Diverse: Application-Layer Service Differentiation in Peer-to-Peer Communications

Chuan Wu, *Student Member, IEEE,* Baochun Li, *Senior Member, IEEE*
Department of Electrical and Computer Engineering
University of Toronto
{*chuanwu, bli*}*@eecg.toronto.edu*

***Abstract*— The peer-to-peer communication paradigm, when used to disseminate bulk content or to stream real-time multimedia, has enjoyed the distinct advantage of *scalability* when compared to the client-server model, since it takes advantage of available upload bandwidth at participating peers to alleviate server load. As multiple concurrent peer-to-peer sessions co-exist in the Internet, it is natural to demand differentiated services in different sessions, with respect to Quality of Service metrics such as bit rates and latencies. The problem of service differentiation across sessions, however, has never been addressed in the literature at the application layer. In this paper, we open a new direction of research that treats different peer-to-peer sessions with different priorities, and present *Diverse*, a novel application-layer approach to achieve service differentiation across different sessions. An extensive evaluation of our implementation of *Diverse* in an emulated peer-to-peer environment has demonstrated its effectiveness in achieving our design objectives.**

I. Introduction

The paradigm of peer-to-peer (P2P) communications over the Internet [1], [2], [3] has been successfully implemented in current-generation Internet applications. Well known example applications include large-volume peer-to-peer content distribution (*e.g.,* BitTorrent [4]), as well as on-demand or live peer-to-peer multimedia streaming [5], [6]. As one of the most significant benefits of peer-to-peer communications, participating peers in a content distribution or streaming session seek to maximally utilize their upload bandwidth capacities to serve other peers in the same session, alleviating the load on dedicated content distribution or streaming servers. Such an advantage is especially prominent when the number of participating peers in a peer-to-peer session scales up, as in a typical "flash crowd" scenario when a particular data item or media stream interests many peers at the same time.

As the peer-to-peer communication paradigm evolves in the Internet, we expect to experience *multiple* content distribution or media streaming sessions, running *concurrently* in the peer-to-peer application-layer overlay. Similar to the concept of *multicast groups* in traditional IP multicast, each *session* has its own group of participating peers, and its own data item to be disseminated to all the participants (or media to be streamed). It is natural to expect that some of these sessions expect a better service with respect to Quality of Service (QoS) metrics such as bit rates and latencies. As good examples, a live peer-to-peer streaming session of premium television channels to paid subscribers should enjoy a higher priority and a better quality than another streaming session of regular broadcast television channels to the general public. Furthermore, a streaming session of live multimedia should be handled with a higher priority than a content distribution session of bulk data (*e.g.,* on-line backups). In this work, such a need for differentiated services to different peer-to-peer communication sessions is referred to as *service differentiation* in application-layer peer-to-peer networks. Although many mechanisms have been proposed to support service differentiation in the network layer [7], [8], [9], very few have been implemented in core and edge switches in the current Internet, which are still largely best-effort.

In this paper, we propose *Diverse*, a new paradigm of peer-to-peer communication with support for different levels of service differentiation. In *Diverse*, we treat different sessions with different priorities, and propose a novel application-layer approach to achieve service differentiation across peer-to-peer communication sessions. Towards this objective, our original contributions are two-fold. *First*, we seek to construct and customize optimal peer-to-peer topologies for each of the concurrent sessions in the peer-to-peer overlay, based on their priority levels. *Second*, based on these optimal topologies, we apply priority-based message scheduling at each of the participating peers in the application layer, which further improves the quality experienced by high-priority sessions over those of lower priorities. Highlights of *Diverse* include the design of a practical protocol to implement service differentiation in peer-to-peer networks, which adapts to network dynamics including peer joins, departures and network congestion. To our best knowledge, this paper is the first that identifies the need for service differentiation across peer-to-peer communication sessions, and that addresses the corresponding challenges with pure application-layer approaches.

The remainder of this paper is organized as follows. In Sec. II, we present our system model and motivate the design of *Diverse*. A detailed description of the main components in *Diverse* are discussed in Sec. III. In Sec. IV, we present practical and fully distributed protocols that implement *Diverse*. Evaluation results of its implementation in an emulated peer-to-peer environment are presented in Sec. V. We discuss related work and conclude the paper in Sec. VI and Sec. VII, respectively.

II. DIVERSE: SYSTEM MODEL AND MOTIVATIONAL INSIGHTS

In this paper, we consider a collection of peers, with each peer participating in one or multiple *peer-to-peer communication sessions*. In each peer-to-peer communication session, there is one original data *source*, and the rest of the peers are *receivers*. The connectivity graph among the source and receivers in one session constitutes a mesh overlay topology. The entire *peer-to-peer overlay* consists of all such mesh topologies, each in an peer-to-peer communication session.

Such a peer-to-peer overlay can be modeled as a directed graph $G = (N, A)$, where N is the set of all the vertices (peers) and A is the set of directed arcs (directed peer-to-peer links). Let S be the set of sessions that concurrently exist in the peer-to-peer overlay. Let $v_s^{(0)}$ be the source for session s, $\forall s \in S$, and $T_s = \{v_s^{(1)}, v_s^{(2)}, v_s^{(3)}, \ldots\}$ be its set of receivers. A_s is the set of peer-to-peer links in session s. When peer i is the *upstream* peer serving peer j in session s, *i.e.*, peer j is the *downstream* peer of i in s, the link (i, j) is in A_s. We then have $N = \cup_{s \in S}(\{v_s^{(0)}\} \cup T_s)$ and $A = \cup_{s \in S} A_s$. Each session s is assigned to a certain priority level, denoted by C_s, with a larger number denoting a higher priority level. An example of such a peer-to-peer overlay with two sessions is illustrated in Fig. 1.

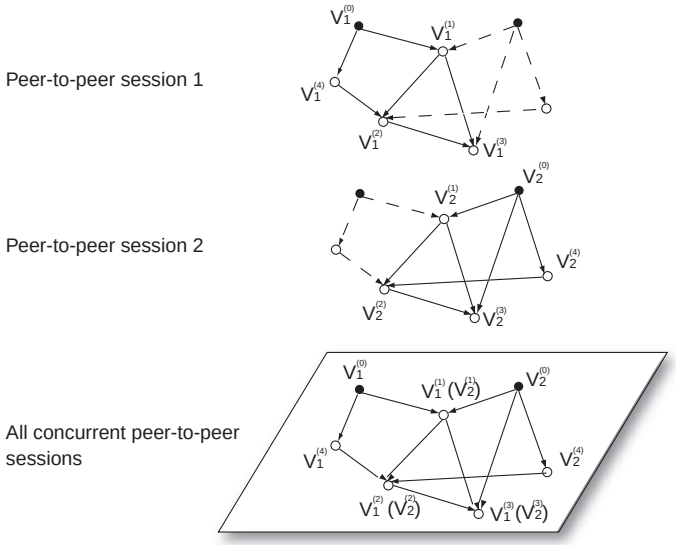

Fig. 1. Concurrent peer-to-peer communication sessions: an example.

In the design of *Diverse* in this paper, we make the following realistic assumptions:

- ▷ At different times in a communication session, a peer may switch to different upstream peers to retrieve its available data content at the moment. Therefore, the mesh topologies of the sessions are changing dynamically.
- ▷ In such a mesh network featuring parallel downloading, there are risks the same content may be unnecessarily supplied by multiple upstream peers. In this paper, we assume such delivery redundancy problem is solved by applying a certain coding scheme, *e.g.*, network coding, and there is no need to reconcile the content difference during retrieval.
- ▷ In a peer-to-peer overlay, performance bottlenecks mainly occur at the peers at the edge of the Internet, rather than at Internet backbone routers. This is due to the fact that peers usually have very limited and heterogeneous processing and bandwidth capacities.

Based on these assumptions, the design objective of an effective service differentiation mechanism is to provide better performance to high-priority sessions and a certain degree of fairness to low-priority sessions, while still adapting well to the dynamics in the network. In addition, we wish to make the best use of the limited upload capacities at each of the peers in the application layer.

We now motivate the design of *Diverse* by identifying a few influential parameters that determine the main performance metrics of a peer-to-peer communication session: *end-to-end delay* and *throughput* experienced by each peer in the session.

A careful inspection of the end-to-end delay in the peer-to-peer session leads to the following parameters: (1) The message queueing delay at each intermediate peer; (2) the bandwidth share allocated by each peer for the session, which determines its transmission delay; (3) the number of overlay hops traversed by data flows in the session from the source; and (4) the delay on each overlay link between peers (determined by the sum of delays in the underlying IP-layer links).

In *Diverse*, to reduce the message queueing delay at each peer, we apply an application-layer priority scheduling algorithm, henceforth referred to as the *overlay priority scheduling* algorithm. Such scheduling of messages belonging to different sessions allows for differentiated queueing delays at the peers, with respect to sessions at different priority levels.

To address the challenges of differentiating other delay parameters and throughput across different sessions, we propose a priority-based optimal topology construction and bandwidth allocation algorithm, referred to as *optimal bandwidth allocation* algorithm. In this algorithm, by choosing better overlay paths for high-priority sessions, we take into consideration the number of overlay hops the paths traverse, as well as the quality of overlay links. By allocating upload bandwidths based on the priority levels, we further guarantee the bandwidth share for high-priority sessions. *Diverse* represents the design of efficient algorithms that implement these key insights.

III. DIVERSE: EFFECTIVE SERVICE DIFFERENTIATION

Diverse represents a complete solution towards the realization of service differentiation across sessions of different priorities. It consists of two main components: *overlay priority scheduling* and *priority-based optimal bandwidth allocation.*

A. Overlay priority scheduling

When data messages belonging to different sessions arrive at a peer, the scheduling of the order in which they are processed and relayed to other peers plays a significant role in differentiating their queueing delays. Therefore, we design two priority schedulers at each peer, for incoming and outgoing messages, respectively. An example of the two priority schedulers is shown in Fig. 2, in which Peer i is participating in 4 sessions,

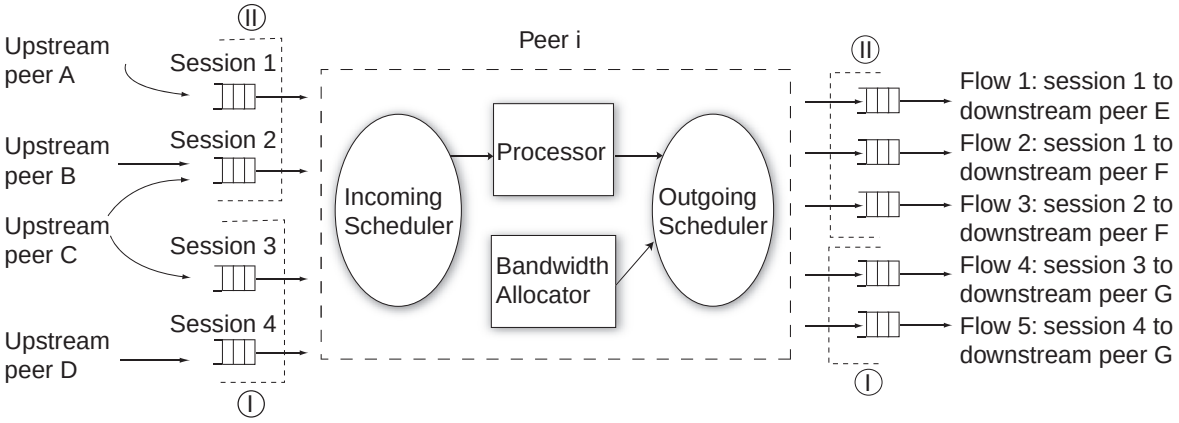

Fig. 2. Overlay scheduling with two priority schedulers: an example.

with session 1 and 2 in the high priority level *II*, session 3 and 4 in the low priority level *I*.

At the incoming scheduler of each peer, an incoming message queue is established for each of the sessions the peer is participating in, and these queues are organized based on their priority levels. Messages in different queues are processed starting from the highest priority level. If there are C priority levels in total, the scheduler serves the messages from the queues at priority level c only if there does not exist any message in the queues at priority levels $c+1$, $c+2$, ..., C. For queues in the same priority level, a round robin approach is applied to schedule messages from different queues to the processor. In the example shown in Fig. 2, the incoming queues of session 1 and 2 are processed before those of session 3 and 4, as long as they are not empty.

At the outgoing scheduler, a message queue is set up for each of the outgoing message flows of a session, which is destined to a downstream peer. These queues are scheduled to send data messages in the order of their session's priorities, and at the optimal rates computed by the priority-based bandwidth allocator, which implements the optimal bandwidth allocation algorithm to be discussed in the following section. The regulation of allocated bandwidths for the message flows are implemented by keeping track of the *time to send the next message* for each of the queues. The *time to send the next message* for an outgoing queue is initialized to zero, and increased by the time used to send one message, *i.e.*, the message size divided by the allocated bandwidth, whenever a message of this queue is sent. Therefore, the outgoing scheduler goes through the outgoing queues in the order of their priorities, and sends messages from a queue as long as the current time is later than the *time to send the next message* of this queue. As an analytical example based on Fig. 2, if data messages are all of the same size and the bandwidths allocated to five outgoing flows are 2, 4, 3, 2, 1 respectively, the outgoing scheduler sends (on average) 2, 4, 3, 2, 1 messages in one round for queues 1 to 5, respectively.

With the two schedulers, data messages belonging to high-priority sessions are guaranteed a lower queueing delay at the peers. However, the risk of starving low-priority sessions may arise with such priority scheduling. To eliminate such a risk, in our design, we also optimally allocate upload capacity of a peer to all the sessions based on their priority levels.

In *Diverse*, the implementation of priority scheduling at peers in the application layer effectively compensates the lack of such support at underlying IP routers. As we assume performance bottlenecks of a peer-to-peer overlay lie in the peers rather than the routers, we believe such application-layer priority scheduling can actually achieve better service differentiation across multiple peer-to-peer sessions.

B. Priority-based optimal bandwidth allocation

We next present the derivation of our optimal bandwidth allocation algorithm, which constructs priority-based optimal topologies and computes bandwidth share along the links for all the sessions in the peer-to-peer overlay. Towards this objective, we first set up a convex optimization model that maximizes overall utilities received by the peers in all the sessions. Better path selection and favorable bandwidth allocation for high-priority sessions are achieved by effectively formulating session priorities and the quality of overlay links into utility functions. We then discuss its solution algorithm based on Lagrangian relaxation technique and subgradient algorithm.

1) Problem formulation: We use $U_s^{(ij)}$ to represent the utility that peer j can receive by retrieving content of session s from upstream peer i. It is a non-decreasing function of the variable $x_s^{(ij)}$, the bandwidth allocated for session s along link (i,j). We further assume $U_s^{(ij)}(x_s^{(ij)})$ is strictly concave and twice differentiable [10]. Based on our bottleneck assumption, we consider constraints of the heterogeneous upload and download capacities of each peer i, denoted by O_i and I_i respectively. Let R_s be the maximum rate of session s. We assume that a receiver only participates in sessions whose total rate can be accommodated by its download capacity, *i.e.*, $\sum_{s:i\in T_s} R_s \leq I_i$. In this case, we include the session rate constraints at each receiver in the convex program, but omit the download capacity constraint, which will otherwise be redundant. The convex program is formulated as follows:

P:

$$\max \sum_{s\in S} \sum_{(i,j)\in A_s} U_s^{(ij)}(x_s^{(ij)}) \tag{1}$$

subject to

$$\begin{aligned} \textstyle\sum_{s\in S}\sum_{j:(i,j)\in A_s} x_s^{(ij)} \le O_i, \quad & \forall i \in N, \\ \textstyle\sum_{i:(i,j)\in A_s} x_s^{(ij)} \le R_s, \quad & \forall j \in T_s, \forall s \in S, \\ x_s^{(ij)} \ge 0, \quad & \forall (i,j) \in A_s, \forall s \in S. \end{aligned} \tag{2}$$

An optimal solution to convex program **P**, $\{x_s^{(ij)*}, \forall (i,j) \in A_s, \forall s \in S\}$, provides an optimal bandwidth allocation strategy, and determines the optimal delivery paths for all the sessions in the network. In order to select better paths and guarantee the rate for high-priority sessions, we determine the utility function $U_s^{(ij)}$, $\forall (i,j) \in A_s$, $\forall s \in S$, as follows: its value is larger if session s is in a higher priority level, *and* if the quality of link (i,j) is better. The quality of link (i,j) can be determined by the delay or loss rate on the link, or the stability of upstream peer i. We will mainly use *delay* to evaluate the link quality in our protocol. With C_s being the priority of session s and $Q^{(ij)}$ denoting the quality of link (i,j), we formulate the utility function as

$$U_s^{(ij)} = C_s \log(1 + Q^{(ij)} x_s^{(ij)}).$$

By maximizing utility functions in this form, the convex optimization guarantees delivery rates of higher priority sessions rather than those of lower priority sessions, and assigns larger bandwidths for the former on the links with better quality. In addition, fairness of bandwidth allocation to all sessions is also guaranteed with this strictly-concave logarithmic function [11]. We also note that the input session topologies to the optimization problem are dynamically determined by the content availability at the peers. Therefore, the upstream peers of a receiver in the input topologies are those which already have the contents it attempts to retrieve. This implicitly takes overlay hop counts into consideration in the optimization, and thus the receivers are guaranteed to be in close proximity to the source in terms of overlay hop counts in the resulting optimal topologies.

2) Subgradient solution: In order to derive a fully decentralized algorithm to solve convex program **P**, we utilize the subgradient algorithm based on Lagrangian relaxation technique, which is an efficient solution technique for convex programs and can naturally achieve distributed implementation [12], [13].

We start by analyzing the decomposition of convex program **P** into multiple subproblems which can be independently solved by each peer with its local information. For this purpose, we utilize Lagrangian relaxation technique and relax the constraints in (2) to derive Lagrangian dual of the primal problem **P**. To apply Lagrangian relaxation, we consider the equivalent standard minimization form of the objective function in (1):

$$\min -\sum_{s\in S}\sum_{(i,j)\in A_s} U_s^{(ij)}(x_s^{(ij)}). \tag{3}$$

Associating Lagrangian multipliers μ_s^j, $\forall j \in T_s$, $\forall s \in S$, with the constraints in (2), we modify the objective function in (3):

$$\begin{aligned} & -\sum_{s\in S}\sum_{(i,j)\in A_s} U_s^{(ij)}(x_s^{(ij)}) + \sum_{s\in S}\sum_{j\in T_s} \mu_s^j \Big(\sum_{i:(i,j)\in A_s} x_s^{(ij)} - R_s \Big) \\ &= -\sum_{i\in N}\sum_{s\in S}\sum_{j:(i,j)\in A_s} (U_s^{(ij)}(x_s^{(ij)}) - \mu_s^j x_s^{(ij)}) - \sum_{s\in S}\sum_{j\in T_s} \mu_s^j R_s. \end{aligned}$$

Then we obtain the Lagrangian dual as follows:

$$\max_{\mu \ge 0} L(\mu) \tag{4}$$

where

$$\begin{aligned} & L(\mu) \\ = \; & \min_P \Big(-\sum_{i\in N}\sum_{s\in S}\sum_{j:(i,j)\in A_s} (U_s^{(ij)}(x_s^{(ij)}) - \mu_s^j x_s^{(ij)}) \\ & - \sum_{s\in S}\sum_{j\in T_s} \mu_s^j R_s\Big) \\ = \; & -\Big(\max_P \sum_{i\in N}\sum_{s\in S}\sum_{j:(i,j)\in A_s} (U_s^{(ij)}(x_s^{(ij)}) - \mu_s^j x_s^{(ij)})\Big) \\ & - \sum_{s\in S}\sum_{j\in T_s} \mu_s^j R_s, \end{aligned} \tag{5}$$

and the polytope P is defined by the following constraints:

$$\begin{aligned} \textstyle\sum_{s\in S}\sum_{j:(i,j)\in A_s} x_s^{(ij)} \le O_i, \quad & \forall i \in N, \\ x_s^{(ij)} \ge 0, \quad & \forall s \in S, \forall (i,j) \in A_s. \end{aligned}$$

Here, the Lagrangian multiplier μ_s^j can be understood as the price provided by receiver j to its upstream peers for session s. Such an interpretation will become clear as we come to the adjustment of μ_s^j with the subgradient algorithm.

We observe that the Lagrangian subproblem in (5) can be decomposed into multiple sub convex optimization problems, each to be independently solvable by a peer i, $\forall i \in N$:

$$\max \sum_{s\in S}\sum_{j:(i,j)\in A_s} (U_s^{(ij)}(x_s^{(ij)}) - \mu_s^j x_s^{(ij)}) \tag{6}$$

subject to

$$\textstyle\sum_{s\in S}\sum_{j:(i,j)\in A_s} x_s^{(ij)} \le O_i, \tag{7}$$

$$x_s^{(ij)} \ge 0, \forall s \in S, \forall j : (i,j) \in A_s. \tag{8}$$

Based on this nice decomposable structure of the Lagrangian subproblem in (5), we are able to solve the Lagrangian dual in (4) with subgradient algorithm in a distributed fashion. In what follows, we discuss the subgradient algorithm, which solves the Lagrangian dual and also derives the optimal solution to the primal problem **P**.

We start with a set of initial non-negative Lagrangian multipliers $\mu_s^j[0]$, $\forall j \in T_s$, $\forall s \in S$. During each iteration k of the subgradient algorithm, $k = 1, 2, \ldots$, given the current Lagrangian multiplier values $\mu_s^j[k]$, we solve the $|N|$ subproblems in (6) by an approach similar to water filling (pp. 245, [14]), which goes as follows:

The water-filling approach. Let $f(x)$ be the objective function in (6), *i.e.*,

$$f(x) = \sum_{s\in S}\sum_{j:(i,j)\in A_s} (U_s^{(ij)}(x_s^{(ij)}) - \mu_s^j x_s^{(ij)}).$$

Then the marginal utility for $x_s^{(ij)}$ is

$$\frac{df(x)}{dx_s^{(ij)}} = U_s^{'(ij)}(x_s^{(ij)}) - \mu_s^j.$$

Beginning with $x_s^{(ij)} = 0, \forall j : (i,j) \in A_s, \forall s \in S$, we find one $x_s^{(ij)}$ with the largest positive marginal utility and increase this $x_s^{(ij)}$. As $U_s^{(ij)}(x_s^{(ij)})$ is strictly concave, $U_s^{'(ij)}(x_s^{(ij)})$ decreases with the increase of $x_s^{(ij)}$. We increase this $x_s^{(ij)}$ until its marginal utility is no longer the largest. Then we find a new $x_s^{(ij)}$ with the currently largest marginal utility and increase it. This process repeats until the sum of all $x_s^{(ij)}$'s, $\forall j : (i,j) \in A_s$, $\forall s \in S$, reaches O_i or all the marginal utilities become zero. □

This water-filling approach can be intuitively understood as follows: we always allocate the upload capacity of peer i, *i.e.*, O_i, to such a bandwidth variable $x_s^{(ij)}$, that the objective function $f(x)$ achieves the largest gains. We further prove the correctness of the approach in the following theorem:

Theorem 1. *The above water-filling approach obtains an optimal solution for the convex optimization problem in (6).*

We postpone detailed proof of theorem 1 to Appendix I.

With the optimal $x_s^{(ij)}[k]$'s computed by the water-filling method, we then update the Lagrangian multipliers by

$$\mu_s^j[k+1] = \max(0, \mu_s^j[k] + \theta[k](\sum_{i:(i,j)\in A_s} x_s^{(ij)}[k] - R_s)), \quad \forall j \in T_s, \forall s \in S, \quad (9)$$

where θ is a sequence of step sizes that guarantees the convergence of the subgradient algorithm, if it satisfies the following conditions (pp. 26, [12]):

$$\theta[k] > 0, \lim_{k\to\infty}\theta[k] = 0, \text{ and } \sum_{k=1}^{\infty}\theta[k] = \infty. \quad (10)$$

In our algorithm, we choose the step length sequence $\theta[k] = a/(b+ck), \forall k, a > 0, b \geq 0, c > 0$, which satisfies the above conditions.

Eq. (9) shows the adjustment of prices at each receiver j for all the sessions it participates in, based on the allocated upload bandwidths from its upstream peers. If the total acquired bandwidth for session s exceeds the session rate, constraint (2) is violated, and thus price μ_s^j is raised; if the total bandwidth fails to reach the session rate, the price is reduced.

The subgradient algorithm to solve Lagrangian dual in (4) is summarized in Table I.

TABLE I

OPTIMAL BANDWIDTH ALLOCATION ALGORITHM

1. Choose initial Lagrangian multiplier values $\mu_s^j[0]$, at each receiver j in session s, $\forall s \in S$.

2. Repeat the following iteration until convergence:
at times $k = 1, 2, \ldots$

1) Solve the convex program in (6) with the water-filling approach at each peer i in N, and derive $x_s^{ij}[k]$, $\forall j : (i,j) \in A_s$, $\forall s \in S$;

2) Update Lagrangian multiplier $\mu_s^j[k+1] = \max(0, \mu_s^j[k] + \theta[k](\sum_{i:(i,j)\in A_s} x_s^{(ij)}[k] - R_s))$, where $\theta[k] = a/(b+ck)$, at each receiver j in session s, $\forall s \in S$.

This subgradient algorithm not only converges to the optimal Lagrangian multiplier values for the Lagrangian dual, but also derives the optimal solution to the primal problem **P**, which is formally stated in the following theorem.

Theorem 2. *With the subgradient algorithm described in Table I, starting from any nonnegative multiplier vector μ, where $\mu = (\mu_s^j, \forall j \in T_s, \forall s \in S)$, the sequence of vectors $\{\mu[k]\}$ converges to its optimum μ^*, and the sequence of vectors $\{x[k]\}$, where $x = (x_s^{(ij)}, \forall (i,j) \in A_s, \forall s \in S)$, converges to the unique optimal solution x^* of the primal problem **P**.*

We again postpone the proof to Appendix II.

3) An illustrative example: We now give a simple illustrative example to demonstrate the convergence of the algorithm to priority-based optimal topologies. Fig. 3 shows a peer-to-peer overlay network with two sessions from two sources to four common receivers, with session rates $R_1 = R_2 = 500$ Kbps, and priorities $C_1 = 2$ and $C_2 = 1$. O_i for the six peers, $v_1^{(0)}$, $v_2^{(0)}$, $v^{(1)}$, $v^{(2)}$, $v^{(3)}$ and $v^{(4)}$, are 1, 1, 0.5, 0.9, 0.3 and 0.5 (in Mbps), respectively. Numbers labeled on the links of Fig. 3 (A) indicate link delays. With the subgradient algorithm, the price vector μ converges within 50 iterations, as shown in Fig. 3 (B). The resulting optimal topologies for both sessions are depicted in Fig. 3 (C) and (D) respectively, with the allocated bandwidths labeled on the links (in Mbps). Evidently, high-priority session 1 uses better links with allocate rates for all the peers up to the maximum session rate, while session 2 picks up the remaining available bandwidth, and its maximum rate is not achieved by all the participants.

IV. DIVERSE: PRACTICAL DISTRIBUTED PROTOCOLS

In this section, we design practical protocols to implement *Diverse*'s optimal bandwidth allocation algorithm and overlay priority scheduling, as well as to handle various dynamics in realistic peer-to-peer networks.

A. Optimal bandwidth allocation protocol

The subgradient algorithm in Table I is directly implementable with a fully decentralized protocol. In the protocol, at each peer i, as a downstream peer, it maintains its prices μ_s^i for all the sessions it participates in; as an upstream peer, it is responsible for allocating upload bandwidth to its downstream peers, by solving the sub optimization problem in (6).

Let S_i be the set of sessions peer i is in. In each iteration, peer i sends its prices $\mu_s^i, \forall s \in S_i$ to its upstream peers in the corresponding sessions. Meanwhile, after it receives the current prices μ_s^j from all its downstream peers for all the sessions in S_i, it allocates its upload capacity with the water-filling approach. Then it communicates the computed optimal bandwidths, $x_s^{(ij)}, \forall j : (i,j) \in A_s, \forall s \in S_i$, to the corresponding downstream peers. As its upstream peers are allocating their own upload capacities as well, peer i receives its allocated share, $x_s^{(ui)}, \forall u : (u,i) \in A_s, \forall s \in S_i$, at the same time. When it has collected all the allocated bandwidths for session s, it adjusts its price for the session, μ_s^i, based on Eq. (9), and again communicates the updated prices to the upstream peers.

We implement two types of messages in the protocol: Price Update (*PU*) messages, carrying updated price from a downstream peer; Bandwidth Update (*BU*) messages, containing

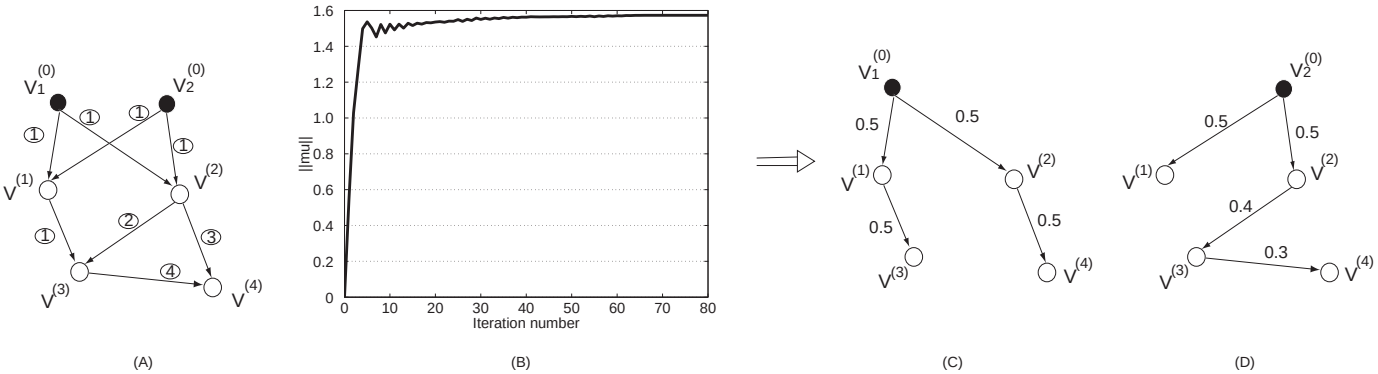

Fig. 3. Priority-based optimal bandwidth allocation: an example with two sessions.

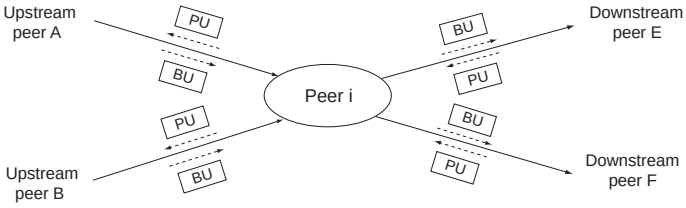

Fig. 4. Messaging model for the optimal bandwidth allocation protocol.

allocated bandwidths from an upstream peer. An illustrative example for the messaging in each iteration of the subgradient algorithm is given in Fig. 4, and details of the distributed optimal bandwidth allocation protocol are summarized in Table II.

B. Overlay priority scheduling protocols

The pseudocode for the priority scheduling protocols to be implemented at the incoming and outgoing schedulers of a peer is given in Table III.

C. Handling network dynamics

A peer-to-peer communication network is inherently dynamic: peers may join and leave at will, delay may fluctuate and congestion may occur along an overlay link, etc. Such dynamics pose a significant challenge to our protocol implementation, especially for the priority-based optimal bandwidth allocation protocol. In *Diverse*, we consider appropriate handling of network dynamics as one of our main design objectives.

In our implementation, time is divided into *slots*, which are the time intervals for updating local measurements and topology information, and for executing the optimal bandwidth allocation protocol with updated information.

In every time slot, each peer actively collects or updates local information for its optimal upload bandwidth allocation, *e.g.*, pinging neighbor peers to measure the overlay link delays. With the updated information, at the beginning of a time slot, a peer initiates the protocol execution by sending out its new prices if it has detected any changes in the previous time slot, *e.g.*, join/departure of a neighbor peer, delay variation on the adjacent overlay links, etc. A peer who has not experienced any change in its local topology will also participate in the protocol execution, when it receives any price updates or allocated bandwidth updates from its neighbor peers. In this way, local changes will propagate throughout the network, and all the affected peers cooperate in a new round of optimal bandwidth allocation.

Note that in this scenario, the optimization protocol always runs from the previous optimal values, thus expediting its convergence to the new optimum. Only slight modifications to the protocol in Table II are required for it to run in such a dynamic environment. Instead of initializing μ_s^j's to zero, we start the optimization protocol with μ_s^j's being the optimal prices obtained in the previous time slot.

In addition to executing the optimal bandwidth allocation protocol once every time slot, each peer also promptly adapts to local dynamics inside each time slot, in order to provide consistent performance guarantee for high-priority sessions. We divide our discussion into three cases.

1) Peer joins: In a time slot, when a peer joins session s, it is bootstrapped with U initial upstream peers. Then it requests bandwidth $\frac{R_s}{U}$ from each of these upstream peers. Upon receiving such a request from a new peer, an upstream peer first assigns its spare upload bandwidth to it. If that is not sufficient, it further shares the bandwidth allocated to sessions whose priorities are lower than s, in proportion to their priority values.

2) Peer departures: When a peer detects the departure or failure of an upstream peer from session s, it tries to obtain more upload bandwidth from its remaining upstream peers to compensate its rate loss. When an upstream peer receives such a request for more bandwidth, its handling is similar to the peer joining case. The upstream peer first assigns its spare upload bandwidth to the requesting peer, and then proportionally deprives bandwidth allocated to lower priority sessions.

3) Network congestion: In our system model, we assume capacity bottlenecks occur at the peers, rather than at routers underlying overlay links. Therefore, while available overlay link bandwidths may fluctuate due to cross traffic variation at the routers, it still does not constitute the bottleneck in most cases, and thus does not affect optimal upload bandwidth allocation. Nevertheless, in *Diverse*, we do consider handling of severe congestion that may occur along the overlay links in some time slots.

In this case, delay on a congested overlay link becomes much larger than previously detected delays on the same link.

TABLE II

OPTIMAL BANDWIDTH ALLOCATION PROTOCOL EXECUTED AT PEER i

Notations:

S_i: the set of sessions peer i participates in.
M_i: the set of prices peer i offers for all sessions in S_i.
P_i: the set of received prices from downstream peers.
UB$_i$: the set of received allocated bandwidths from upstream peers.
DB$_i$: the set of allocated bandwidths to downstream peers.

Initialization:

$P_i \leftarrow \phi$
UB$_i \leftarrow \phi$
DB$_i \leftarrow \phi$
$M_i \leftarrow \{\mu_s^i | \mu_s^i = 0, \forall s \in S_i\}$
Send *PU* messages containing initial μ_s^i's to all the upstream peers, $\forall u, (u,i) \in A_s, \forall s \in S_i$

Upon receiving a *PU* message:

Retrieve price μ_s^j from the message
$P_i \leftarrow P_i \cup \{\mu_s^j\}$
if P_i contains all the prices $\mu_s^j, \forall s \in S_i, \forall j : (i,j) \in A_s$
    Compute $x_s^{(ij)}$, $\forall s \in S_i, \forall j : (i,j) \in A_s$, by solving (6) with the water-filling method
    Update $x_s^{(ij)}$ in *DB*$_i$, $\forall s \in S_i, \forall j : (i,j) \in A_s$
    for each downstream peer $j, (i,j) \in A_s, \forall s \in S_i$
        Send j a *BU* message containing $x_s^{(ij)}, \forall s \in S_i$
    end for
    $P_i \leftarrow \phi$
end if

Upon receiving a *BU* message:

Retrieve allocated bandwidth $x_s^{(ui)}$ from the message
UB$_i \leftarrow$ *UB*$_i \cup \{x_s^{(ui)}\}$
for each session s, $\forall s \in S_i$
    if *UB*$_i$ contains all the allocated bandwidths for session s, *i.e.*, $x_s^{(ui)}, \forall u : (u,i) \in A_s$
        $\mu_s^i \leftarrow \max(0, \mu_s^i + \theta(\sum_{u:(u,i)\in A_s} x_s^{(ui)} - R_s))$
        Update μ_s^i in M_i
        for each upstream peer u in session s, $(u,i) \in A_s$
            Send u a *PU* message containing μ_s^i
            UB$_i \leftarrow$ *UB*$_i - \{x_s^{(ui)}\}$
        end for
    end if
end for

TABLE III

PRIORITY SCHEDULING PROTOCOLS AT PEER i

Notations:

S_i: the set of sessions peer i is participating in
P_i: the highest priority level among sessions in S_i
inQueues[c]: the set of incoming data message queues at priority level c, in which each queue contains the incoming messages for one session in S_i
outQueues[c]: the set of outgoing data message queues at priority level c, in which each queue contains the outgoing messages of a session in S_i to one downstream peer
$I[c]$: the number of peer i's incoming queues at priority level c
$O[c]$: the number of peer i's outgoing queues at priority level c
timeForNextMsg[q]: time to send the next message for queue q
$x_s^{(ij)}$: allocated bandwidth to downstream peer j in session s

Incoming scheduler:

for $c = P_i$ to 1
    while the queues in *inQueues*[c] are not empty
        for $q = 1$ to $I[c]$
            if the incoming queue *inQueues*[c][q] is not empty
                $m \leftarrow$ first message in *inQueues*[c][q]
                dequeue and process m
            end if
        end for
    end while
end for

Outgoing scheduler:

for $c = P_i$ to 1
    for $q = 1$ to $O[c]$
        while current time $\geq$ *timeForNextMsg*[q]
            $m \leftarrow$ first message in the queue *outQueues*[c][q], which belongs to session s and is destined to downstream peer j
            dequeue m and send m to j
            timeForNextMsg[q]$\leftarrow$ *timeForNextMsg*[q] + $\frac{m.size}{x_s^{(ij)}}$
        end while
    end for
end for

Then, in the optimal bandwidth allocation protocol executed at the next time slot, less upload bandwidth will be allocated along this link, as we consider link delays in the utility functions of the optimization.

Besides, inside the time slot, such congestion is actively detected and adapted as well, in the follow way: Each peer periodically estimates its achieved receiving rate in each session from each upstream peer. When its receiving rate is less than the allocated upload bandwidth, which is the sending rate at the upstream peer, we know congestion has occurred along the link. The downstream peer then feeds back this rate discrepancy to the sender. At the upstream peer, it decreases its allocated bandwidths along the congested links correspondingly. When multiple sessions are sharing a same link, it starts to reduce bandwidth from those sessions at the lowest priority level. Meanwhile, an affected downstream peer will actively seek more upload bandwidth from its other upstream peers, so as to compensate its rate loss.

With such adaptation to network dynamics inside and across time slots, *Diverse* maximally ensures the service quality for high-priority sessions and is able to achieve excellent service differentiation results at all times.

V. EXPERIMENTAL EVALUATION

With the C++ programming language, we have implemented *Diverse* protocols in an emulated peer-to-peer overlay environ-

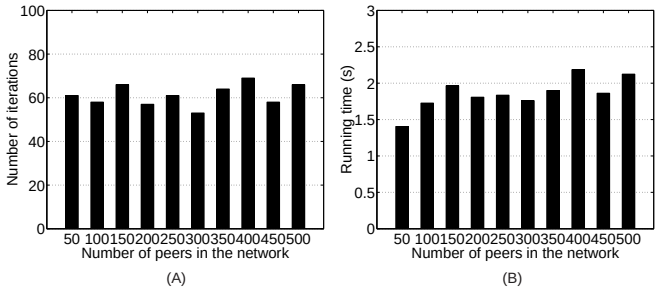

Fig. 5. Convergence speed in static networks.

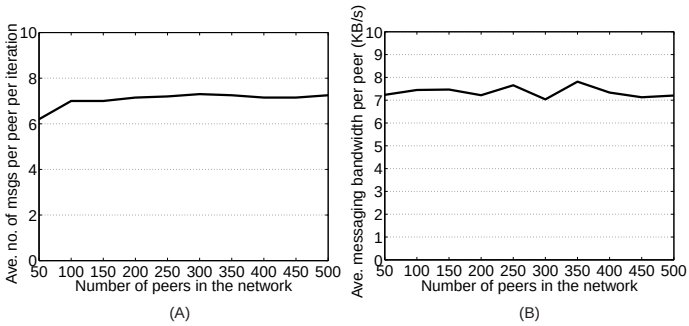

Fig. 6. Communication overhead in static networks.

ment. Our implementation includes the priority-based message scheduling and queue administration in the application layer, and the distributed optimal bandwidth allocation protocol at each peer. Our implementation also provides measurement of QoS metrics and support for network parameter emulation, such as peer upload/download capacities. It runs in the UNIX operating system (Linux, Mac OS and other BSD variants), and uses standard Berkeley sockets to establish TCP connections between peers. Actual data messages in each peer-to-peer session are transmitted from the source, scheduled at intermediate peers, and received at the receiving peers.

With *Diverse* implementation, we have conducted extensive experimental evaluations on a high-performance cluster consisting of 50 Dell 1425SC and Sun v20z dual-CPU servers, each equipped with dual Intel Pentium 4 Xeon 3.6GHz and dual AMD Opteron 250 processors. In order to emulate our protocols in realistic network settings, we generate random network topologies based on power-law degree distributions with the Boston topology generator, *BRITE* [15]. In each topology, a peer has six neighbors on average. Download and upload capacities for the peers are uniformly chosen from the range of $1.5 - 4.5$ Mbps and $0.6 - 0.9$ Mbps, respectively. Overlay link delays are uniformly selected between 1 ms and 10 ms. Except for the experiments in Sec. V-D.2, we run two sessions in each network: Session 1, a high-priority streaming session with priority $C_1 = 2$; Session 2, a low-priority data downloading session, with priority $C_2 = 1$. In each network, there is one source for each session, and the remaining peers participate in both sessions.

A. Performance of optimal bandwidth allocation in static networks

We first investigate the convergence speed and messaging overhead of *Diverse*'s optimal bandwidth allocation protocol in static networks of different numbers of peers. In each of the static networks, the protocol runs from the beginning, where all the prices μ are initialized to zero. Maximum session rates of the streaming and downloading sessions are set to 500 Kbps and 1 Mbps respectively in this experiment.

In Fig. 5, we observe that the number of iterations executed for algorithm convergence remains almost the same for networks of peer numbers from 50 up to 500. Also, the total running time to converge does not change much, which reveals that the running time for each iteration is similar regardless of the network sizes. Such results meet our expectation, as our optimal bandwidth allocation algorithm is fully decentralized, and thus exhibit its outstanding scalability as the network size increases.

The communication overhead for sending *PU* and *BU* messages in the protocol is illustrated in Fig. 6. Fig. 6 (A) shows that each peer sends out $6 - 7$ messages on average in each iteration in all the networks, as the average edge density is six links per peer, and communication mainly occurs among neighboring peers. Fig. 6 (B) shows the average messaging bandwidth used at each peer for the protocol execution. As a *PU* or *BU* message is simply composed of an application-layer message header of 24 bytes and a payload of a few double values, the messaging overhead is as low as $7-8$ KB/s, which is trivial compared to the data rates in the sessions. Also noting that the protocol execution takes only a very short period of time, we conclude that our distributed protocol to derive the optimal bandwidth allocation is indeed very lightweight.

B. Performance of optimal bandwidth allocation in dynamic networks

We next examine the dynamic behavior of our protocol in practical networks with peer joins and departures. We investigate two peer dynamics scenarios. In each scenario, we consider two sessions, streaming and downloading as stated earlier, with maximum rates of 500 Kbps and 1 Mbps respectively. The length of each time slot is set to 30 seconds. Based on discussions in Sec. IV-C, our optimal bandwidth allocation protocol is executed at the beginning of each time slot, based on peer dynamics occurred in the previous time slot, and from the previous optimal values. Meanwhile, our dynamics handling protocol also actively adapts to peer dynamics in each time slot.

1) Scenario 1: In the first scenario, both sessions run for a total of 22.5 minutes. In the first 10 minutes, 200 peers join the sessions at random time; then starting from 12.5 minutes, peers start to leave the network.

The results of this experiment are summarized in Fig. 7. Fig. 7 (A) shows the dynamics of peer numbers in the network. Fig. 7 (B) illustrates the convergence of price vector μ from its previous optimal values in each time slot during the entire process, with the additional number of iterations for the convergence shown in Fig. 7 (C). We can see that in each time slot, convergence to a new optimal price vector from the previous one takes much fewer iterations and much shorter time, compared to running from the very beginning in

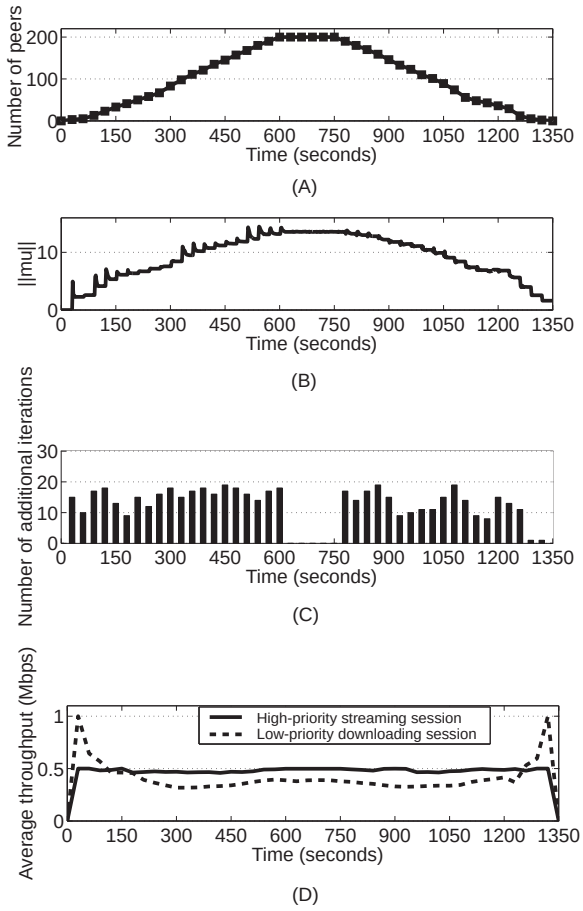

Fig. 7. Convergence during a peer join phase and a departure phase: a 200-peer dynamic network with two sessions.

the cases of static networks of the same sizes. As peer joins and departures are practical scenarios in realistic peer-to-peer networks, this suggests that our optimal bandwidth allocation algorithm can actually delivery good performance in practice.

The average throughput at each peer in each of the two sessions is illustrated in Fig. 7 (D). We note that throughput for the high-priority streaming session always remains at the required streaming rate of 500 Kbps. For the data downloading session, at the beginning when there are few peers and more spare upload capacities in the network, it is able to achieve high downloading throughput. However, with more peers joining and competing for bandwidth, its downloading throughput decreases. It will increase again when more bandwidth capacities are provided in the network due to further peer joining. Similarly, in the peer departure phase, the downloading session loses bandwidth when there are less upload bandwidth in the network, and picks up the spare bandwidth when more bandwidth capacities are left.

2) Scenario 2: In the second scenario, during a 10-minute period, 200 peers join and leave the sessions following an ON/Off model, with On/Off periods both following an exponential distribution with an average of T seconds. We monitor the achieved throughput at each peer in both sessions during this process and illustrate the average throughput in Fig. 8.

The results demonstrate that our protocol maximally guarantees steady throughput for the sessions even under high peer

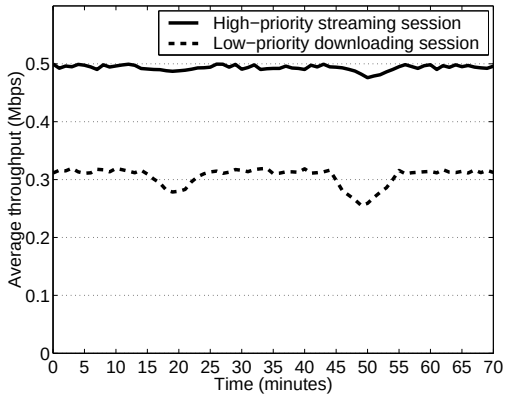

Fig. 9. Average throughput in case of congestion: a 100-peer network with two sessions.

churn rates, based on the combination of optimal bandwidth allocation and active dynamics handling. For example, in the case that each peer joins/leaves every 20 seconds, on average there are 10 joins/departures every second in such a 200-peer network. We can see the throughput achieved at existing peers in the high-priority streaming session is still quite consistent, while that for the downloading session represents more fluctuations, due to our priority-based dynamics handling.

C. Adaptation to network congestion

In our evaluations, we have also investigated the impact of varying available overlay link capacities on the optimal upload bandwidth allocation. We experimented with realistic link bandwidths, chosen from the distribution of measured available bandwidth between PlanetLab nodes [16]. However, we find the bandwidths are generally much larger than the session rates, so their slight variations do not affect our results much. Therefore, we omit the experiment results for this case. Given the over provisioning situation in the core of the current Internet, we believe such findings from available bandwidth distribution among PlanetLab nodes can represent the general scenario.

Still, we examine the adaptation of *Diverse* in the case that severe network congestion occurs somewhere at an underlying router. For this purpose, we design the following experiment: We introduce congestion to a steady-stage peer-to-peer network of 100 peers, where the average throughput achieved at peers in the two sessions are around 500 Kbps and 310 Kbps respectively. In a period of 70 minutes, congestion occurs twice in the underlying IP network, at $15-25$ and $45-55$ minutes respectively, and affects 10% and 20% overlay links, respectively.

From Fig. 9, we observe that the throughput for the high-priority session is almost never affected during the first congestion period and is slightly affected during the second. This is achieved at the cost of the low-priority session, whose throughput is reduced whenever congestion occurs.

The observations from Sec. V-B and V-C clearly demonstrate the effectiveness of our priority-based optimal bandwidth allocation and dynamics handling protocols, which consistently guarantees the delivery rate for high-priority sessions

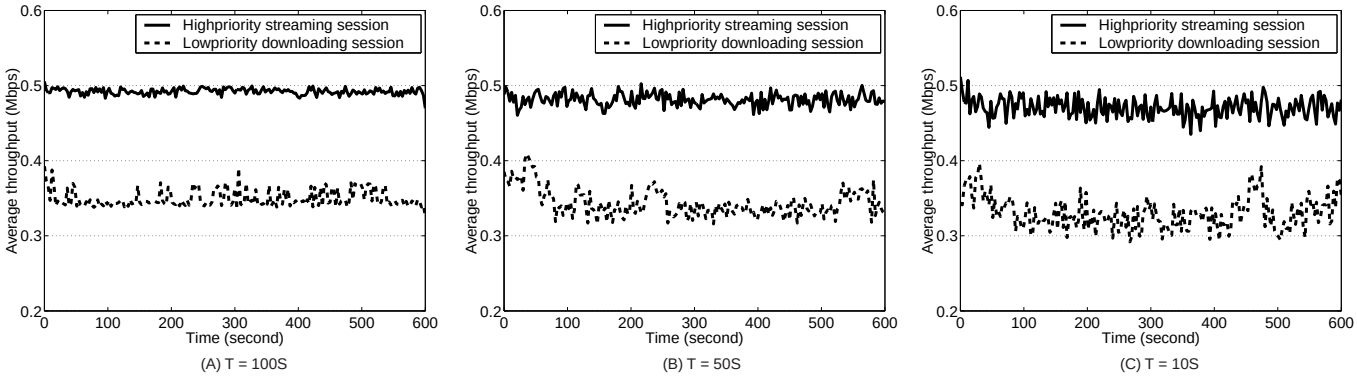

Fig. 8. Average throughput with different on/off interval lengths: a 200-peer dynamic network with two sessions.

in cases of all kinds of network dynamics, and also maximize the utilization of network bandwidths.

D. Effectiveness of application-layer service differentiation

We next design two experiments to further evaluate the efficiency of *Diverse* in improving service differentiation.

1) Comparison among four schemes: The first experiment aims at investigating the effects of both overlay priority scheduling and optimal bandwidth allocation protocols. Towards this objective, we compare end-to-end delays in two peer-to-peer sessions with four different schemes:

- ▷ *NSNA*: No overlay priority scheduling and no optimal bandwidth allocation. Messages of the two sessions are processed in a round robin fashion and they fairly share the upload bandwidth of their common upstream peers.
- ▷ *WSNA*: Overlay priority scheduling is applied, but not the optimal bandwidth allocation protocol.
- ▷ *NSWA*: Priority-based optimal bandwidth allocation protocol is applied, but not overlay priority scheduling.
- ▷ *Diverse*: Both overlay priority scheduling and optimal bandwidth allocation are applied.

In this experiment, 100 peers sequentially join the two sessions in 30 minutes. The maximum rates for both sessions are 500 Kbps, and the size of data messages is 5K bytes. We monitor the average end-to-end delay per peer in both sessions, which is computed as the *bit transmission delay* (*BTD*), *i.e.*, message delay divided by the message size.

Fig. 10 shows the average end-to-end delay consistently increases due to the expansion of network diameter with peer joins. With *NSNA*, the two sessions are not differentiated and thus their curves overlap each other. By applying overlay priority scheduling in *WSNA*, the delay in the high-priority session is slightly reduced and that for the low-priority session increases. With the schemes where optimal bandwidth allocation is applied, *i.e.*, *NSWA* and *Diverse*, there is a distinct drop of end-to-end delays in the high-priority session and a corresponding notable delay increase in the low-priority session. This is because our optimal bandwidth allocation protocol always chooses the best low-delay paths and guarantees the maximum rate for the high-priority session. A comparison between *WSNA* and *NSWA* further reveals that the end-to-end delays in peer-to-peer sessions are more dominated by transmission rates and overlay hop counts, than

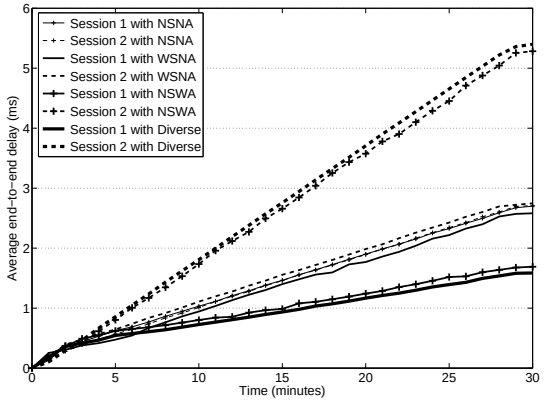

Fig. 10. Comparison among four service differentiation schemes: a 100-peer joining case with two sessions.

by queueing delays at the peers. Therefore, in *Diverse*, the optimal bandwidth allocation protocol plays a significant role in achieving service differentiation across sessions. Nevertheless, the combination of overlay priority scheduling and optimal bandwidth allocation is able to achieve the best results.

2) Comparison across different numbers of sessions: In the second experiment, we compare the effects of service differentiation achieved by *Diverse* in cases that there are more than two concurrent sessions in the peer-to-peer network.

We deploy multiple sessions in a peer-to-peer network with 50 peers. Each session is at a different priority level, whose priority decreases with the increase of session indices. The maximum rates of all the sessions are 500 Kbps. At the steady stage of the sessions, we measure the average throughput achieved and the average end-to-end delay experienced by each peer in each session. Again, the end-to-end delay is computed as the end-to-end *BTD*.

Fig. 11 (A) shows that even with more sessions competing for peer bandwidth, the session with the highest priority can always achieve a throughput near its maximum session rate, while the allocated bandwidths for lower-priority sessions are reduced to accommodate the additional sessions. Correspondingly in Fig. 11 (B), with the increase of session numbers, delay in low-priority sessions increases rapidly, while that for the highest-priority session remains unchanged. These results reveal that *Diverse* can effectively provide differentiated performance for multiple sessions based on their priority levels.

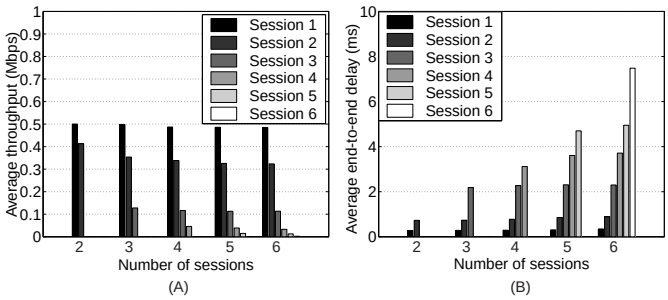

Fig. 11. Comparison across different numbers of sessions: a 50-peer network.

VI. Related Work

There exists little literature that touches upon the topic of application-layer service differentiation in peer-to-peer networks. Their main focus has been on the assignment of priorities to different peers and the provision of differentiated service quality to them thereafter [17], [18], [19]. As these priorities are usually decided by their contribution of services and resources, such service differentiation serves as a natural incentive to encourage the peers to cooperate. Ma *et al.* [17] propose such a peer-to-peer service differentiation scheme, which distributes the bandwidth among competing peers by designing a competition game for them. Similarly, Gupta *et al.* [18] provide differentiated service quality levels to peers during the content searching and downloading process, based on their reputation scores, which are determined by the satisfaction of other peers with their contributed services.

There are fundamental differences between our work and existing literature. Other than prioritizing peers, we assign different priority levels to different peer-to-peer *communication sessions*, and provision differentiated Quality of Service for each session. Further, we guarantee better service qualities for high-priority sessions across the entire peer-to-peer network, and aim at achieving global utility maximization. Existing work usually only considers a one-hop scenario, where multiple peers are directly retrieving from the source [17].

To the best of our knowledge, we are not aware of any existing work that discusses service differentiation across multiple peer-to-peer sessions with application-layer approaches. One piece of slightly similar work is by Key *et al.* [20], which regulates the rates of "background" data transfer sessions, such as downloading operating system updates, by adjusting their receiver windows. In this way, transfer of such low-priority flows only dynamically utilize any available bandwidth, without affecting active sessions of high-priority flows. Though this paper considers a client-server model, its result is remotely similar to that achieved by *Diverse*, when only two priority levels are considered in the peer-to-peer overlay. However, its proposed mechanism still requires changing the TCP receiver window. In contrast, rate regulation in *Diverse* is computed with the optimal bandwidth allocation algorithm and implemented at the outgoing scheduler of each peer, based on the calculation of the time to send the next message. *Diverse* does not involve any changes to the network protocol stack, yet still able to adapt to network dynamics.

In the general area of enhancing service QoS with application-layer approaches, Service Overlay Networks [21] purchase bandwidth from network domains to build service networks which guarantee quality for QoS-sensitive services. This proposal relies on bandwidth provisioning from the underlying domains to meet performance requirements of these services. As another representative piece in this area, OverQoS [22] aims at improving the perceived QoS of a session by prioritizing packets within the session itself, other than improving performance for high-priority sessions at the cost of low-priority sessions. In this way, it reduces the loss rate of important packets at the expense of less important ones.

Overlay bandwidth allocation has been utilized to achieve differentiation of service quality towards different peers [17], [19], or to improve the overall performance in peer-to-peer sessions [23], [24], [25]. In the former case, Clevenot *et al.* [19] formulate the bandwidth allocation problem in a multi-class peer-to-peer network into systems of differential equations. They mainly address the problem in static settings with a centralized solution, without considering any peer dynamics. In our work, we achieve optimal bandwidth allocation by maximizing the overall utilities in a convex program. Compared to previous work that uses optimization models to improve performance — such as maximizing throughput or minimizing cost — in overlay multicast [23], [24], [25], our optimization model is tailored to the special requirements of service differentiation across multiple peer-to-peer sessions in heterogeneous peer-to-peer networks. We have also derived a fully distributed protocol to achieve optimality, as well as to adapt to network dynamics. These realistic concerns have not been addressed in previous optimization-based approaches, most of which are largely theoretical in nature.

Finally, in *Diverse*, we have introduced overlay scheduling algorithms to achieve better service differentiation. Based on our knowledge, this is the first paper that investigates benefits of applying priority scheduling at the application layer in a peer-to-peer network. Combining overlay scheduling with optimal bandwidth allocation, we take advantage of every tool at our disposal within the application layer, in order to provide differentiated service qualities for multiple concurrent peer-to-peer sessions with different performance requirements.

VII. Conclusion

We conclude this paper with the belief that application-layer service differentiation is an effective mechanism to compensate the lack of such support in the current IP Internet infrastructure. *Diverse* pioneers this direction of research, which addresses the challenges of providing differentiated service qualities for peer-to-peer communication sessions over the best effort Internet. In *Diverse*, its optimal bandwidth allocation algorithm efficiently constructs optimal session topologies that maximizes priority-based utilities. In addition, priority scheduling is applied at the peers in the application layer, which further improves the service quality experienced by high-priority sessions. With examples, analysis and experimental results using a realistic implementation, we have demonstrated a complete *Diverse* mechanism, that effectively diversifies service quality of different sessions. *Diverse* is fully distributed, optimal and adaptive to dynamics at the same

time. We believe that our positive experimental results — from emulating realistic peer-to-peer environments in high-performance server clusters — have revealed the effectiveness of *Diverse* in a real-world scenario. As part of our future work, we plan to deploy the *Diverse* implementation in large-scale peer-to-peer applications over the Internet.

REFERENCES

[1] Y. Chu, S. Rao, and H. Zhang, "A Case for End System Multicast," in *Proc. of ACM SIGMETRICS'00*, June 2000.

[2] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in *Proc. of ACM SIGCOMM 2001*, September 2001.

[3] "Gnutella: Distributed Information Sharing," http://gnutella.wego.com, 2000.

[4] "Bittorent," http://www.bittorrent.com.

[5] X. Zhang, J. Liu, B. Li, and T. P. Yum, "CoolStreaming/DONet: A Data-Driven Overlay Network for Live Media Streaming," in *Proc. of IEEE INFOCOM 2005*, March 2005.

[6] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "PROMISE: Peer-to-Peer Media Streaming Using CollectCast," in *Proc. of ACM Multimedia 2003*, November 2003.

[7] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," RFC 2475, Oct. 1998.

[8] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," Internet RFC 1633, June 1994.

[9] J. Crowcroft and P. Oechslin, "Differentiated End-to-end Internet Services Using a Weighted Proportional Fair Sharing TCP," *ACM Computer Communication Review*, 28(3), July 1998.

[10] S. Shenker, "Future Design Issues for the Future Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1176–1188, 1995.

[11] F. Kelly, A. Maulloo, and D. Tan, "Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability," *Journal of the Operational Research Society*, pp. 237–252, 1998.

[12] N. Z. Shor, *Minimization Methods for Non-Differentiable Functions*, Springer-Verlag, 1985.

[13] D. Bertsekas, *Nonlinear Programming*, Athena Scientific, 1995.

[14] S. Boyd, *Convex Optimization*, Cambridge University Press, 2004.

[15] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: Boston University Representative Internet Topology Generator," Tech. Rep., http://www.cs.bu.edu/brite, 2000.

[16] "PlanetLab IPerf," http://jabber.services.planet-lab.org/php/iperf/.

[17] R. T. B. Ma, S. C. M. Lee, J. C. S. Lui, and D. K. Y. Yau, "A Game Theoretic Approach to Provide Incentive and Service Differentiation in P2P Networks," in *Proc. of ACM SIGMETRICS/Performance'04*, June 2004.

[18] M. Gupta and M. Ammar, "Service Differentiation in Peer-to-Peer Networks Utilizing Reputations," in *Proc. of ACM Fifth International Workshop on Networked Group Communications (NGC 2003)*, September 2003.

[19] F. Clevenot, P. Nain, and K.W. Ross, "Multiclass P2P Networks: Static Resource Allocation for Bandwidth for Service Differentiation and Bandwidth Diversity," in *Proc. of Performance 2005*, October 2005.

[20] P. Key, L. Massoulie, and B. Wang, "Emulating Low-priority Transport at the Application Layer: A Background Transfer Service," in *Proc. of ACM SIGMETRICS/Performance'04*, June 2004.

[21] Z. Duan, Z. Zhang, and Y. T. Hou, "Service Overlay Networks: SLAs, QoS, and Bandwidth Provisioning," *IEEE/ACM Transactions on Networking*, vol. 11, no. 6, pp. 870–883, December 2003.

[22] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz, "OverQoS: An Overlay Based Architecture for Enhancing Internet QoS," in *Proc. of the 1st Symposium on Networked Systems Design and Implementation (NSDI)*, March 2004.

[23] Y. Cui, Y. Xue, and K. Nahrstedt, "Optimal Resource Allocation in Overlay Multicast," in *Proc. of 11th International Conference on Network Protocols (ICNP 2003)*, November 2003.

[24] Y. Cui, B. Li, and K. Nahrstedt, "On Achieving Optimized Capacity Utilization in Application Overlay Networks with Multiple Competing Sessions," in *Proc. of 16th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2004)*, Barcelona, Spain, June 2004.

[25] Z. Li and B. Li, "Efficient and Distributed Computation of Maximum Multicast Rates," in *Proc. of IEEE INFOCOM 2005*, March 2005.

APPENDIX I
PROOF OF THEOREM 1

Proof: Let $x_s^{(ij)*}$, $\forall j : (i,j) \in A_s$, $\forall s \in S$, be the optimal solution to (6). Introducing Lagrangian multiplier λ for the constraint in (7) and $\nu_s^{(ij)}$ for constraints in (8), we obtain the KKT conditions for the convex problem in (6) as follows (pp. 244, [14]):

$$\sum_{s\in S}\sum_{j:(i,j)\in A_s} x_s^{(ij)*} \le O_i,$$

$$x^* \ge 0, \lambda^* \ge 0, \nu^* \ge 0,$$

$$\lambda^*(\sum_{s\in S}\sum_{j:(i,j)\in A_s} x_s^{(ij)*} - O_i) = 0, \tag{11}$$

$$x_s^{(ij)*}\nu_s^{(ij)*} = 0, \forall j : (i,j) \in A_s, \forall s \in S, \tag{12}$$

$$-(U_s^{'(ij)}(x_s^{(ij)*}) - \mu_s^j) + \lambda^* - \nu_s^{(ij)*} = 0,$$
$$\forall j : (i,j) \in A_s, \forall s \in S. \tag{13}$$

For $x_s^{(ij)*} > 0$, we have $\nu_s^{(ij)*} = 0$ from (12), and $U_s^{'(ij)}(x_s^{(ij)*}) - \mu_s^j = \lambda^*$ from (13). Since $U_s^{(ij)}$ is increasing, strictly concave and twice differentiable, the inverse function of $U_s^{'(ij)}$, i.e., $U_s^{'(ij)-1}$, exists. Then we have

$$x_s^{(ij)*} = \begin{cases} 0 & \text{if } \lambda^* \ge U_s^{'(ij)}(0) - \mu_s^j, \\ U_s^{'(ij)-1}(\lambda^* + \mu_s^j) & \text{if } \lambda^* < U_s^{'(ij)}(0) - \mu_s^j. \end{cases} \tag{14}$$

Therefore, in order to achieve a same marginal utility value λ^* for all the positive $x_s^{(ij)}$'s, we always decrease the largest marginal utility $U_s^{'(ij)}(x_s^{(ij)}) - \mu_s^j$ by increasing the corresponding $x_s^{(ij)}$. In this way, we are reducing the marginal utilities towards the same value of λ^*. If $\lambda^* > 0$, we will finally have $\sum_{s\in S}\sum_{j:(i,j)\in A_s} x_s^{(ij)*} - O_i = 0$ based on (11); if $\lambda^* = 0$, all the marginal utilities for positive $x_s^{(ij)}$'s will be reduced to zero. Therefore, this water-filling process continues until O_i is used up, i.e., $\sum_{s\in S}\sum_{j:(i,j)\in A_s} x_s^{(ij)*} = O_i$, or all the marginal utilities, $U_s^{'(ij)}(x_s^{(ij)*}) - \mu_s^j, \forall j : (i,j) \in A_s, \forall s \in S$, are zero.

□

APPENDIX II
PROOF OF THEOREM 2

Proof:

We first prove that the sequence of vectors $\{\mu[k]\}$ converges to its optimum μ^*. We know μ^* is bounded, as it is lower bounded by 0 and also upper bounded based on the interior point assumption (pp. 226 [14]). Further, because the bandwidth variables $x_s^{(ij)}$'s are bounded, the dual subgradients $\sum_{i:(i,j)\in A_s} x_s^{(ij)} - R_s$, $\forall j \in T_s, \forall s \in S$, are bounded too. Therefore, based on Theorem 3 in [12] (pp. 26), by the subgradient algorithm with step length specified by (10), the Lagrangian multiplier vector $\{\mu[k]\}$ converges to μ^*.

Then under the assumption that $U_s^{(ij)}$ is increasing, strictly concave and twice differentiable, $U_s^{'(ij)-1}$ exists and is continuous and one-to-one. Based on Eq. (14) in Appendix I, given $\mu[k]$, there is a unique maximizer $x[k]$ for the Lagrangian subproblem in (5). Therefore, we have $x = g(\mu)$, where g is

a continuous function decided by (14). Let x^* be the unique maximizer of the Lagrangian subproblem in (5) for μ^*, *i.e.*, $x^* = g(\mu^*)$. Based on the KKT conditions for convex program **P** in (1), we know an optimal solution to **P** is also a maximizer to its Lagrangian subproblem in (5) with μ^* (pp. 244, [14]). Therefore, x^* is the unique optimal solution to **P**. Further, since g is a continuous function, we conclude that the sequence $x[k]$ converges to the primal optimum x^*.

□

Chuan Wu. Chuan Wu received her B.Engr. and M.Engr. degrees from Department of Computer Science and Technology, Tsinghua University, China, in 2000 and 2002, respectively. She is currently a Ph.D. candidate at the Department of Electrical and Computer Engineering, University of Toronto, Canada. Her research interests include distributed algorithm and protocol design to improve Quality of Service of overlay multicast applications. She is particularly interested in deriving insights from optimization and game theory to guide practical protocol design.

Baochun Li. Baochun Li received his B.Engr. degree in 1995 from Department of Computer Science and Technology, Tsinghua University, China, and his M.S. and Ph.D. degrees in 1997 and 2000 from the Department of Computer Science, University of Illinois at Urbana-Champaign. Since 2000, he has been with the Department of Electrical and Computer Engineering at the University of Toronto, where he is currently an Associate Professor. He holds the Nortel Networks Junior Chair in Network Architecture and Services since October 2003, and the Bell University Laboratories Chair in Computer Engineering since July 2005. In 2000, he was the recipient of the IEEE Communications Society Leonard G. Abraham Award in the Field of Communications Systems. His research interests include application-level Quality of Service provisioning, wireless and overlay networks. He is a senior member of IEEE, and a member of ACM. His email address is `bli@eecg.toronto.edu`.