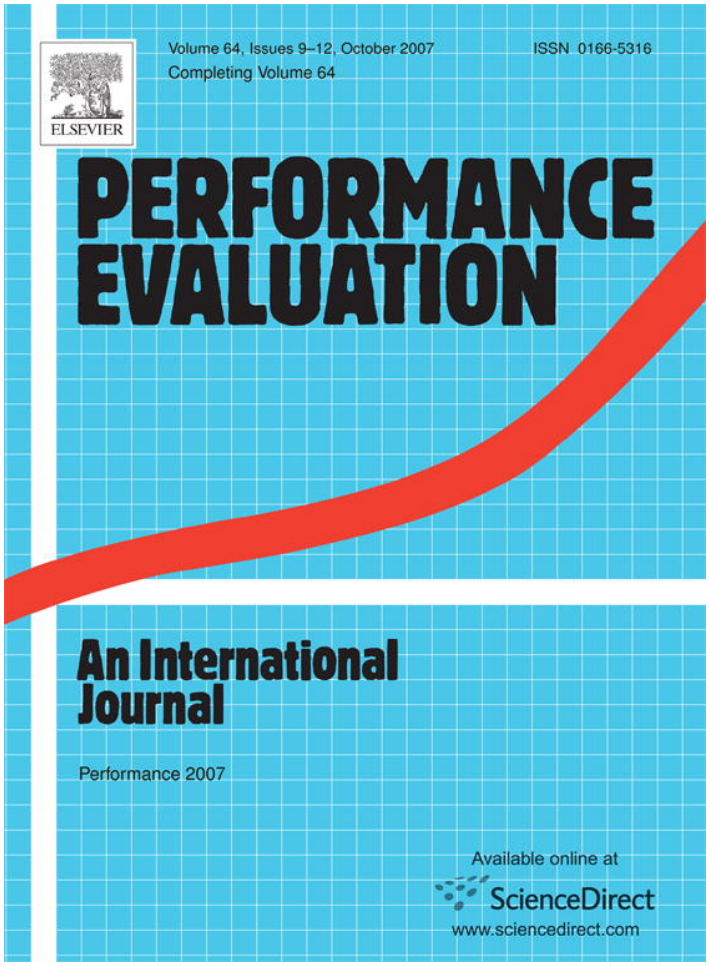


Available online at www.sciencedirect.com
ScienceDirect
Performance Evaluation 64 (2007) 965–977
PERFORMANCE EVALUATION
An International Journal
www.elsevier.com/locate/peva

Application-layer multipath data transfer via TCP: Schemes and performance tradeoffs

Bing Wang[a,*], Wei Wei[b], Jim Kurose[c], Don Towsley[c], Krishna R. Pattipati[d], Zheng Guo[a], Zheng Peng[a]

[a] *Computer Science and Engineering Department, University of Connecticut, Storrs, CT 06269, United States*
[b] *United Technologies Research Center, East Hartford, CT 06108, United States*
[c] *Computer Science Department, University of Massachusetts, Amherst, MA 01003, United States*
[d] *Electrical and Computer Engineering Department, University of Connecticut, Storrs, CT 06269, United States*

Available online 26 June 2007

Abstract

For applications involving data transmission from multiple sources, an important problem is: when sources are allowed to use multiple paths, how does one select paths and control the sending rates on the paths to maximize the aggregate sending rate of the sources? We consider this problem in the context of an overlay network by allowing a source to send data over k ($k \geq 1$) overlay paths to its destination. This problem is NP-hard, and we develop an *iterative distributed* heuristic to solve it. In each iteration, we first select paths and then control the sending rates on the multiple paths to maximize the aggregate sending rate of the sources. For rate control, we develop an *application-level multipath rate controller via TCP*. This controller is easy to deploy and maximizes the aggregate sending rate of the sources in certain settings. To the best of our knowledge, this is the first distributed application-level controller with such an optimality property. For path selection, we prove that the problem of optimal overlay path selection is NP-hard and propose randomized path-selection algorithms. Our performance evaluation demonstrates that our iterative heuristic performs very well in a wide range of settings. Furthermore, a small number of paths, 2–4, and a small amount of extra bandwidth in the network are sufficient to realize most of the performance gains.

Keywords: Multipath data transfer; Application-level rate control; Path selection

1. Introduction

A wide range of applications require data transmission from geographically distributed sources to one destination or multiple destinations using the Internet. For instance, in the Engineering Research Center for Collaborative Adaptive Sensing of the Atmosphere (CASA) [1], multiple X-band radar nodes are placed at geographically distributed locations, each remotely sensing the local atmosphere. Data collected at these radar sites are transmitted to a central

* Corresponding author.
E-mail addresses: bing@engr.uconn.edu (B. Wang), weiw@utrc.utc.com (W. Wei), kurose@cs.umass.edu (J. Kurose), towsley@cs.umass.edu (D. Towsley), krishna@engr.uconn.edu (K.R. Pattipati), guozheng@engr.uconn.edu (Z. Guo), zhengpeng@engr.uconn.edu (Z. Peng).

0166-5316/$ - see front matter
doi:10.1016/j.peva.2007.06.013

destination or multiple destinations using a state-wide public network for hazardous weather detection. In another example, high-volume astronomy data are stored at multiple geographically distributed locations (e.g., the Sloan Digital Sky Survey data). Scientists may need to retrieve and integrate data from archives at several locations for temporal and multi-spectral studies using the Internet (e.g., via SkyServer [2]). In yet another example, an ISP places multiple data monitoring sites inside its network. Each monitoring site collects traffic data and transmits them to a central location for analysis and network diagnosis.

A crucial factor for the success of the above applications is efficient data transfer from the multiple sources to the destination(s). In these applications, the sources and destinations typically have high access bandwidths, while non-access links may limit the sending rate of the sources as indicated by recent measurement studies [3]. This is clearly true in CASA: the sending rates of the radar nodes are constrained by low-bandwidth links inside the public network. When the bandwidth constraints are inside the network, using multiple paths between a source and destination can provide a much higher throughput [4,5]. The problem we address is: *when sources are allowed to use multiple paths, how does one select paths and control the sending rate on the paths to maximize the aggregate sending rate of the sources?*

Our focus is on scenarios where multiple paths between a source and destination are formed using an overlay network, which has been shown to be an effective multipath architecture for throughput improvement ([5] shows an improvement of 20%–55% over that using a single path). More specifically, we address the following problem. Consider a set of sources and a set of relays. Each source has its corresponding destination. A network path via one relay or multiple relays is referred to as an *overlay path*. A source selects k ($k \geq 1$) overlay paths and spreads its data among these paths. (The restriction on the number of overlay paths is to limit overheads, e.g., meta data required to reassemble data at the destination.) Our goal is to select the overlay paths for each source and control the sending rate on the paths to maximize the aggregate sending rate of the sources.

The above problem is NP-hard since it contains the unsplittable flow problem (i.e., $k = 1$) as a special case [6]. Because centralized algorithms require complete knowledge of the network and are often unrealistic in practice, we develop an *iterative distributed heuristic* to solve the above problem. In each iteration, we first select paths (referred to as *path selection*) and then control the sending rates on the multiple paths (referred to as *rate control*) to maximize the aggregate sending rate of the sources. The iteration continues until all source demands are satisfied (i.e., each source achieves its maximum sending rate). We require the rate-control algorithm to run at the application level, on top of TCP. This is motivated by several reasons. First, the applications in consideration require reliable data transfer, which makes TCP a natural choice. Secondly, application-layer approaches via TCP are easier to deploy than control approaches at lower levels. Furthermore, all applications in the Internet are expected to be TCP-friendly [7], and using TCP is by definition TCP-friendly. On the other hand, TCP is a rate controller that runs on a *single* path between a *single* source–destination pair. Designing an application-level *multipath* rate controller on top of TCP to maximize the aggregate sending rate of *multiple* sources is a challenging task. Our main contributions are:

- We develop a *distributed application-level multipath* rate controller that runs on top of TCP. This controller does not require explicit network knowledge (e.g., topology, available bandwidth). Furthermore, we prove that it maximizes the aggregate sending rate of the sources in settings with two logical hops and a single destination. To the best of our knowledge, this is the first distributed application-level controller with the above optimality property.
- We prove that, for any given rate controller, the problem of optimal overlay path selection is NP-hard even in extremely simple settings. Motivated by the results in [8], we propose randomized path-selection algorithms.
- We evaluate the performance of our rate-control and path- selection algorithms using a combination of numerical study and *ns-2* simulation. Our performance evaluation demonstrates that our iterative heuristic performs very well in a wide range of settings. Furthermore, a small number of paths, 2–4, and a small amount of extra bandwidth in the network (compared to source demands) are sufficient to realize most of the performance gains.

Related work. The studies of [9–11] consider multipath routing at the network layer, as an improvement to the single-path IP routing. We, in contrast, consider multipath data transfer at the application level, without any change to IP routing; consequently, our approach is easy to deploy. The studies of [12] and [13] focus on data uploading and replication respectively, allowing a source to use multiple paths inside an overlay network. They develop *centralized* algorithms to minimize the transfer time. Our focus is on developing efficient *distributed* algorithms to maximize the aggregate sending rate of the sources.

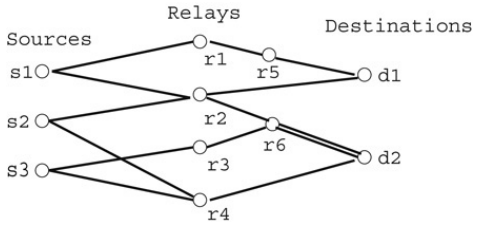


Fig. 1. Problem setting, $k = 2$ in this example.

Several studies developed distributed multipath rate controllers by solving optimization problems to maximize the aggregate utility of the sources [14–21]. These controllers, however, require feedback from the network in the form of congestion prices and are difficult to realize in practice. Our application-level rate controller requires simple rate regulation at the application level and therefore is readily deployable. The study in [22] considers both path selection and rate control to locate and utilize the available capacity in the network. Their rate control relies on congestion price feedback from the network. Furthermore, they do not present complexity results on path selection or performance evaluation on path selection combined with rate control.

Most application-level rate-control algorithms via TCP are for a single source–destination pair (e.g., [23,24]) or parallel downloading to minimize the downloading time (e.g., [25]). To the best of our knowledge, our controller is the first distributed application-level controller that maximizes the aggregate sending rate of the sources. Finally, [26] points out the limitations of application-level approaches when the access network has high bandwidth fluctuations. Our study focuses on networks with stable access bandwidths where application-level approaches are suitable.

Roadmap. The rest of this paper is organized as follows. Section 2 presents the problem setting and describes an overview of our approach. Application-level multipath rate control and path selection are studied in Sections 3 and 4, respectively. Section 5 presents a performance evaluation of our distributed heuristic. Finally, Section 6 concludes this paper and presents future work.

2. Problem setting and approach

We now state the problem and provide a high-level description of our approach. Consider a set of sources S, each associated with a destination. Let D denote the set of destinations. A network path containing one relay or multiple relays is referred to as an *overlay path*. Let R denote the set of relays. A source selects k ($k \geq 1$) overlay paths and spreads its data over the overlay paths,[1] as illustrated in Fig. 1. The sources and destinations have high access bandwidths. The relays are placed (e.g., using techniques in [27]) such that multiple overlay paths do not share performance bottlenecks.

Let m_s denote the *demand* of source s, which may come from the bandwidth limit of the source or the data generation rate at the source. Let x_s denote the sending rate of source s, referred to as *source rate*. We have $x_s \leq m_s$. A source is *satisfied* if its maximum sending rate is achieved, i.e., $x_s = m_s$. Each source can send over k paths, indexed from 1 to k. We denote by *path rate* the rate at which a source sends data over a path. For source s, let x_{sj} denote its path rate on the jth path, $x_{sj} \geq 0$. Then, $x_s = \sum_{j=1}^{k} x_{sj}$.

The problem we address is how to select overlay paths and control the path rates to maximize the aggregate source rate. This is an NP-hard problem since it contains the unsplittable flow problem (i.e., $k = 1$) as a special case [6]. We develop a *distributed iterative* heuristic to solve it. In each iteration, we first select paths (i.e., *path selection*) and then control the sending rates on the multiple paths (i.e., *rate control*) to maximize the aggregate source rate. The iteration continues until all source demands are satisfied (at that time, the maximum aggregate source rate is achieved).

We next describe the problem formulations for path selection and rate control. For ease of exposition, we only consider sources using multiple paths; including single-path sources in the problem formulation is straightforward [28]. Let L denote the set of links in the network. The capacity of link l is c_l, $l \in L$. Let L_{sj} denote the set of links traversed by the jth path of source s. The path-selection problem determines L_{sj} for $s \in S$, $j = 1, \ldots, k$. For rate control, as mentioned earlier, we desire a distributed application-level rate controller. Let y_{sj} denote the sending rate that source s sets at the application level on path j. We have $\sum_{j=1}^{k} y_{sj} \leq m_s$. Furthermore, we have

[1] In practice, a source may also send data over its default IP path (i.e., the path from the source to its receiver determined by IP). We only consider overlay paths since our path selection selects overlay paths and rate control does not differentiate the default IP path and overlay paths.

$x_{sj} \le y_{sj}$, since x_{sj} represents the actual rate that source s sends into the network (i.e., the rate reaching the receiver) on path j, which cannot exceed that set at the application level. More specifically, the rate-control problem is stated as the following optimization problem:

$$\mathbf{P}: \text{ maximize: } \sum_{s \in S} x_s \tag{1}$$

$$\text{subject to: } x_s = \sum_{j=1}^{k} x_{sj}, \quad x_{sj} \ge 0, \ s \in S \tag{2}$$

$$x_{sj} \le y_{sj}, \quad s \in S, \ j = 1, \ldots, k \tag{3}$$

$$0 \le \sum_{j=1}^{k} y_{sj} \le m_s, \quad s \in S \tag{4}$$

$$\sum_{s, j: l \in L_{sj}} x_{sj} \le c_l, \quad \forall l \in L \tag{5}$$

where (5) describes the link capacity constraints.

The above formulation for rate control differs from that in [14–21] in two important aspects: (1) our objective is to maximize the aggregate source rate, instead of summing the typically strictly concave utility functions; (2) we maximize $\sum_{s \in S} \sum_{j=1}^{k} x_{sj}$ though setting the sending rates at the application level (i.e., y_{sj}'s), instead of controlling x_{sj}'s directly.

We describe our distributed application-level rate controller in Section 3. In Section 4, we analyse the complexity of path selection and describe our approach to selecting paths.

3. Application-level multipath rate control

For a given path selection, rate control determines the path rates for each source to maximize the aggregate source rate. The basic idea of our application-level multipath rate controller is the following: based on an initial valid rate allocation (i.e., satisfying the link capacity constraint (5)), each source independently probes for paths with *spare* bandwidths and increases its sending rates on those paths. The above may increase the aggregate source rate under two conditions: (1) an unsatisfied source increases its sending rate on one path; (2) a satisfied source increases its sending rate on one path, j, and correspondingly reduces its sending rate on another path, i; the leftover bandwidth on path i is then utilized by another source.

We now detail our algorithm. Each source divides time into control intervals (the lengths of control intervals for different sources need not to be the same). In the nth control interval, let $y_{sj}(n)$ and $x_{sj}(n)$ denote respectively the application-level and actual path rate for source s on path j. Source s sets $y_{sj}(n)$ at the beginning of the nth control interval and obtains an estimate of $x_{sj}(n)$ during the nth control interval. As described earlier, $x_{sj}(n) \le y_{sj}(n)$. At the beginning of the first control interval, a source selects one path and increases the sending rate on that path by a certain amount to probe for available network bandwidth (referred to as the *rate-increment step*). At the beginning of later control intervals, a source first checks the result of its rate increment in the previous control interval to adjust the various parameters (referred to as the *parameter-adjustment step*), and then performs the rate-increment step.

We now detail how an arbitrary source s performs the parameter-adjustment and rate-increment steps at the beginning of the nth control interval, $s \in S$, $n \ge 1$ (see Fig. 2). For ease of exposition, we define the following notation. In the nth control interval, let $p_{sj}(n)$ represent the probability that source s chooses to probe path j, let $g_s(n)$ denote the path that source s selects for bandwidth probing, and let $\beta_{sj}(n)$ denote a *rate increment term* associated with the jth path of source s, $\beta_{sj}(n) \ge 0$. Their initial values are set as follows: $y_{sj}(0)$ and $p_{sj}(0)$ can be set to any valid value, $\beta_{sj}(0)$ is set to 0 or a small positive value (e.g., 0.1). In the parameter-adjustment step, source s first checks whether the rate increment on the selected path in the previous control interval is successful or not, defined as follows. Suppose source s chooses path g in the $(n-1)$th control interval. Then we say that the rate increment is *successful* iff the rate increment is successfully achieved by the network, i.e., $x_{sg}(n-1)/y_{sg}(n-1) \ge 1-\delta$. Here δ is a small nonnegative constant, chosen to accommodate measurement noises and network delays. If the rate increment on the selected path is successful, the probability of choosing this path is doubled and the rate increment term is

```
for (j = 1; j ≤ k; j++) {
    p_sj(n) = p_sj(n − 1); y_sj(n) = y_sj(n − 1); β_sj(n) = β_sj(n − 1)
}
if (n == 1) Randomly select one path, recorded as g_s(n)
else {
  g = g_s(n − 1)
  if (x_sg(n − 1)/y_sg(n − 1) < 1 − δ) {
       y_sg(n) = (y_sg(n − 1) − ε_s)/(1 + β_sg(n − 1))
       p_sg(n) = p_sg(n − 1)/2
       Normalize p_sj(n), j = 1, ..., k s.t. Σ_{j=1}^{k} p_sj(n) = 1
       β_sg(n) = β_sg(n − 1)/γ
       Randomly select one path (other than g), recorded as g_s(n)
    }
    else {
          p_sg(n) = min(2p_sg(n − 1), 1)
          Normalize p_sj(n), j = 1, ..., k s.t. Σ_{j=1}^{k} p_sj(n) = 1
          β_sg(n) = β_sg(n − 1)α
          Randomly select one path, recorded as g_s(n)
    }
}
g = g_s(n); z = y_sg(n)
y_sg(n) = min(y_sg(n)(1 + β_sg(n)) + ε_s, m_s)
if (y_sg(n) == m_s) β_sg(n) = max((y_sg(n) − ε_s)/z − 1, 0)
if (Σ_{j=1}^{k} y_sj(n) > m_s) Normalize y_sj(n), j = 1, ..., k, j ≠ g s.t. Σ_{j=1}^{k} y_sj(n) = m_s
```

Fig. 2. Application-level multipath rate control: source s performs parameter-adjustment and rate-increment steps at the beginning of the nth control interval, $n \geq 1$, α and γ are constants, $\alpha > 1$, $\gamma > 1$, ϵ_s is a small positive constant.

multiplied by a constant $\alpha > 1$. Otherwise, the sending rate of this path is reduced to the original value (i.e., before the rate increment), the probability of choosing this path is halved, and the rate increment term associated with this path is divided by a constant $\gamma > 1$. In the rate-increment step, suppose source s selects path g in the nth control interval. Then the sending rate of path g is increased to the minimum of $y_{sg}(n)(1 + \beta_{sg}(n)) + \epsilon_s$ and the demand m_s, where $\epsilon_s > 0$ is a small constant. If the minimum is m_s, the corresponding rate increment term $\beta_{sg}(n)$ is adjusted accordingly to reflect the actual rate increment. In the above, the rate increment and adjustment of the rate increment terms are inspired by the Bertsekas' bold-step strategy used with the subgradient method [29]. We explore the choice of the constants (including α, γ, ϵ_s, $\beta_{sj}(0)$) in Section 5.

Our scheme runs in a distributed manner — each source independently adjusts its path rates based on localized information. It does not require explicit network knowledge (e.g., topology, available bandwidth) or any additional support from the network. The rate adjustment is essentially MIMD (Multiplicative Increment Multiplicative Decrement) when $\beta_{sj}(0) > 0$ and AIAD (Additive Increment Additive Decrement) when $\beta_{sj}(0) = 0$. Note that, for each source, our control algorithm does not lead to a throughput higher than that allowed by the underlying transport-level controller (e.g., TCP), and hence does not introduce further congestion into the network.

Optimality property. We next describe an optimality property of our rate controller when each source is allowed to use a single relay on each overlay path. This scenario is of special interest to us because routing in this type of overlay networks is very simple. Furthermore, recent studies have shown that using a single relay on overlay paths provides performance close to those using multiple relays [8,27,30]. The optimality property is stated in the following theorem; the proof follows from representing this problem as a network flow problem and can be found in [31].

Theorem 1. *For perfect congestion detection, infinitely small ϵ_s and $\beta_{sj}(0) = 0$, $\forall s \in S, j = 1, \ldots, k$, our application-level rate controller maximizes the aggregate sending rate of the sources when all sources have the same destination and each overlay path allows a single relay.*

An example. We now illustrate the rate adjustment by our controller using an example in Fig. 3. In the figure, all source demands are 6 Mbps; only the bandwidths from the relays to the receiver (each of 6 Mbps) constrain the sending rates of the sources. We represent the path rates for each source using a matrix. Initially, for sources s_1 and s_2, the sending rates via relays r_1 and r_2 are 3 and 2 Mbps, respectively. For source s_3, the sending rates via relays r_2 and r_3 are

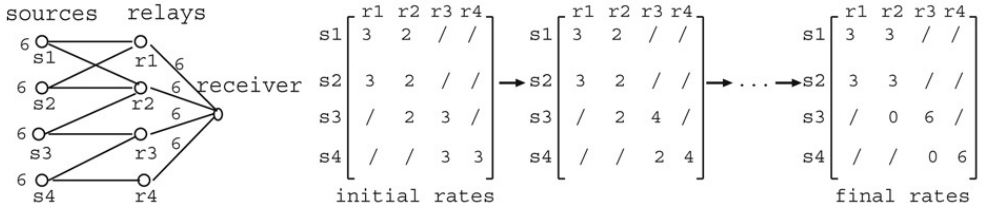


Fig. 3. An example illustrating the rate adjustment by our rate controller. In the rate adjustment sequence, $\epsilon_s = 1$ Mbps, $\beta_{sj}(0) = 0$.

2 and 3 Mbps, respectively. For source s_4, the sending rates via relays r_3 and r_4 are both 3 Mbps. When using our rate controller, s_4 gradually shifts its data from relay r_3 to r_4; correspondingly, s_3 shifts its data from relay r_2 to r_3. Eventually, all source demands are satisfied.

Initial path rates. Our rate-control algorithm can start from any valid rate allocation. A straightforward way is to assign initial path rates to 0. We next describe a *maxmin procedure* to set the initial path rates: each source, when having data to send, independently cycles through its overlay paths in a round-robin fashion and sends a unit of data (e.g., a fixed-size packet) on a path that it can send on. We refer to the above as a maxmin procedure, because it is similar to the maxmin flow-control algorithm [32]: cycling over the paths in a round-robin fashion and sending a data unit when possible is similar to increasing the path rates linearly (i.e., the "filling" procedure) in maxmin flow control. In Fig. 3, the initial path rates are obtained using the maxmin procedure.

Realization on top of TCP. Our application-level multipath rate controller can run on top of any transport-level rate controller. We now briefly describe how to realize this controller on top of TCP. A source establishes a TCP connection on each logical hop to the receiver. The TCP receiver of the first logical hop (i.e., a relay) forwards incoming data to the second hop, and so on. When a relay cannot forward data to its next hop, its receiving buffer will be full, hence slowing down the sending rate in the previous hop. Therefore, the throughput on a path is the minimum throughput over all logical hops on the path. The actual sending rate of source s, $x_{sj}(n)$, can be measured at its receiver and fed back to the source (e.g., using a separate TCP connection). Implementing the maxmin procedure on top of TCP is straightforward: a source cycles through its TCP sockets on the first logical hop in a round-robin fashion, finds a TCP socket that is writable, and writes a unit of data into that socket. We have implemented our rate controller (with the maxmin procedure) in a testbed [31].

4. Overlay path selection

Our approach towards path selection is as follows. In the first iteration, all sources select paths. In a later iteration, only unsatisfied sources (i.e., those whose current sending rates lie below their demands) reselect paths. The iteration continues until all source demands are satisfied (no source reselects path at that time). We next describe path selection in the first and later iterations.

Path selection in the first iteration. For a given rate controller, the goal of path selection is to maximize the aggregate sending rate of the sources. We prove that this problem is NP-hard even in an extremely simple setting, referred to as *single-receiver 2nd-hop-constrained setting*. In this setting, all sources have the same receiver. Furthermore, only a single relay is allowed on each overlay path and the 2nd hop (i.e., from the relays to the destination) constrains the source rates. For a source, selecting an overlay path in this setting is equivalent to selecting a relay, which is similar to load balancing (see, e.g., [33]). However, existing work on load balancing chooses a single relay for a source instead of distributing the load of the source onto multiple relays simultaneously. The complexity results on optimal path selection are summarized below; all proofs can be found in [28].

Theorem 2. *In a single-receiver 2nd-hop-constrained setting, for any given rate controller, the problem of optimal path selection is strongly NP-hard for $k \geq 1$. Even when all sources' demands are the same or when the bandwidths from the relays to the receiver are the same, the problem of optimal path selection is NP-hard; in particular, when $k \geq 3$, this problem is strongly NP-hard.*

Corollary 1. *For a given rate controller, the problem of optimal path selection is NP-hard.*

Due to the complexity of optimal path selection, we use two randomized heuristics to select paths motivated by results in [8]. In the first method, a source uniformly chooses k distinct paths, referred to as *uniform choice rule*. In

the second method, a source chooses a path with a probability proportional to the available bandwidth on that path, referred to as *proportional choice rule*. Note that the uniform choice rule requires no knowledge of the network while the proportional choice rule requires knowing the available bandwidth on each path.

Path (re)selection in later iterations. In later iterations, only unsatisfied sources reselect paths. We consider two rules to reselect paths: *k-reselection* and 1-*reselection*. In k-reselection, an unsatisfied source randomly reselects (using uniform or proportional choice rule) all of its k paths. In 1-reselection, an unsatisfied source only replaces the path with the minimum rate by randomly reselecting one path. Both k-reselection and 1-reselection can be easily implemented in a distributed and asynchronous manner. We prove the following convergence property for k-reselection; the proof is found in [28].

Theorem 3. *In a network, when all source demands are the same and there exists a path selection so that all source demands are satisfied for a given rate controller, k-reselection converges to such a path selection.*

Combining the above theorem and Theorem 1, we have the following result:

Corollary 2. *When all sources have the same destination and each overlay path allows a single relay, if there exists a path selection and rate allocation so that all source demands are satisfied, our path-selection and rate-control algorithms converge to a solution that finds them.*

5. Performance evaluation

In this section, we evaluate the performance of our distributed iterative heuristic (with path selection and rate control as two basic components) for multipath data transfer. Since empirical studies have shown that using a single relay on overlay paths provides performance close to those using multiple relays [8,27,30], we focus on settings with a single relay (i.e., two logical hops) from a source to a destination. Furthermore, all sources transmit to the same receiver and the second hops constrain the sending rate of sources (they are more likely to be shared by multiple sources and hence congested). In this setting, the problem of optimal path selection remains NP-hard (see Theorem 2). Furthermore, if there exists a path selection and rate allocation so that all source demands are satisfied, our approach converges to find them (see Corollary 2).

We consider 100 sources, i.e., $|S| = 100$. The source demands are *homogeneous* or *heterogeneous*. The number of relays equals the number of sources, i.e., $|R| = 100$. We index the relays in decreasing order of their bandwidths to the receiver. The bandwidth from the jth relay to the receiver is set in proportion to $1/j^b$, where $0 \leq b \leq 1$. We refer to b as the *skew factor*. When $b = 0$, all relays have the same bandwidth to the receiver. As b increases, the bandwidth distribution among the relays becomes more skewed. Note that even under homogeneous source demands or homogeneous relay bandwidths, the problem of optimal path selection remains NP-hard (Theorem 2). Let a_r represent relay r's bandwidth to the receiver. Let $f = \sum_{r\in R} a_r / \sum_{s\in S} m_s$, that is, f represents the ratio of network bandwidth over the aggregate source demands. We vary f from 0.6 to 2.2. Our performance metric is the aggregate source rate (i.e., the actual sending rate into the network) normalized by the aggregate source demands, i.e., $\sum_{s\in S} x_s / \sum_{s\in S} m_s$, referred to as *normalized aggregate source rate*.

We next progressively explore the components in our approach: first our application-level rate controller (Section 5.1), then the initial path selection and rate control (Section 5.2), finally the iterative path selection and rate control (Section 5.3). Note that each of the above components is interesting in its own right: rate control is important for scenarios where paths are fixed beforehand; initial path selection and rate control is important for scenarios where paths cannot be changed after the initial selection; iterative path selection and rate control is the full-fledged approach when allowing multiple path selections.

5.1. Evaluation of application-level rate controller

We now evaluate our distributed application-level rate controller when paths are fixed *a priori*. Our focus is on the impact of various parameters and the comparison of results from our controller with the optimal results (by solving optimization problem **P** using CPLEX [34]). We use both numerical study and *ns-2* simulation: the numerical study is more scalable while the simulation takes into account practical issues (e.g., network delay, packetized network flows, and bursty packets transmission).

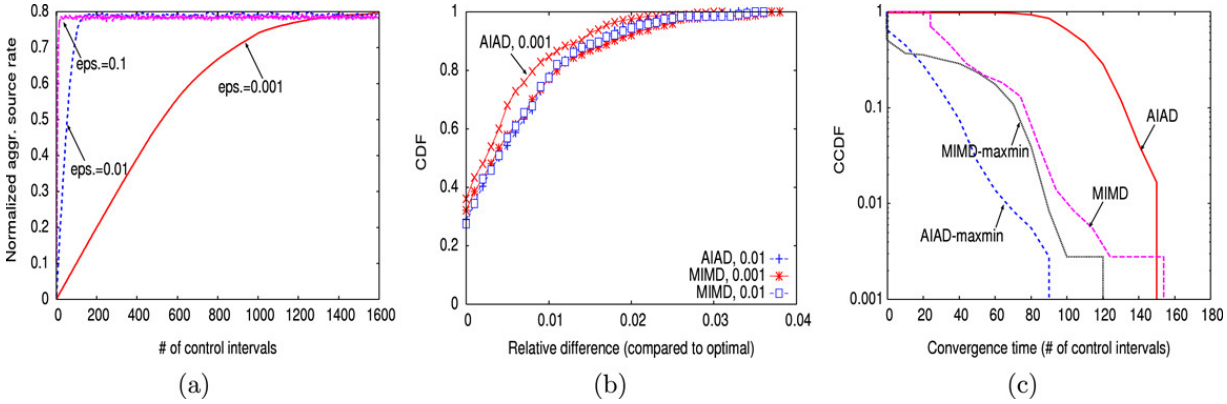


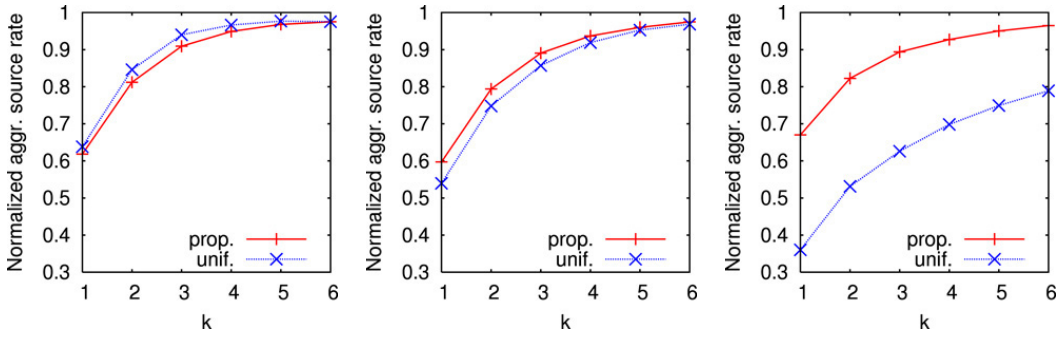
Fig. 4. Evaluation of our rate controller for homogeneous source demands and $k = 2$: (a) Results in a single run, using AIAD, $b = 0.5$, $f = 1$; (b) CDF of the relative difference of results from our controller and from CPLEX; (c) CCDF of the convergence time, $\epsilon = 0.01$.

Numerical study. When source demands are homogeneous, the demand of each source is either 1.2 or 120 Mbps, representing respectively a low and high demand. When source demands are heterogeneous, they are uniformly chosen from the discrete set {1.0, 10, 100} Mbps. For source s, the small increment, ϵ_s, is set to ϵ times the source demand, $\epsilon = 0.001$, 0.01 or 0.1. The constants, α and γ, are both set to 1.1 or set to 1.1 and 2.0 respectively. The initial rate increment term, $\beta_{sj}(0)$, is set to 0 or 0.1, corresponding to AIAD and MIMD respectively. For each source, the number of paths, k, is 2 or 3; the initial path rates are set to 0 or determined by the maxmin procedure (see Section 3); the initial path selection probability for a path is set to $1/k$. The constant to detect whether a rate increment succeeds, δ, is set to 0. Our results below are for homogeneous source demands ($m_s = 1.2$ Mbps, results under $m_s = 120$ Mbps are very similar), $k = 2$, $\alpha = 1.1$ and $\gamma = 2.0$ (this setting leads to less variance than $\alpha = \gamma = 1.1$). We observe similar trends under other settings ($k = 3$ or heterogeneous source demands).

We first look at the behavior of our rate controller in a single run. Fig. 4(a) plots the normalized aggregate source rate versus time (in terms of control intervals) when using AIAD for $m_s = 1.2$ Mbps, $b = 0.5$, $f = 1$, $k = 2$. The initial path rates are 0. We observe that the normalized aggregate source rate increases with time until reaching the steady state and a smaller ϵ leads to a slower convergence. When $\epsilon = 0.01$ or 0.001, the normalized aggregate source rate upon convergence is very close to the optimal value (i.e., 0.80) obtained by CPLEX. The slight difference might be because ϵ is not close to 0, as required by Theorem 1. When $\epsilon = 0.1$, the aggregate source rate is lower and exhibits a significantly larger variance (these are even more clearly observed when source demands are heterogeneous), indicating that $\epsilon = 0.1$ is too large for rate increment.

We now explore convergence values and convergence times of our rate controller (obtained as follows) more systematically. The convergence value is the average normalized aggregate source rate of the last 5000 control intervals (each run contains 10 000 control intervals). The convergence time is the time after which all values are within 98% of the convergence value. Fig. 4(b) plots the CDF of the relative difference of the normalized aggregate source rates from our controller and from CPLEX for $m_s = 1.2$ Mbps, $k = 2$, using AIAD or MIMD and $\epsilon = 0.01$ or 0.001. Each curve is from 360 runs ($b = 0$, 0.5 or 1.0, $f = 0.6$, 1.0, 1.6, or 2.0 and 30 uniformly random path selections for each combination of b and f values). The relative difference is less than 4% for all the settings, indicating that the results from our controller are very close to the optimal values. We observe that, although Theorem 1 requires ϵ to be close to 0, a relatively large ϵ ($\epsilon = 0.01$) leads to convergence values very close to the optimal values and much faster convergence times, and hence is preferable in practice. Fig. 4(c) plots the CCDFs (Complementary Cumulative Distribution Functions) of the convergence time for AIAD and MIMD when using the maxmin procedure to set the initial path rates (referred to as *AIAD-maxmin* and *MIMD-maxmin* respectively) and when the initial path rates are 0, $\epsilon = 0.01$. When the initial path rates are 0, MIMD converges much faster than AIAD. Using the maxmin procedure generally reduces the convergence time for AIAD and MIMD. Furthermore, AIAD-maxmin outperforms MIMD-maxmin and converges in only tens of control intervals.

Simulation results. We implemented our rate controller in *ns-2* simulator and evaluated the performance of our rate controller using simulation. In our simulation, each packet is 500 bytes. When source demands are homogeneous, the demand of each source is either 1.2 or 6 Mbps (larger demands take a long time in *ns-2*). When source demands are heterogeneous, they are uniformly chosen from {1.2, 3.6, 6} Mbps. Round-trip times are set to 40 ms when



(a) $b = 0$ (b) $b = 0.5$ (c) $b = 1.0$

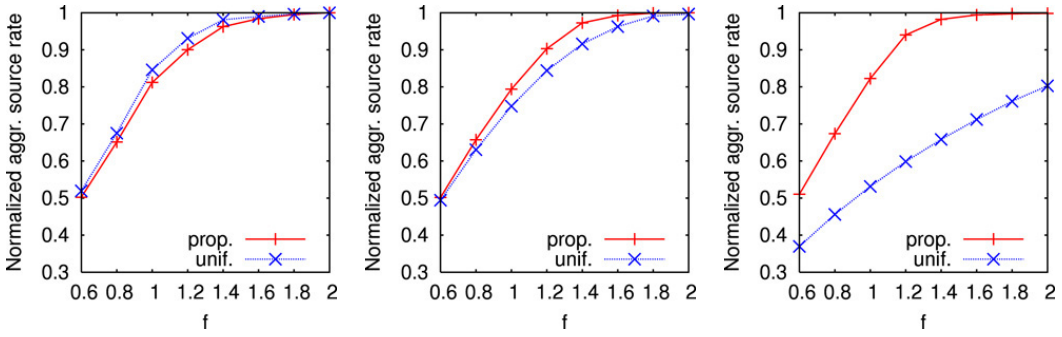
Fig. 5. Effect of k, homogeneous source demands, $f = 1$.

$m_s = 1.2$ Mbps and 20 ms otherwise (the smaller value is to ensure that the TCP throughput on a single path can reach 6 Mbps when not congested). The length of a control interval is 10 times the round-trip time. We use AIAD-maxmin with $\epsilon = 0.01$ for rate control. Each run contains 1000 control intervals. To account for network delays and measurement noises, the small constant to detect whether a rate increment succeeds or not, δ, is set to 0.03. The convergence time is the time point after which all values are within 95% of the convergence value (which is the average of the last 500 control intervals). We observe similar results as in the numerical study: the relative differences from our controller and CPLEX are less than 7% for all the cases; the convergence time is in tens of control intervals. *Summary*. To summarize, results from our numerical study and *ns-2* simulation are consistent, both demonstrating that AIAD-maxmin with $\epsilon = 0.01$ leads to a fast convergence (only in tens of control intervals) and a convergence value very close to the optimal value. Our evaluation in the rest of this section is through numerical study using AIAD-maxmin with $\epsilon = 0.01$ for rate control.

5.2. Evaluation of initial path selection and rate control

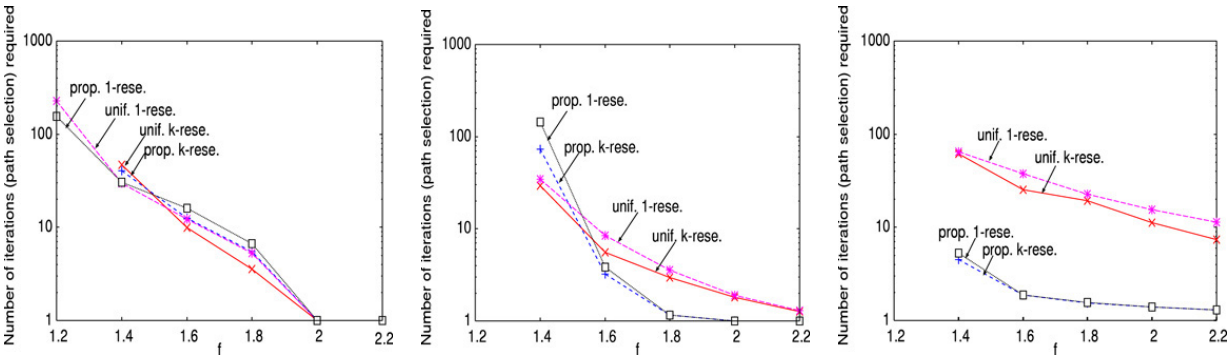
We now evaluate the performance of our initial path-selection algorithms (i.e., uniform or proportional choice rule) combined with our rate controller. Since the proportional choice rule requires knowing bandwidths on the overlay paths and existing bandwidth estimation techniques (e.g., [35–37]) have estimation errors, we first investigate its performance in the presence of bandwidth estimation errors. We assume that the relative estimation error is uniformly distributed in $[0, err]$, where err denotes the maximum relative estimation error. Our results demonstrate that the proportional choice rule is not sensitive to estimation errors: the performance with an estimation error up to 50% is very close to that without estimation error. For instance, the relative difference in the presence and absence of errors is in $[-5\%, 5\%]$ for homogeneous sources and $k = 2$ (figure omitted). In the rest of the paper, we assume $err = 0.5$ for proportional choice rule.

It is expected that for the same initial path selection rule, the normalized aggregate source rate increases with k (the number of paths selected by each source) and f (representing the amount of bandwidth in the network). We first explore the effect of k. Fig. 5 plots the normalized aggregate source rate versus k for homogeneous source demands, $f = 1$, and skew factors $b = 0$, 0.5 and 1. The 95% confidence intervals (from 30 runs) are tight and hence omitted. We observe a significant improvement when k increases from 1 to 2, which demonstrates the benefits of using multipath. On the other hand, there is a diminishing gain from increasing k on the aggregate source rate. This indicates that small values of k (i.e., 2 to 4) can realize most of the performance gains. We also observe that proportional choice rule leads to dramatic gains only under a highly skewed bandwidth distribution (i.e., $b = 1$). This indicates that, unless relay bandwidths are very skewed, it may not be worthwhile to estimate the relay bandwidths to use the proportional choice rule. The results under heterogeneous source demands are similar. We now explore the impact of f. Fig. 6 plots the normalized aggregate source rate versus f for homogeneous source demands and $k = 2$. The 95% confidence intervals (from 30 runs) are tight and hence omitted. We observe a diminishing gain on increasing f on performance: as f increases from 0.4 to 1.0, the performance improves dramatically and the improvement is less dramatic afterwards. Furthermore, the normalized aggregate source rate approaches 1 (i.e., all source demands are satisfied) when f approaches 2 in most cases.



(a) $b = 0$ (b) $b = 0.5$ (c) $b = 1.0$

Fig. 6. Effect of f, homogeneous source demands, $k = 2$.



(a) $b = 0$ (b) $b = 0.5$ (c) $b = 1.0$

Fig. 7. Number of iterations required to find an optimal path selection, homogeneous source demands, $k = 2$.

5.3. Evaluation of iterative path selection and rate control

We now examine the performance of our path reselection schemes, k-reselection and 1-reselection. A path selection under which all source demands are satisfied is referred to as an *optimal path selection*. We next present the number of iterations required to find an optimal path selection for homogeneous source demands and $k = 2$; the results under heterogeneous source demands exhibit similar trends.

We vary the ratio of the aggregate relay bandwidth over the aggregate source demands, f, from 1.2 to 2.2. For each value of f, we make 30 runs (with different initial path selection). In each run, we iteratively select paths until all source demands are satisfied or stop after 1000 iterations. Fig. 7 plots the number of iterations required to find an optimal path selection under k-reselection and 1-reselection using uniform or proportional choice rule. The confidence intervals are omitted from the figure for clarity (they are significant only when the average value is large). We observe that the number of iterations required to find an optimal path selection decreases when increasing f. Furthermore, typically only a few iterations are required when $f \geq 1.6$. This is in contrast to a baseline scheme where *each* source (even if its demand is satisfied) reselects paths until all source demands are satisfied. When using this baseline scheme, the source demands are not satisfied even after 1000 iterations when $f = 1.8$. The effectiveness of k-reselection and 1-reselection is also clearly shown when $b = 1$ (bandwidth highly skewed) and using uniform choice rule: k-reselection and 1-reselection quickly improve the normalized aggregate source rate (the average initial value is only 0.71, see Fig. 6), and find an optimal path selection in an average of 24.3 and 36.8 iterations respectively.

6. Conclusions and future work

In this paper, we have considered multipath data transfer when each source is allowed to send data on k ($k \geq 1$) overlay paths to its destination. We proposed a distributed iterative heuristic for this problem. Each iteration contains two components — path selection and rate control. The iteration continues until all source demands are satisfied. For rate control, we developed a distributed application-level multipath rate controller that runs on top of TCP. For path

selection, we proved that, for any given rate controller, the problem of optimal overlay path selection is NP-hard and proposed randomized path-selection algorithms. Our performance evaluation demonstrates that our iterative heuristic performs very well in a wide range of settings. Furthermore, a small number of paths, 2–4, and a small amount of extra bandwidth in the network are sufficient to realize most of the performance gains. As future work, we plan to evaluate the performance of our iterative scheme in more general settings (e.g., when there are multiple destinations).

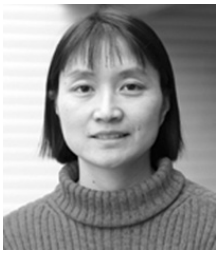
Acknowledgments

We thank Peter Key for many helpful comments. We also thank Micah Adler, Jun-hong Cui, Junning Liu, Yoo-ah Kim and Chun Zhang for helpful discussions. This research was supported in part by the ERC Program of the National Science Foundation under Award Number EEC-0313747, NSF RI EIA-0080119, NSF ANI-0240487, NSF ANI-0325868 and UConn Large Grant FRS 449251.

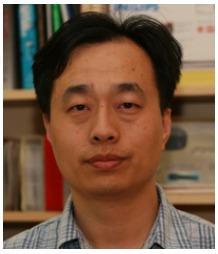
References

[1] Engineering Research Center for Collaborative Adaptive Sensing of the Atmosphere. http://www.casa.umass.edu.
[2] J. Gray, A.S. Szalay, The world-wide telescope, an archetype for online science, Tech. Rep. MSR-TR-2002-75, Microsoft Research, June 2002.
[3] A. Akella, S. Seshan, A. Shaikh, An empirical evaluation of wide-area internet bottlenecks, in: IMC, Miami, Florida, 2003.
[4] A. Akella, B. Maggs, S. Seshan, A. Shaikh, R. Sitaraman, A measurement-based analysis of multihoming, in: Proc. ACM SIGCOMM, August 2003.
[5] A. Akella, J. Pang, B. Maggs, S. Seshan, A. Shaikh, A comparison of overlay routing and multihoming route control, in: Proc. ACM SIGCOMM, 2004.
[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, The MIT Press, 1990.
[7] S. Floyd, K. Fall, Promoting the use of end-to-end congestion control in the Internet, IEEE/ACM Transactions on Networking (1999).
[8] K.P. Gummadi, H.V. Madhyastha, S.D. Gribble, H.M. Levy, D.J. Wetherall, Improving the reliability of internet paths with one-hop source routing, in: Proceedings of the 6th Symposium on Operating Systems Design and Implementation, San Francisco, CA, December 2004.
[9] J. Chen, P. Druschel, D. Subramanian, An efficient multipath forwarding method, in: Proc. IEEE INFOCOM, 1998.
[10] S. Vutukury, J. Garcia-Luna-Aceves, MPATH: A loop-free multipath routing algorithm, Elsevier Journal of Microprocessors and Microsystems (2000) 319–327.
[11] W.T. Zaumen, J.J. Garcia-Luna-Aceves, Loop-free multipath routing using generalized diffusing computations, in: INFOCOM, vol. 3, 1998, pp. 1408–1417.
[12] B. Cheng, C. Chou, L. Golubchik, S. Khuller, Y.-C. Wan, Large scale data collection: A coordinated approach, in: Proc. IEEE INFOCOM, San Francisco, CA, March 2003.
[13] S. Ganguly, A. Saxena, S. Bhatnagar, S. Banerjee, R. Izmailov, Fast replication in content distribution overlays, in: Proc. IEEE INFOCOM, Miami, FL, March 2005.
[14] F. Kelly, A. Maulloo, D. Tan, Rate control in communication networks: Shadow prices, proportional fairness and stability, Journal of the Operational Research Society 49 (1998) 237–252.
[15] W.-H. Wang, M. Palaniswami, S.H. Low, Optimal flow control and routing in multi-path networks, Performance Evaluation 52 (2003) 119–132.
[16] X. Lin, N.B. Shroff, The multi-path utility maximization problem, in: 41st Annual Allerton Conference on Communication, Control, and Computing, Monticello, IL, October 2003.
[17] S.H. Low, Optimization flow control with on-line measurement, in: Proceedings of the 16th International Teletraffic Congress, Edinburgh, UK, June 1999.
[18] B.A. Movsichoff, C.M.L.H. Che, Decentralized optimal traffic engineering in the Internet, IEEE Journal on Selected Areas in Communications (2005).
[19] K. Kar, S. Sarkar, L. Tassiulas, Optimization based rate control for multipath sessions, in: Proceedings of Seventeenth International Teletraffic Congress, ITC, Salvador da Bahia, Brazil, December 2001.
[20] R. Srikant, The Mathematics of Internet Congestion Control, Springer-Verlag, 2004.
[21] H. Han, S. Shakkottai, C. Hollot, R. Srikant, D. Towsley, Overlay TCP for multi-path routing and congestion control, in: IMA Workshop on Measurements and Modeling of the Internet, January 2004.
[22] P. Key, L. Massoulié, D. Towsley, Combining multipath routing and congestion control for robustness, in: Conference on Information Sciences and Systems, March 2006.
[23] B. Allcock, J. Bester, J. Bresnahan, A.L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, S. Tuecke, Data management and transfer in high-performance computational grid environments, Parallel Computing 28 (5) (2002) 749–771.
[24] R. Sivakumar, S. Bailey, R. Grossman, Psockets: The case for application-level network striping for data intensive applications using high speed wide area networks, in: ACM/IEEE Conference on supercomputing, Dallas, TX, March 2000.
[25] P. Rodriguez, E.W. Biersack, Dynamic parallel access to replicated content in the internet, IEEE/ACM Transactions on Networking (August) (2002).

[26] H.-Y. Hsieh, R. Sivakumar, A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts, in: MOBICOM, Atlanta, GA, September 2002.
[27] J. Han, D. Watson, F. Jahanian, Topology aware overlay networks, in: Proc. IEEE INFOCOM, March 2005.
[28] B. Wang, J. Kurose, D. Towsley, W. Wei, Multipath overlay data transfer, Tech. Rep. 05-45, Department of Computer Science, University of Massachusetts, Amherst, 2005.
[29] D.P. Bertsekas, Nonlinear Programming, 2nd ed., Athena Scientific, 1999.
[30] H. Pucha, Y.C. Hu, Overlay TCP: Ending end-to-end transport for higher throughput, in: Proc. ACM SIGCOMM, August 2005.
[31] B. Wang, W. Wei, J. Kurose, D. Towsley, Z. Guo, Z. Peng, K.R. Pattipati, Application-layer rate control for efficient multipath data transfer via TCP, Tech. Rep. BECAT/CSE-TR-06-17, Computer Science and Engineering Department, University of Connecticut, 2006.
[32] D.P. Bersekas, R. Gallager, Data Networks, 2nd ed., Prentice Hall, 1992.
[33] M. Mitzenmacher, The power of two choices in randomized load balancing, IEEE Transactions on Parallel and Distributed Systems 12 (October) (2001) 1094–1104.
[34] http://www.ilog.com/products/cplex/.
[35] V. Ribeiro, M. Coates, R. Riedi, S. Sarvotham, R.G. Baraniuk, Multifractal cross-traffic estimation, in: Proc. ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management, September 2000.
[36] B. Melander, M. Bjorkman, P. Gunningberg, A new end-to-end probing and analysis method for estimating bandwidth bottlenecks, in: Proc. IEEE GLOBECOM, November 2000.
[37] M. Jain, C. Dovrolis, End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput, in: Proc. ACM SIGCOMM, 2002.



Bing Wang received her B.S. degree in Computer Science from the Nanjing University of Science & Technology, China in 1994, and M.S. degree in Computer Engineering from the Institute of Computing Technology, Chinese Academy of Sciences in 1997. She then received M.S. degrees in Computer Science and Applied Mathematics, and a Ph.D. in Computer Science from the University of Massachusetts, Amherst in 2000, 2004, and 2005 respectively. Afterwards, she joined the Computer Science & Engineering Department at the University of Connecticut as an assistant professor. Her research interests are in computer networks, multimedia, and distributed systems. More specifically, she is interested in topics on Internet technologies and applications, wireless and sensor networks, overlay networks, content distribution, network management and measurement, network modeling and performance evaluation. She is a member of ACM, ACM SIGCOMM, IEEE, IEEE Computer Society, and IEEE Communications Society.



Wei Wei is currently a senior research engineer at the United Technologies Research Center. He received his B.S. degree in Applied Mathematics from Beijing University, China in 1992, and M.S. degree in Statistics from Texas A & M University in 2000. He then received M.S. degrees in Computer Science and Applied Mathematics, and a Ph.D. in Computer Science from the University of Massachusetts, Amherst in 2004, 2004, and 2006 respectively. His research interests are in the areas of computer networks and distributed embedded systems.

Jim Kurose received his Ph.D. degree in Computer Science from the Columbia University, and is currently Distinguished University Professor in the Department of Computer Science at the University of Massachusetts. Professor Kurose has been a Visiting Scientist at IBM Research, INRIA, Institut EURECOM, the University of Paris, LIP6, and Thomson Research Labs.

His research interests include network protocols and architecture, network measurement, sensor networks, multimedia communication, and modeling and performance evaluation. Dr. Kurose has served as Editor-in-Chief of the IEEE Transactions on Communications and was the founding Editor-in-Chief of the IEEE/ACM Transactions on Networking. He has been active in the program committees for IEEE Infocom, ACM SIGCOMM, and ACM SIGMETRICS for a number of years, and has served as Technical Program Co-Chair for these conferences.

He has received a number of awards for his educational activities, including the IEEE Taylor Booth Education Medal. With Keith Ross, he is the co-author of the textbook, “Computer Networking, a top down approach (4th ed.)” published by Addison-Wesley Longman.

Don Towsley holds a B.A. in Physics (1971) and a Ph.D. in Computer Science (1975) from the University of Texas. From 1976 to 1985 he was a member of the faculty of the Department of Electrical and Computer Engineering at the University of Massachusetts, Amherst. He is currently a Distinguished Professor at the University of Massachusetts in the Department of Computer Science. He has held visiting positions at the IBM T.J. Watson Research Center, Yorktown Heights, NY; Laboratoire MASI, Paris, France; INRIA, Sophia-Antipolis, France; AT&T Labs-Research, Florham Park, NJ; and Microsoft Research Lab, Cambridge, UK. His research interests include networks and performance evaluation.

He currently serves as Editor-in-Chief of IEEE/ACM Transactions on Networking and is on the editorial boards of the Journal of the ACM and the IEEE Journal on Selected Areas in Communications, and has previously served on numerous other editorial boards. He was Program Co-chair of the joint ACM SIGMETRICS and PERFORMANCE ’92 conference and the Performance 2002 conference. He is a member of ACM and ORSA, and Chair of IFIP Working Group 7.3.

He has received the 2007 IEEE Koji Kobayashi Award, the 2007 ACM SIGMETRICS Achievement Award, the 1998 IEEE Communications Society William Bennett Best Paper Award, and numerous best conference/workshop paper awards. Last, he has been elected Fellow of both the ACM and IEEE.

Krishna R. Pattipati is a Professor of Electrical and Computer Engineering at the University of Connecticut, Storrs, CT, USA. He has published over 330 articles, primarily in the application of systems theory and optimization techniques to large-scale systems. Prof. Pattipati received the Centennial Key to the Future award in 1984 from the IEEE Systems, Man and Cybernetics (SMC) Society, and was elected a Fellow of the IEEE in 1995 for his contributions to discrete-optimization algorithms for large-scale systems and team decision-making. He received the Andrew P. Sage award for the Best SMC Transactions Paper for 1999, Barry Carlton award for the Best AES Transactions Paper for 2000, the 2002 NASA Space Act Award, the 2003 AAUP Research Excellence Award and the 2005 School of Engineering Teaching Excellence Award at the University of Connecticut. He also won the best technical paper awards at the 1985, 1990, 1994, 2002, 2004 and 2005 IEEE AUTOTEST Conferences, and at the 1997 and 2004 Command and Control Conferences. Prof. Pattipati served as Editor-in-Chief of the IEEE Transactions on SMC-Cybernetics (Part B) during 1998-2001.

Zheng Guo received his bachelor degree in Electronic Engineering from the University of Science and Technology of China (USTC) in 2005. He is currently a Ph.D. student in the Computer Science & Engineering Department of the University of Connecticut, Storrs. His research interests are broadly in computer networks. He is currently conducting research on network coding and delay/disruption tolerant network.

Zheng Peng received his B.S. degree in both Control Science and Engineering Department and Computer Science Department from the Zhejiang University, Hangzhou, China in 2002. He received a M.S. degree in Computer Science from the University of Electrical Science and Technology of China in 2005. He is currently pursuing his Ph.D and working as a research assistant at the Underwater Sensor Network Lab, University of Connecticut, Storrs. His main research interests are in wireless sensor network, underwater acoustic network, sensor node architecture and design and underwater testbed.