2 \\ \Rightarrow &\alpha_1 \epsilon_2 - (1 - \alpha_1)\epsilon_3 \geq 0. \end{aligned} \tag{19}$$

Similarly, because X_2 and Y_2 are aggregate download rate and network cost achieved by optimal solution to Eq. 15 when $\alpha_L = \alpha_2$, we can also derive

$$\begin{aligned} &\alpha_2 X_2 - \beta_2 Y_2 \geq \alpha_2 X_1 - \beta_2 Y_1 \\ \Rightarrow &\alpha_2 X_2 - \beta_2 Y_2 \geq \alpha_2 (X_2 + \epsilon_2) - \beta_2 (Y_2 + \epsilon_3) \\ \Rightarrow &0 \geq \alpha_2 \epsilon_2 - \beta_2 \epsilon_3 \\ \Rightarrow &(\alpha_1 + \epsilon_1)\epsilon_2 - (1 - \alpha_1 - \epsilon_1)\epsilon_3 \leq 0 \\ \Rightarrow &\epsilon_1(\epsilon_2 + \epsilon_3) \leq -[\alpha_1 \epsilon_2 - (1 - \alpha_1)\epsilon_3]. \end{aligned} \tag{20}$$

According to Eq. 19, we further have

$$\epsilon_1(\epsilon_2 + \epsilon_3) \leq 0.$$

Since $\epsilon_1 > 0$, $\epsilon_2 > 0$, we derive

$$\epsilon_3 < 0. \tag{21}$$

Therefore, $\alpha_2 \epsilon_2 > 0$, $-\beta_2 \epsilon_3 > 0$, and

$$\alpha_2 \epsilon_2 - \beta_2 \epsilon_3 > 0. \tag{22}$$

Here Eq. 22 contradicts with Eq. 20. So the assumption, that $X_2 < X_1$ if $\alpha_2 > \alpha_1$, does not hold. We have proven instead that if $\alpha_2 > \alpha_1$, $X_2 \geq X_1$.

The second part of the theorem, that if $\alpha_2 > \alpha_1$, $Y_2 \geq Y_1$, can be proven in a similar fashion, which we omit here. □

Theorem 2 shows that by adjusting weight α_L (and thus β_L) in optimal peer selection, peer i can effectively control the tradeoff between downloading rate and network cost it experiences. We will make use of this conclusion in our dynamic peer selection algorithm in Section 6.2.

On the other hand, from the perspective of the entire network, a similar relationship exists, between the global weights in the global optimization problem in Eq. 10 and the overall download rate /network cost in the network, as shown by the following theorem.

Theorem 3 *Let x^{G*}_{ji} denote the optimal solution to the global optimization in Eq. 10 when $\alpha_G = \alpha^G_1$ and $\beta_G = 1 - \alpha^G_1$. Suppose $X^G_1 = \sum_{i \in V} \sum_{j \in N_i} r_{ji} x^{G*}_{ji}$ and $Y_1 = \sum_{i \in V} \sum_{j \in N_i} c_{ji} x^{G*}_{ji}$. Similarly, let $x^{G'}_{ji}$ be the optimal solution to Eq. 10 when $\alpha_G = \alpha^G_2$ and $\beta_L = 1 - \alpha^G_2$, and suppose $X^G_2 = \sum_{i \in V} \sum_{j \in N_i} r_{ji} x^{G'}_{ji}$ and $Y^G_2 = \sum_{i \in V} \sum_{j \in N_i} c_{ji} x^{G'}_{ji}$. If $\alpha^G_2 > \alpha^G_1$ ($\beta^G_1 > \beta^G_2$), then $X^G_2 \geq X^G_1$ and $Y^G_2 \geq Y^G_1$, i.e., the overall download rate and the overall network cost in the entire network are non-decreasing on α_G.*

Theorem 3 can be proven in a similar fashion as Theorem 2. It shows that the tradeoff between download performance and inter-ISP traffic in the entire network can be effectively adjusted, by setting different global weights α_G and β_G: if the P2P provider (and the ISP through negotiation) may set a smaller α_G, less inter-ISP traffic will result, at the cost of lower download performance in the system.

6.2 Dynamic weight adjustment and peer selection algorithm

In practical BitTorrent-like networks, the progress of file download at each peer is affected by the availability of supplying neighbors overtime, as well as the availability of chunks at the peer for tit-for-tat exchanges. We can divide the download progress of a peer into three stages:

- ◇ Startup stage (S): When a peer starts to download a file, it has a limited number of available chunks of the file, and thus few neighbors may like to match and exchange chunks with this peer. In this stage, it is desirable for the peer to download chunks as fast as possible.
- ◇ Intermediate stage (I): After the peer has bootstrapped itself after obtaining a sufficient number of chunks, it comes into the intermediate download stage. In this stage, the requirement for fast download is less urgent, and more considerations should be given to reducing inter-ISP traffic in its peer selection.
- ◇ End stage (E): When the file download approaches the end, the peer may experience the "end-game mode" [27], where the last few chunks in need are difficult to locate in the network. In this stage, fast download of the chunks should be prioritized over inter-ISP traffic reduction, in order for the peer to complete its file download and start seeding as fast as possible. The latter is beneficial to the performance in the entire P2P system in the long run.

To address practical needs in different download stages, we design a dynamic weight adjustment and peer selection algorithm, to be carried out at each peer overtime. The algorithm varies local weights α_L and β_L in the peer's neighbor selection, to prioritize download rate maximization or network cost reduction at different times, while guaranteeing a preset global weight pair (α_G, β_G) is achieved on average over the entire download period; meanwhile, it periodically updates its preference/receiving/proposal lists according to dynamics of neighbors. The algorithm is given in Algorithm 2, with notations defined in Table 2.

For dynamic weight adjustment (implemented in WeightAdjustment()), we divide the entire download period of file F at peer i into M consecutive intervals, $F_1, F_2, \cdots, F_M$, each corresponding to the download of $\frac{1}{M}$ of the file. The length of each interval is different as download rates vary, while the total size of file pieces downloaded in each interval is the same.

It is difficult to decide an optimal divide of the start stage and the end stage suitable for all peers in the network, given the different situations at each peer. In practice, we can always adjust the lengths of the stages and in a relatively suitable values. We suppose the startup stage S corresponds to the download of $\frac{s}{M}$ of the file, i.e., download intervals $F_1, \cdots, F_s$, and the end stage E corresponds to the download of $\frac{e}{M}$ of the file, i.e., download intervals $F_{M-e+1}, \cdots, F_M$, where $s + e < M$. Local weight $\alpha_L = 1$ is used for peer i's neighbor selection during startup stage S and end stage E, in order for it to download at the highest speed upon new join and in the end-game mode (Lines 2–3 in WeightAdjustment()). In intermediate stage I, α_L is dynamically adjusted every interval as follows, in order to achieve the preset average weight α_G over the entire download period.

Algorithm 2 Peer Selection Algorithm with Dynamic Weight Adjustment and Churn Handling at Peer i

```
1:  procedure CurrentAverage(int v)
2:    Return $(\alpha_G M - \sum_{w=s+1}^{v-1} \alpha_w - s - e)/(M - e - v + 1)$
3:  end procedure

1:  procedure Update(int* counter, int value, list Y)
2:    if $\big($- -$(*counter) = 0\big)$, then
3:      update Y, $(*counter) := value$
4:    endif
5:  end procedure

1:  procedure WeightAdjustment(int v)
2:    if ($v \le s$ or $v \ge M - e + 1$) then
3:      $\alpha_v := 1$
4:    else then
5:      $\widehat{\alpha_G} := CurrentAverage(v)$
6:      $\alpha_v := \alpha_{v-1}$
7:        while ($\alpha_v \ge 2\widehat{\alpha_G}$) do
8:            $\alpha_v := \alpha_v/2$
9:        end while
10:       if ($\frac{X_{v-2}-X_{v-1}}{X_{v-2}} < \frac{\rho\delta}{\alpha_{v-2}}$ and $\alpha_{v-1}$ is reduced by $\delta$
      in the last interval) then
11:         $\alpha_v := \alpha_{v-1} - \delta$;
12:     endif
13:     if $\big(\alpha_v < \widehat{\alpha_G}$ or $CurrentAverage(v + 1) < 0\big)$ then
14:         $\alpha_v := \widehat{\alpha_G}$;
15:     endif
16:   endif
17: end procedure

1:  procedure Main( )
2:  Get W(i) from tracker server
3:  $counter_p := \tau_p, counter_w := \tau_w, counter_r := \tau_r$
4:  Repeat Line 5 – 18 until download complete:
5:    $v$:= index of the current download interval
6:    Call WeightAdjustment(v)
7:    if number of peers in matching list $|M(i)| < b$ then
8:        Call Proc1(i) in Algorithm 1
9:        Update(&$counter_w$, $\tau_w$, W(i))
10:   else /*$|M(i)| = b$*/
11:       Call Proc2(i) in Algorithm 1
12:       Update(&$counter_p$, $\tau_p$, P(i))
13:   endif
14:   if Receiving list R(i) is not empty then
15:       Call Proc3(i) in Algorithm 1
16:       Update(&$counter_r$, $\tau_r$, R(i))
17:   endif
18:   Record aggregate download speed $X_v$ for interval $v$
19: end procedure
```

Table 2 Notations in Algorithm 2

Notation	Description
M	The number of intervals download of file F is divided into.
F_v	The vth download interval.
s	The number of download intervals startup stage S corresponds to.
e	The number of download intervals end stage E corresponds to.
α_G	The global weight on download rate in global optimization (10).
α_v	The simplified form of $\alpha_{L(v)}$, i.e., the local weight on download rate peer i uses in its peer selection in the vth interval.
δ	The decrement of weight at the vth interval.
X_v	The aggregate download rate in the vth interval.
ρ	A parameter to judge whether the decrease of aggregate download rate from X_{v-2} to X_{v-1} is significant with the decrease of local weight from α_{v-2} to α_{v-1}.
τ_p	Interval for i to update proposal list $P(i)$.
τ_w	Interval for i to update preference list $W(i)$.
τ_r	Interval for i to update receiving list $R(i)$.

In the vth interval ($s < v < M-e+1$), we first calculate $CurrentAverage()$ (Line 5), which provides a value keeping which as its local weight until the beginning of the end stage, the peer will achieve an average weight equivalent to α_G, throughout its download period. We next judge if keeping the weight as in the previous interval α_{v-1} (we use α_{v-1} as the simplified form of $\alpha_{L(v-1)}$) to the half way between the current interval, F_v, and the last interval in the intermediate stage, $F_{(M-e)}$, and making the weights 0 for the later half of intervals in the intermediate stage, whether the resulting average $\bar{\alpha}_L = \sum_{v=1}^{M} \alpha_v/M$ would exceed the preset α_G. If so, keep reducing the weight by half (Lines 6–9). The next step is to evaluate the impact on aggregate download rate decrease due to weight reduction in the previous two rounds (Lines 10–12): if the downgrade of aggregate download rate $X_{v-2} - X_{v-1}$ at peer i is not significant as compared to the decrease of the local weight, α_v in this interval can be further reduced. Finally, we check if α_v has been reduced to be even lower than $\widehat{\alpha_G}$ or it would make the next result of $CurrentAverage()$ below 0; if so, we will make $\alpha_v = \widehat{\alpha_G}$ (Lines 13–15).

An illustration of the dynamic weight adjustment is given in Fig. 1. In this example, the download of file F is divided into 40 intervals, the startup stage and the end stage correspond to 5 intervals, respectively, and the preset global weight is $\alpha_G = 0.65$.

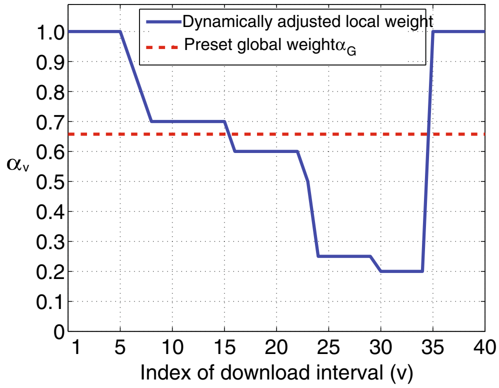


Fig. 1 An example of dynamic weight adjustment

Theorem 4 proves that the average of local weights, decided by the weight adjustment algorithm in Algorithm 2, achieves the preset global weight α_G over the file download period.

Theorem 4 *Under the conditions that $s + e \leq \alpha_G M$ and that s, e, δ, P are positive numbers, the dynamic weight adjustment in Algorithm 2 achieves $\sum_{v=1}^{M} \alpha_v/M = \alpha_G$.*

Proof

(1) We first prove that the average local weight $\sum_{v=1}^{M} \alpha_v/M \geq \alpha_G$. In procedure $WeightAdjustment()$ of Algorithm 2, after all the adjustments of α_v, we have one condition in Line 13 to check whether α_v is smaller than $\widehat{\alpha_G}$. If this is true, we make $\alpha_v = \widehat{\alpha_G}$, and this local weight will be used until the end of the intermediate stage.

The average weight becomes

$$\left(\sum_{v=1}^{M} \alpha_v\right)/M = \frac{s+(\sum_{w=s+1}^{v-1}\alpha_w)+\widehat{\alpha_G}(M-v-e+1)+e}{M}$$
$$=\alpha_G. \qquad (23)$$

If condition $\alpha_v < \widehat{\alpha_G}$ is not satisfied, α_v is no less than α_G, and then $(\sum_{v=1}^{M} \alpha_v)/M \geq \alpha_G$.

(2) We next prove $(\sum_{v=1}^{M} \alpha_v)/M \leq \alpha_G$ by contradiction. Suppose

$$\left(\sum_{v=1}^{M} \alpha_v\right)/M > \alpha_G. \qquad (24)$$

Since $s + e \leq \alpha_G M$, there must be a download interval F_k in intermediate stage I ($s < k < M - e$), such that the following inequality is satisfied before interval F_k:

$$\frac{s + e + \sum_{v=s+1}^{k-1} \alpha_v}{M} \leq \alpha_G,$$

and the following is satisfied after interval F_k:

$$\frac{s + e + \sum_{v=s+1}^{k} \alpha_v}{M} > \alpha_G. \tag{25}$$

However, the algorithm will never assign α_k such a value that makes the above inequality (25) holds: if so, we know $CurrentAverage(k+1) < 0$; in interval F_k, according to Line 13 of $WeightAdjustment()$, α_k would be made equal to $\widehat{\alpha_G}$, which gives $(\sum_{v=1}^{M} \alpha_v)/M = \alpha_G$.

Therefore, a contradiction occurs that the assumption in Eq. 24 does not hold, and we can derive

$$\Big(\sum_{v=1}^{M} \alpha_v\Big)/M \leq \alpha_G.$$

In summary, we have shown $\displaystyle\sum_{v=1}^{M} \alpha_v/M = \alpha_G$.

□

Algorithm 2 also handles peer dynamics, by having peer i update its lists periodically, since new neighbors are joining and previous neighbors may depart from the system. In our algorithm, $W(i)$, $R(i)$, and $P(i)$ are updated every τ_w, τ_r, and τ_p iterations of the algorithm, respectively. A peer can update its preference list $W(i)$ by requesting updated peer information from the tracker server, update its connection request receiving list $R(i)$ according to the proposal messages received from its neighbors, and modify its proposal list $P(i)$ according to the current connection status. In this way, peer i is always striving to discover better matches among the existing neighbors and chasing the Pareto optimality in its dynamic peer selection.

7 Performance evaluation

We now evaluate the performance of proposed algorithms. In Section 7.1, we present results of large-scale simulations under realistic settings, using a BitTorrent simulator that we develop. In Section 7.2, we implement a prototype BitTorrent system with our dynamic weight adjustment and peer selection algorithm, and evaluate it in an emulation environment.

7.1 Simulation results

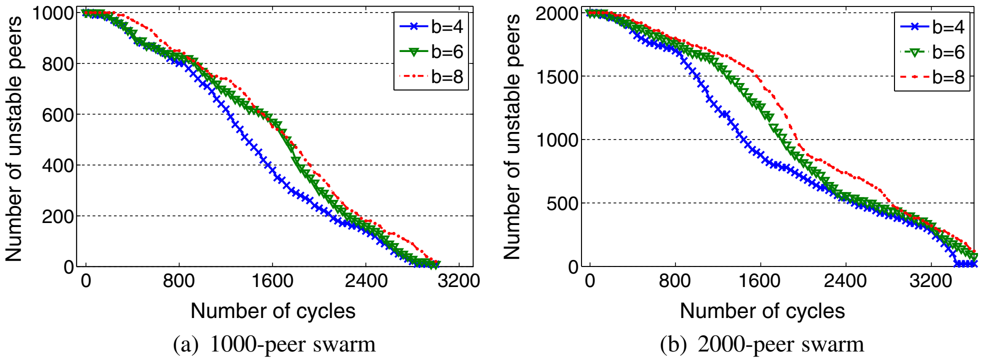
We implement a BitTorrent simulator according to the original BitTorrent protocol with a combination of C++ and Bram Cohen's Python code: the tracker part follows the BitTorrent code and the client part especially the functions related to peer selection are re-written by C++. We simulate a P2P swarm with up to 2,000 peers, which download a file of 128 MB. Parameter settings of our simulation experiments are based on practical data/distributions derived from real-world traces from field tests of P4P [12] using Pando clients in 2008, as provided by authors of P4P: Peers' upload capacities follow a heavy-tailed Pareto distribution in the major range of [256 Kbps, 10 Mbps] with the shape parameter of $k = 3$, corresponding to a mean upload capacity of 384 Kbps. The maximum rate that a peer i can download from a peer j with upload capacity u_j, i.e., r_{ji}, is decided by the upload bandwidth share u_j/b that j can provide. Each peer is assigned 15 existing peers in the swarm upon joining, and the inter-connection topologies among peers follow those summarized from the traces.

The peers in our swarm are uniformly randomly assigned to 10 ISPs. We assign a cost value to each pair of ISPs to represent their peering relationship, which are different numbers chosen from the range of [0, 700]; a larger number represents a higher traffic relay cost from one ISP to another, and the cost is 0 within the same ISP. The network cost incurred by downloading from peer j to peer i, c_{ij} in Algorithm 1, is set to be the cost value between their corresponding ISPs. Based on our settings, we observe that a preference cycle is fairly rare in the networks (lower than 0.1 % to the number of edges in graph) and we believe that ticking them out by breaking a random edge in each cycle does not affect the original properties of the whole network.

We make a common practical assumption throughout this section, that there is no malicious user that can change the P2P protocol by itself, because the protocol is based on negotiation between users and ISPs so anyone who did not follow the rules would be punished. Free-riders problem is also widely discussed in literatures because they affect resource allocation and peer collaborations in the overlay, we treat the free-riders as malicious users and do not consider their existence as it is not the main issue that we aim to solve in this implementation.

7.1.1 Without peer dynamics

We first investigate the convergence of our distributed iterative peer selection algorithm in Algorithm 1 (as is also the key component of Algorithm 2), in P2P swarms of different sizes with different numbers of download connections allowed at each peer (i.e., b). In our experiments, when a peer finishes its own file downloading, it remains and continues uploading to its matched peers until all finish downloading. Figure 2 shows the evolution of the number of unstable peers in the network (i.e., those who are still changing their peer selection), in P2P swarms with



(a) 1000-peer swarm

(b) 2000-peer swarm

Fig. 2 Convergence of P2P matching in swarms of different sizes

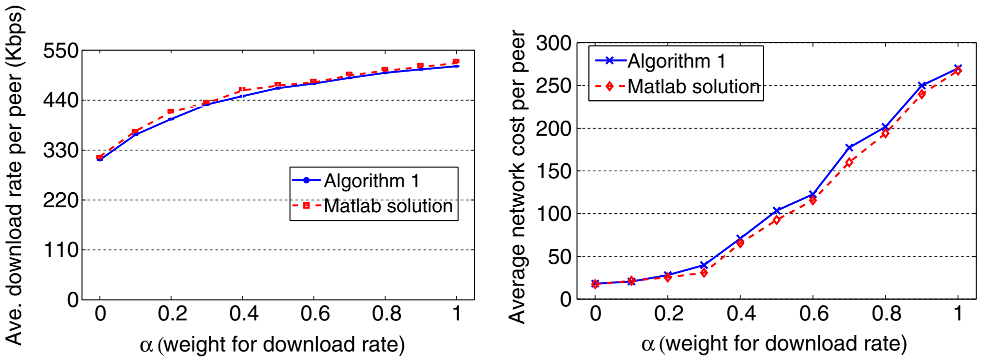
1,000 peers and 2,000 peers, respectively. In all cases, the number of unstable peers decreases quickly, i.e., peer selection in the network converges quickly to the stable b-matching. Under our experimental settings, considering that a peer needs about 40–50 min (which correspond to 24,000–30,000 cycles in our simulator) on average to download the entire file of 128 MB, such a convergence time about 5–6 min (corresponding to 3,000–3,500 cycles in our simulator) is minor, and the peers are already downloading using the matched peers while adjusting to the best peer selection. The maximal number of a peer's neighbor b is selected according to popular BitTorrent clients. The larger value of b may contribute to the unstability of P2P swarms at the beginning, but the differences are not significant in the comparison of convergence times. Noting that in these experiments as well as results and figures plotted in following sections, the data are collected from multiple runs of the experiments under same environment, and we calculate the average.

We then investigate the optimality of the stable b-matching (peer selection in the network), by comparing the download rates and network costs in the converged network to the optimal solutions of the global optimization problem in Eq. 6 derived using Matlab. We experiment in P2P swarms with 2,000 peers and $b = 6$, under different settings of the weight for download rate, α, and the weight for network cost, β ($\beta = 1 - \alpha$). Comparing Fig. 3a and b, we clearly observe the tradeoff between performance and locality in P2P swarms under realistic settings: the larger α is, i.e., the more weight a peer puts on download rate maximization in its peer selection, the higher the aggregate download rate per peer is in the resulting b-matching, at the cost of increased aggregate network cost at each peer simultaneously.

In both figures, the curve derived by Algorithm 1 and the curve showing Matlab solution largely overlap with each other, validating that the Pareto optimality of the resulting peer selection with Algorithm 1 is very close to the global optimality. The small gaps between the curves can be explained that the download rate (network cost) by Algorithm 1 is computed as the average per-peer download rate over its file downloading process, including the rates (costs) it obtains when the peer selection has not stabilized; on the other hand, the Matlab solutions shown represent the download rates (network costs) in stabilized matchings. The fact that the gaps are minor has further validated the insignificant influence of algorithm convergence time, as shown in Fig 2, in the overall download process at the peers.

7.1.2 With peer dynamics

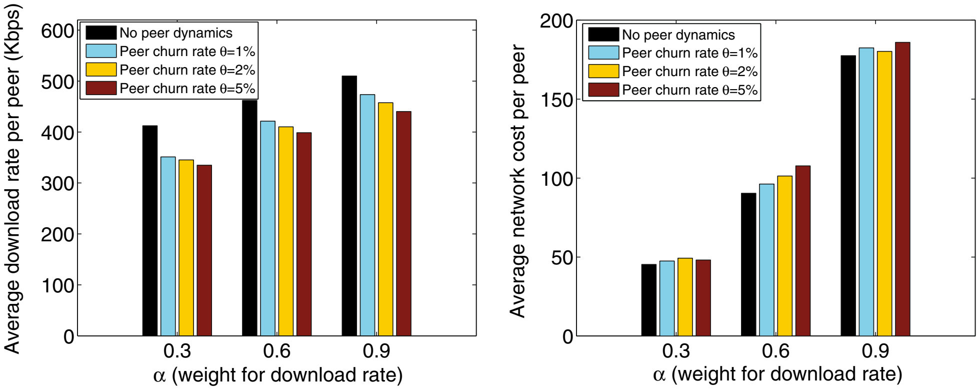
We next investigate Algorithm 2 in dynamic P2P networks with Poisson peer arrival and departure. We define the average *peer joining rate* (θ_j) and *peer departure rate* (θ_d) as



(a) Download rate

(b) Network cost

Fig. 3 Performance and locality comparison between optimal peer selections derived by Algorithm 1 and Matlab



(a) Download rate

(b) Inter-ISP traffic rate

Fig. 4 Performance and locality comparison at different dynamics levels

fractions in the initial number of peers in the network with details as follows:

$$\theta_j = \frac{\text{number of peers joining per minute}}{\text{initial number of peers}},$$

$$\theta_d = \frac{\text{number of peers departing per minute}}{\text{initial number of peers}}.$$

Average peer churn rate θ is defined as the sum of peer joining and departure rates, i.e., $\theta = \theta_j + \theta_d$.

The experiment settings are as follows. There are initially 2,000 peers in the network. The number of download connections allowed at each peer is $b = 5$. The receiving / proposal / preference lists at each peer are updated every $\tau_r = 3$, $\tau_p = 3$, and $\tau_w = 6$ iterations of the algorithm, respectively. While keeping peer joining and departure rates equal (i.e., $\theta_j = \theta_d$), we set peer churn rate θ to 1 %, 2 %, and 5 %, respectively. Each experiment runs for 70 min. We temporarily set fixed weights and do not execute the dynamic weight adjustment in Algorithm 2, but will evaluate its effectiveness in our emulation experiments in the next section.

We compare the average download rate and network cost per peer during peers' lifetime in the overlay under different churn rates. Figure 4 shows that the achieved download rates at the peers are slightly lower when the churn rate is larger, considering the large difference in the numbers of peers in the network at the end of 70 min, 1408 and 340, when the churn rates are 1 % and 5 %, respectively. On the other hand, the average network cost incurred by each peer remains similar at different dynamics levels.

If we only consider peers that finish downloading the entire file in the networks, the average download completion time under different churn rates is given in Fig. 5. We observe that the average download completion time is only slightly prolonged in cases of more significant peer dynamics.

All the above results show acceptable results when applying Algorithm 1 in dynamic networks comparing to static networks. Altough candidate neighbors change all the time, the measurements in Figs. 4 and 5 assure us that the performances do not degrade much in terms of average download rate maximization, network cost reduction and download complete time.

7.2 Emulation results

We have also implemented our dynamic weight adjustment and peer selection in Algorithm 2 in prototype BitTorrent clients and a tracker server, based on the source code of BitTorrent version 3.3 developed with python 2.6 [1]. We build a testbed consisting of 6 Intel Core2 Duo computers interconnected by a Gigabyte Ethernet switch: 3 of them are desktop computers with 2.8 GHz CPU and 4 GB RAM, and 3 are laptop computers with 2.4GHz CPU with 2 GB RAM. On each computer, we run 3–4 paravirtualized Xen guest operating systems on Linux (kernel 3.0), and 4 BitTorrent clients on every guest OS, which use different TCP ports for connections. We have also developed a tool to log system state and data in our experiments.

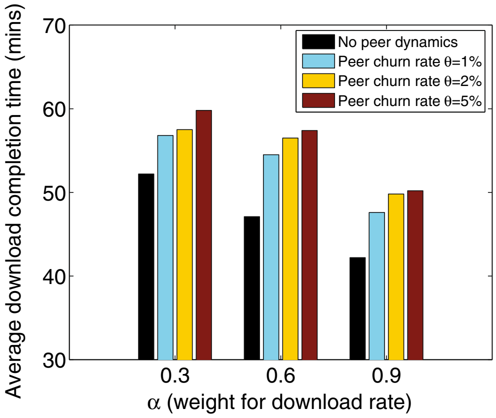
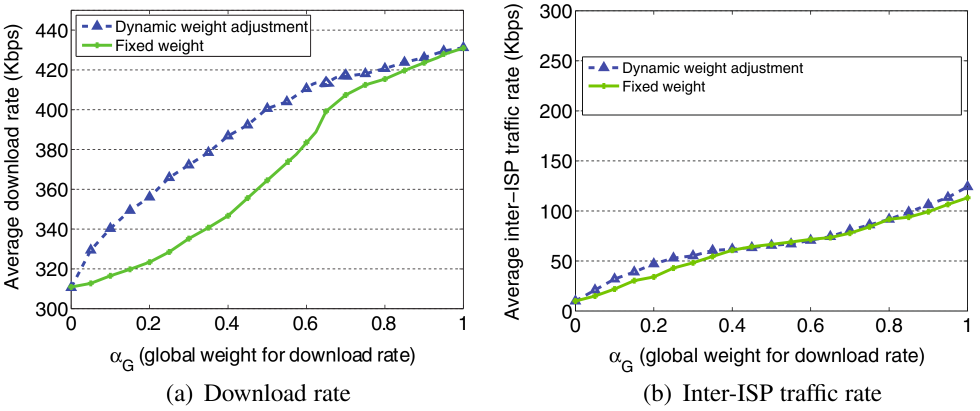


Fig. 5 A comparison of average download completion time in networks of different dynamics levels



(a) Download rate

(b) Inter-ISP traffic rate

Fig. 6 Performance and locality with dynamic weight adjustment and peer selection algorithm

To emulate real-world delay and loss rate, we use *NetEm* [28] to add practical delays to each packet a BitTorrent client sends, which is a utility in the *iproute2* package of tools in the up-to-date distribution of Linux (kernel 2.6). As latency and packet loss are important factors in P2P emulations [29], we set the network parameters according to trace studies of typical P2P overaly networks in our campus local network and logs extracted from network interface on machines running BitTorrent clients: a normal distribution with a mean of 6 ms and variance of 1.0 for packet delays, and the probability of 0.1 % emulated for packet losses.

In our following experiments on the emulation testbed, there are up to 80 clients distributed across 3 ISPs, to download a 128 MB file, and to be realistic, not all peers can complete download as they have probabilities to drop download tasks suddenly in the middle of process. Different upload capacities of the clients are emulated too, following those summarized from the traces and the same as the settings in Section 7. Peers arrive and depart following Poisson distribution with an average of 3 peers joining/departing every 4 min. All other settings are the same as those in the simulation experiments.

Figure 6 compares the average download rate and inter-ISP traffic incurred over a peer's lifetime, achieved with Algorithm 2 and with weights fixed to the preset global weights. We observe that our dynamic weight adjustment algorithm achieves very similar levels of inter-ISP traffic with those by fixed weights, but significantly higher download rates at the peers (and thus shorter file download completion times), especially when the preset weights prioritize inter-ISP traffic reduction more. The results are acceptable since inter-ISP traffic in dynamic weight strategy is controlled in the same level of fixed weight, which means ISPs would probably tolerate the shift from fixed to dynamic weight, meanwhile users gains benefits by improving download rates.

In the case of $\alpha_G = 0.5$, Fig. 7 verifies that peers can finish file download and start seeding faster with dynamic weight adjustment, as compared to the case of fixed weights in our controlled environment of emulation.

This clearly shows the effectiveness of our dynamic weight adjustment and peer selection algorithm in minimizing file download time in practical networks, while guaranteeing the preset inter-ISP traffic reduction goal (reflected by global weights α_G and β_G) is achieved over peers' download periods.

We next investigate the control messaging overhead incurred by Algorithm 2 in Fig. 8. The control overhead consists of messages that clients send to and receive from tracker server, and messages involved in peer selection

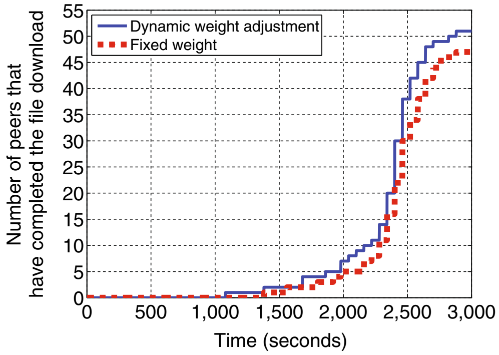


Fig. 7 Number of peers that have finished file download

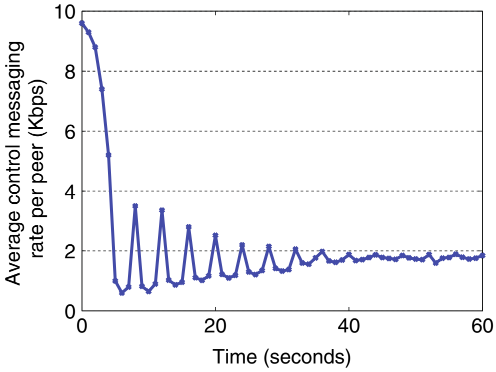


Fig. 8 Control messaging overhead in Algorithm 2

between clients. Specifically for a single peer in the network, the periodic update of a peer's proposal list $P(i)$, preference list $W(i)$, receiving list $R(i)$, with peer churn rate and connections a peer can be allowed to maintain, all have positive effects on communication overheads. Considering the downloading rate at each peer is about $350 - 450$ Kbps, the average overhead is minor. In addition, though the control messaging overhead is relatively high in system warm-up stage when many peers are concurrently looking for neighbors, it drops quickly to $1 - 2$ Kbps after initial sets of connections are established.

8 Concluding remarks

The main focus of this paper is to develop effective peer selection strategies that achieve a desired Pareto optimum in face of the tradeoff between download performance optimization and inter-ISP traffic minimization in a BitTorrent-like file sharing system.

From the theoretical point of view, we focus on the formal characterization of the tradeoff between performance and locality in a BitTorrent-like systems, and effective design of optimal peer selection strategies to achieve any desired tradeoff. Using multi-objective matching-based optimization, we effectively characterize the tradeoff, as well as design fully distributed optimization algorithms to carry out peer selection. Analytical proof verifies that our algorithm achieves global Pareto optimal peer selection, as represents a desired tradeoff between performance and locality in the network.

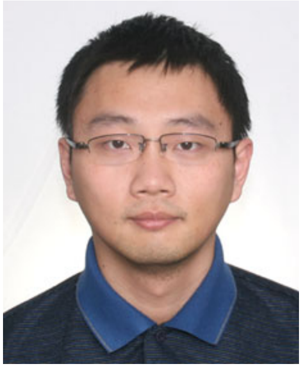
From the practical point of view, we extend our basic algorithm to a dynamic weight adjustment and peer selection algorithm, to be practically applied in networks with varying chunk availability and volatile peer dynamics. The dynamic algorithm allows peers to vary their weights towards performance and locality overtime, to meet specific demands at different stages of their download. Real-world traces from field tests of P4P project have been utilized to support experiments in both a simulator and an emulation testbed. Both the simulation and emulation results confirm the effectiveness of our algorithms.

Acknowledgments The authors would like to thank Prof. Yang Richard Yang in Yale university for his valuable suggestions in developing the work and his generous offering of field test traces from his P4P project.

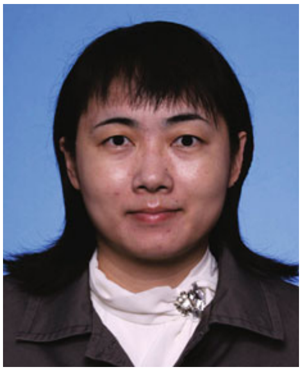
References

1. BitTorrent. http://www.bittorent.org
2. PPLive. http://www.pplive.tv
3. Skype. http://www.skype.com
4. Erman J, Mahanti A, Arlitt M, Williamson C (2007) Identifying and discriminating between web and peer-to-peer traffic in the network core. In: Proceedings of the sixteenth international world wide web conference (WWW'07), Banff, Alberta, Canada
5. Eckersley P, von Lohmann F, Schoen S (2007) Packet forgery By ISPs: a report on the Comcast Affair. Electronic Frontier Foundation, Tech. Rep. Available online: http://www.eff.org/files/eff_comcast_report.pdf
6. BitTorrent Developers Introduce Comcast Busting Encryption (2008). Available online: http://torrentfreak.com/bittorrent-devs-introduce-comcast-busting-encryption-080215/
7. Karagiannis T, Rodriguez P, Papagiannaki K (2005) Should internet service providers fear peer-assisted content distribution? In: Proceedings of the ACM/SIGCOMM internet measurement conference (IMC'05), Berkeley, CA, USA
8. Bindal R, Cao P, Chan W, Medved J, Suwala G, Bates T, Zhang A (2006) Improving traffic locality in bitTorrent via biased neighbor selection. In: Proceedings of the 26th international conference on distributed computing systems (ICDCS'06), Lisboa, Portugal
9. Gkantsidis C, Karagiannis T, Rodriguez P, Vojnovic M (2006) Planet scale software updates. In: Proceedings of ACM SIGCOMM, Pisa, Italy
10. Saleh O, Hefeeda M (2006) Modeling and caching of peer-to-peer traffic. In: Proceedings of 14th international conference on network protocols (ICNP'06), Santa Barbara, CA, USA
11. Shen G, Wang Y, Xiong Y, Zhao B, Zhang Z (2007) HPTP: relieving the tension between ISPs and P2P. In: Proceedings of the 6th international workshop on peer-to-peer systems (IPTPS'07), Bellevue, WA, USA
12. Xie H, Yang YR, Krishnamurthy A, Liu Y, Silberschatz A (2008) P4P: provider portal for applications. In: Proceedings of ACM SIGCOMM, Seattle, WA, USA
13. Aggarwal V, Feldmann A, Scheideler C (2007) Can ISPs and P2P users cooperate for improved performance? In: Proceedings of ACM SIGCOMM, Kyoto, Japan
14. Liu B, Cui Y, Lu Y, Xue Y (2009) Locality-awareness in bittorrent-like p2p applications. IEEE Trans Multimed 11(3):361–371
15. Le Blond S, Legout A, Dabbous W (2011) Pushing bittorrent locality to the limit. Comput Netw 55(3):541–557
16. Choffnes DR, Bustamante FE (2008) Taming the torrent: a practical approach to reducing cross-ISP traffic in peer-to-peer systems. In: In: Proceedings of ACM SIGCOMM, Seattle, WA, USA
17. Ren S, Tan E, Luo T, Chen S, Guo L, Zhang X (2010) TopBT: a topology-aware and infrastructure-independent bittorrent client. In: Proc. of INFOCOM'10, San Diego, CA, USA
18. Cuevas R, Laoutaris N, Yang X, Siganos G, Rodriguez P (2011) Deep diving into bittorrent locality. In: Proc. of INFOCOM'11
19. Huang C, Li J, Ross KW (2007) Can internet video-on-demand be profitable? In: Proceedings of ACM SIGCOMM, Kyoto, Japan
20. Wang J, Chiu DM, Lui J (2006) Modeling the peering and routing tussle between ISPs and P2P applications. In: Proceedings of the 14th international workshop on quality of service (IWQoS'06), New Haven, CT, USA
21. Mathieu F (2008) Self-stabilization in preference-based systems. Springer J P2P Netw Appl 1:104–121
22. Mathieu F, Postelnicu G, Reynier J (2009) The stable configuration in acyclic preference-based systems. In: Proceedings of the 28th IEEE conference on computer communications (INFOCOM'09), Rio de Janeiro, Brazil

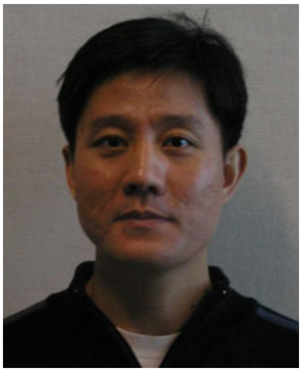
23. Cechlárová K, Fleiner T (2005) On a generalization of the stable roommates problem. ACM Trans Algor 1(1):143–156
24. Huang W, Wu C, Lau FC (2010) In: Proceedings of IEEE international conference on communications 2010 (IEEE ICC 2010). In: The performance and locality tradeoff in bittorrent-like P2P file-sharing systems
25. Vuze. http://www.vuze.com/
26. Collette Y, Siarry P (2004) Multiobjective optimization: principles and case studies. Springer-Verlag Berlin, Heidelberg, New York
27. Cohen B (2003) Incentives build robustness in bitTorrent. In: The first workshop on economics of peer-to-peer systems, Berkeley, CA, USA
28. Hemminger S (2005) Network emulation with NetEm. In: Proceedings of linux conference Australia (linux.conf.au’05), Canberra, Australia
29. Rao A, Legout A, Dabbous W (2010) Can realistic bittorrent experiments be performed on clusters? In: Proceedings of the 10th IEEE international conference on peer-to-peer computing (P2P’10), Delft, Netherlands



Wei Huang is a graduate student in the Department of Computer Engineering, University of Toronto. He got a computer science master’s degree in the University of Hong Kong in 2010 and a bachelor’s degree in Nanjing University, China in 2008. His research interest includes operating systems, computer and communication security and networks. He is a student member of IEEE and ACM.



Chuan Wu received her B.Eng. and M.Eng. degrees in 2000 and 2002 from Department of Computer Science and Technology, Tsinghua University, China, and her Ph.D. degree in 2008 from the Department of Electrical and Computer Engineering, University of Toronto, Canada. She is currently an assistant professor in the Department of Computer Science, the University of Hong Kong, Hong Kong. Her research interests include measurement, modeling, and optimization of large-scale peer-to-peer systems and online/mobile social networks. She is a member of IEEE and ACM.



Zongpeng Li received his B.E. degree in Computer Science and Technology from Tsinghua University (Beijing) in 1999, his M.S. degree in Computer Science from University of Toronto in 2001, and his Ph.D. degree in Electrical and Computer Engineering from University of Toronto in 2005. Since August 2005, he has been with the Department of Computer Science in the University of Calgary. In 2011–2012, Zongpeng was a visitor at the Institute of Network Coding, Chinese University of Hong Kong. His research interests are in computer networks, particularly in network optimization, multicast algorithm design, network game theory and network coding. Zongpeng was named an Edward S. Rogers Sr. Scholar in 2004, won the Alberta Ingenuity New Faculty Award in 2007, was nominated for the Alfred P. Sloan Research Fellow in 2007, and received the Best Paper Award at PAM 2008 and at HotPOST 2012.

Francis C.M. Lau received his PhD in computer science from the University of Waterloo. He is currently a professor in computer science in The University of Hong Kong. His research interests include computer systems, networks, programming languages, and application of computing in arts.