Distributed Application-layer Rate Control for Efficient Multipath Data Transfer via TCP

Bing Wang*, Wei Wei†, Jim Kurose†, Don Towsley†, Zheng Guo*, Zheng Peng*, Krishna R. Pattipati‡
*Computer Science & Engineering Department,
University of Connecticut, Storrs, CT 06269
†Department of Computer Science
University of Massachusetts, Amherst, MA 01003
‡Electrical & Computer Engineering Department,
University of Connecticut, Storrs, CT 06269

UCONN CSE Technical Report BECAT/CSE-TR-06-17

***Abstract*— For applications involving data transmission from multiple sources, an important problem is: when the sources use multiple paths, how to maximize the aggregate sending rate of the sources using application-layer techniques via TCP? We develop an application-level distributed rate controller to solve this problem. Our controller utilizes the bandwidth probing mechanisms embedded in TCP and does not require explicit network knowledge (e.g., topology, available bandwidth). We theoretically prove the convergence of our algorithm in certain settings. Furthermore, using a combination of simulation and testbed experiments, we demonstrate that our algorithm provides efficient multipath data transfer and is easy to deploy.**

I. Introduction

A wide range of applications require data transmission from geographically distributed sources to one or multiple destinations using the Internet. For instance, in the Engineering Research Center (ERC) 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 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 [2]). Scientists may need to retrieve and integrate data from archives at several locations for temporal and multi-spectra studies using the Internet (e.g., via SkyServer [3]). 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 multiple sources to one or multiple destinations. 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 [4]. This is clearly true in CASA: the sending rates of the radar nodes are constrained by low-bandwidth links inside the state-wide public network. When the bandwidth constraints are inside the network, using multiple paths (e.g., through multihoming or an overlay network) between a source and destination can provide a much higher throughput [5], [6]. The problem we address is: *when the sources use multiple paths, how to maximize the aggregate sending rate of the sources?* More specifically, the problem is as follows. Consider a set of sources with their corresponding destinations. Each source is allowed to spread data on k ($k \geq 2$) given network paths. We restrict the source to use no more than k paths since data splitting involves overheads (e.g., meta data are required in order to reassemble data at the destination). The problem we address is how to control the sending rate on each path in order to maximize the aggregate sending rate of the sources.

In this paper, we use *application-layer* techniques running on top of TCP to solve the above problem. We take this approach due to several reasons. First, these applications require reliable data transfer which makes TCP a natural choice. Second, since TCP is the predominant transport protocol in the current Internet, application-layer approaches via TCP are

easy to deploy. Furthermore, all applications in the Internet are expected to be TCP friendly [7] and using TCP is by definition TCP-friendly. Our main contributions are:

- We develop a distributed algorithm for application-layer multipath data transfer. This algorithm utilizes the bandwidth probing mechanisms embedded in TCP and does not require explicit network knowledge (e.g., topology, available bandwidth).
- We analyze the performance of our algorithm in scenarios where multiple paths between a source and destination are formed using an overlay network, which has been shown to be an effective architecture for throughput improvement [6]. We prove that rate allocation under our controller converges to maximize the aggregate sending rate of the sources in settings with two logical-hops and a single destination.
- Using a combination of simulation and testbed experiments, we demonstrate that our scheme provides efficient multipath data transfer and is easy to deploy.

As related work, the studies of [8], [9], [10] 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. Hence, our approach is readily deployable in the current Internet. The studies of [11] and [12] 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. A number of studies [13], [14], [15], [16], [17], [18], [19], [20] develop multipath rate controllers based on an optimization framework [21], [13]. These algorithms require congestion price feedback from the network and are difficult to realize in practice. Our emphasis is on efficient application-level approaches that are easy to implement.

The rest of this paper is organized as follows. Section II presents the problem setting. Section III presents our application-level rate control algorithm. Section IV presents a performance evaluation using *ns-2* simulator. Section V describes experimental results of our multipath rate controllers in a testbed. Finally, Section VI concludes this paper and describes future work.

II. Problem setting

In this section, we formally describe the problem setting. Consider a set of sources S, each associated with a destination. Let D denote the set of destinations. Each source is given k $(k \geq 2)$ network paths and spreads its data over the paths.

We denote by *path rate* the rate at which a source sends data over a path. The sum of the path rates associated with a source is the *source rate*. For ease of exposition, we index a source's paths as paths 1 to k. For source s, let x_{sj} denote its path rate on the j-th path and x_s denote its source rate, $x_s \geq 0, x_{sj} \geq 0$. Then, $x_s = \sum_{j=1}^{k} x_{sj}$. Let m_s be the maximum source rate of source s, referred to as the *demand* of the source. Then $x_s \leq m_s$. This maximum source rate may come from the bandwidth limit of the source or the data generation rate at the source.

For ease of exposition, we only consider sources using multiple paths; including sources using a single path in the problem formulation is straightforward [22]. 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 j-th path of source s. The path-rate control in the network can be stated as an optimization problem $\mathbf{P}$:

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

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

$$0 \leq x_s \leq m_s,\ s \in S \tag{3}$$

$$\sum_{s,j:l \in L_{sj}} x_{sj} \leq c_l,\ \forall l \in L \tag{4}$$

where (4) describes the link capacity constraints.

Note that the above source rate, x_s, and path rate, x_{sj}, refer to the actual sending rates that source s sends into the network. It is important to differentiate them from the sending rates that a source sets at the application-level. Let y_{sj} denote the sending rate that source s sets at the application-level on path j, referred to as *application-level path rate*. Then $x_{sj} \leq y_{sj}$ since the actual sending rate into the network is fundamentally limited by the underlying transport protocol (e.g., TCP). When developing an application-level rate contoller, we only have control over y_{sj} and our goal is to maximize $\sum_{s \in S} \sum_{j=1}^{k} x_{sj}$.

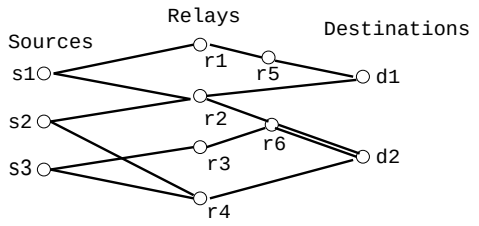

Fig. 1. Illustration of an overlay network. In this example, $k = 2$.

The multiple paths from a source to a destination can be formed using multihoming or an overlay network. Our performance study in this paper focuses on the latter scenario, which can effectively improve throughput [6]. More specifically, the overlay network we consider in this paper is formed by the set of sources S, the set of destinations D, and a set of relays R. A source selects k $(k \geq 2)$ overlay paths (i.e., network paths via one or multiple relays) and spreads its data over the overlay paths, as illustrated in Fig. 1. The sources and destinations have high access bandwidth (e.g., through well-connected access networks or multihoming [23]). The relays are placed (e.g., using techniques in [24]) such that multiple overlay paths do not share performance bottlenecks.

Overlay networks where each overlay path contains a single relay are of special interest to us. This is 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 [25], [24], [26]. Henceforth, we refer to this type of overlay network as *two-logical-hop overlay network*. Our performance evaluation focus on this type of overlay network (see Section IV).

III. Application-level Multipath Rate Control

We now describe our application-level multipath rate control algorithm. A key difference between our application-level approach and a transport-level approach (e.g., by modifying TCP directly) is: the sending rate that a source sets at the application level may be higher than that actually going into the network (since the actual sending rate is fundamentally limited by the underlying transport protocols). Next, we first describe our algorithm, and then describe a convergence property of our algorithm. At the end, we briefly describe how to realize our algorithm using TCP.

$p_{sj}(n) = p_{sj}(n-1)$, $j = 1, \ldots, k$
$y_{sj}(n) = y_{sj}(n-1)$, $j = 1, \ldots, k$
$\beta_{sj}(n) = \beta_{sj}(n-1)$, $j = 1, \ldots, k$
$g = g_s(n-1)$
if $(x_{sg}(n-1)/y_{sg}(n-1) < 1 - \delta)$ {
  $y_{sg}(n) = (y_{sg}(n-1) - \epsilon)/(1 + \beta_{sg}(n-1))$
  $p_{sg}(n) = p_{sg}(n-1)/2$
  Normalize $p_{sj}(n)$, $j = 1, \ldots, k$ s.t. $\sum_{j=1}^{k} p_{sj}(n) = 1$
  $\beta_{sg}(n) = \beta_{sg}(n-1)/\gamma$
  Randomly select one path (other than g), recorded as $g_s(n)$
}
else {
  $p_{sg}(n) = \min(2p_{sg}(n-1), 1)$
  Normalize $p_{sj}(n)$, $j = 1, \ldots, k$ s.t. $\sum_{j=1}^{k} p_{sj}(n) = 1$
  $\beta_{sg}(n) = \beta_{sg}(n-1)\alpha$
  Randomly select one path, recorded as $g_s(n)$
}
$g = g_s(n)$
$z = y_{sg}(n)$
$y_{sg}(n) = \min(y_{sg}(n)(1 + \beta_{sg}(n)) + \epsilon, m_s)$
if $(y_{sg}(n) == m_s)$ {
  $\beta_{sg}(n) = \max((y_{sg}(n) - \epsilon)/z - 1, 0)$
}
if $(\sum_{j=1}^{k} y_{sj}(n) > m_s)$ {
  Normalize $y_{sj}(n)$, $j = 1, \ldots, k, j \neq h$
  s.t. $\sum_{j=1}^{k} y_{sj}(n) = m_s$
}

Fig. 2. Application-level multipath rate control: source s determines its application-level path rates in the n-th control interval, $s \in S$, $\beta_{sj}(n) \geq 0, \alpha > 1, \gamma > 1$.

A. Application-level control algorithm

The basic idea of our algorithm is: based on an initial valid rate allocation (i.e., satisfying the link capacity constraint (4)), each source independently probes for paths with spare bandwidths through the bandwidth probing mechanisms embedded in TCP and increases its sending rates on those paths.

We now detail our algorithm (as shown in Fig. 2). Each source divides time into control intervals (the lengths of the control interval for different sources need not to be the same). For a source, since the sending rates of the multiple paths are correlated (the sum not exceeding the demand), in each control interval, the source probes network bandwidth by randomly selecting one path and increasing its path rate by a certain amount. In the n-th control interval, let $p_{sj}(n)$ represent the probability that source s chooses to probe path j, and let $g_s(n)$ denote the path that source s selects for bandwidth probing. Let $y_{sj}(n)$ denote the application-level path rate that source s sets on path j, and let $x_{sj}(n)$ denote the actual sending

rate that source s sends into the network on path j. Note that $x_{sj}(n) \leq y_{sj}(n)$. Our goal is to maximize the sum of the actual sending rates of the sources through controlling the application-level path rates. Initially, $y_{sj}(0)$ and $p_{sj}(0)$ can be set to any valid values. Furthermore, source s is associated with a *rate increment term* on path j in the n-th control interval, denoted as $\beta_{sj}(n)$, $\beta_{sj}(n) \geq 0$.

We next describe our rate adjustment algorithm, inspired by the Bertsekas' bold step strategy used with the subgradient method [27]. At the beginning of a control interval, a source performs two steps to adjust its application-level path rates, path selection probabilities and rate increment terms (if a quantity is adjusted in neither step, it is kept to be the same as that in the previous interval.). In the first step, the source adjusts the above quantities based on whether the rate increment on the selected path in the previous control interval is successful or not (to be defined shortly). In the second step, the source randomly selects a path and increases the sending rate on that path.

The first step is detailed as follows. We first define how to determine whether a rate increment is successful or not. Suppose source s chooses path g in the n-th control interval. Then we say that the rate increment in the n-th control interval is successful iff $x_{sg}(n)/y_{sg}(n) \geq 1 - \delta$. That is, increasing the application-level sending rate to a value that can be achieved by the network is considered a success and vice versa. Here δ is a small positive constant, chosen to accommodate measurement noises and network delay. If the rate increment on a path is not successful, the sending rate of this path is reduced to the original value (i.e., before the rate increment), the probability to choose this path is halved, and the rate increment term associated with this path is divided by a constant $\gamma > 1$. Otherwise, the probability to choose this path is doubled and the rate increment term is multiplied by a constant $\alpha > 1$. Intuitively, we increase the rate-increment speed for a path after a success and decrease the speed after a failure. This adaptive increment is important for fast convergence as to be demonstrated in Section IV. In principle, we can adjust the probability to choose a path in a similar manner as that for the rate adjustment. However, we find that the above simple probability adjustment works well (see Section IV).

We now describe the second step in detail. Suppose source s chooses path g in the n-th control interval. Then the sending rate of this path is increased to the minimum of $y_{sg}(n)(1 + \beta_{sg}(n)) + \epsilon$ and the demand m_s, where $\epsilon > 0$ 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 compared to the rate in the previous control interval.

The detailed algorithm is depicted in Fig. 2. The normalization of the path rates in the algorithm is to ensure that the sum of the path rates not exceeding the demand of the source. Similarly, the normalization of the path selection probabilities is to ensure that the sum of the probabilities is 1. We explore the choice of the parameters (including α, γ, $\beta_{sj}(0)$, $s \in S$, $j = 1, \ldots, k$) in Section IV.

Our scheme runs in a distributed manner — each source independently adjusts the 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. Note that our algorithm essentially uses MIMD (Multiplicative Increment Multiplicative Decrement) rate adjustment when $\beta_{sj}(0) > 0$ and AIAD (Additive Increment Additive Decrement) when $\beta_{sj}(0) = 0$. However, even under the more aggressive MIMD rate adjustment, for each source, our control algorithm does not lead to a throughput higher than that allowed by the underneath transport-level controller (e.g., TCP) on a path, and hence does not introduce further congestion into the network.

B. Convergence properties

As mentioned in Section II, we are especially interested in two-logical-hop overlay networks since recent findings have demonstrated the benefits of using such overlay networks [25], [24], [26]. We prove that our scheme converges to maximize the aggregate source rate when all sources have the same destination in two-logical-hop overlay networks, as stated in the following theorem. The proof is found in the Appendix.

Theorem 1: When assuming perfect congestion detection, our application-level rate controller converges to maximize the aggregate source rate when all sources have the same destination and each overlay path allows a single relay when $\beta_{sj}(0) = 0$, $\forall s \in S, j = 1, \ldots, k$.

The above convergence result is for $\beta_{sj}(0) = 0$, under which the rate increment/decrement is simply by

adding/substracting the small constant, ϵ. We have not been able to prove that the algorithm converges when $\beta_{sj}(0) > 0$. However, simulations results in Section IV demonstrate that our algorithm converges much faster when $\beta_{sj}(0) > 0$ than when $\beta_{sj}(0) = 0$. Indeed, the theoretical convergence property of the Bersekas' bold step strategy, although used extensively in a wide range of applications (e.g., scheduling, multi-object tracking), are not well understood [27].

C. Realization on top of TCP

The above application-level multipath rate control algorithm can run on top of any transport-level rate controllers. We now briefly describe how to realize this algorithm on top of TCP. When using TCP, a source establishes a TCP connection to the receiver on each path. When there are multiple logical hops on a path (e.g., in an overlay network), a TCP connection is established on each logical hop. The TCP receiver of one logical hop is the TCP sender of its next logical hop; when one logical hop is saturated, it back-pressures its previous hop (implicitly through TCP) such that the throughput on a path is the minimum throughput over all logical hops on the path. The actual sending rate of the source, $x_{sj}(n)$, can be measured at receiver and fed back to the sender (e.g., using a separate TCP connection). We have implemented our algorithm in both *ns-2* simulator and our testbed (see Sections IV and V). More implementation details are discussed in Section V.

IV. Performance Evaluation

We now evaluate the performance of our application-level rate controller through simulation using the *ns-2* simulator. Our evaluation is in an overlay network with a single receiver (i.e., all sources transmit to the same receiver). Furthermore, there are two logical-hops (i.e., a single relay) from a source to a destination. The first hop is from a source to a relay; the second hop is from a relay to a receiver. We assume that the second hops are congested (they are more likely to be shared by multiple sources and hence congested). Each source is given k overlay paths by randomly selecting k relays. Our performance metric is the aggregate source rate 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 stress that the aggregate source rate is the effective sending rate into the network (not that set at the application-level).

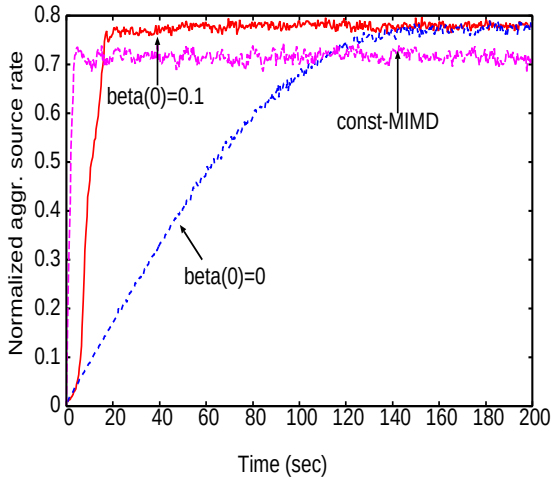

Fig. 3. Impact of the initial rate increment terms and comparison of our scheme with a simple rate adjustment scheme.

We now describe our settings in more detail. The number of paths for each source, k, is 2, 3 or 4. We index the relays in decreasing order of their bandwidths to the receiver. The bandwidth from the j-th relay to the receiver is set to be proportional to $1/j^b$, where $0 \le b \le 1$. 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. 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 3. We set $|S| = |R| = 100$ and $m_s = 1.2$ Mbps or 6 Mbps (higher values of demands lead to very long running time in *ns-2*). Each packet is 500 bytes. The the round-trip propagation delay on each logical hop is set to 20 ms when $m_s = 1.2$ Mbps and 5 ms when $m_s = 6$ Mbps (the shorter value is to ensure that the TCP throughput on one path can reach 6 Mbps).

In our rate adjustment, the length of the control interval for a source is 0.4 second. The small constant value, ϵ is set to 0.5 Kbps, α and γ are both set to 1.1 or 1.1 and 2.0 respectively. The threshold to detect whether a rate increment succeeds or not, δ, is set to 0.03. We set the initial increment term $\beta_{sj}(0)$ to 0.1 or 0; all the paths use the same value. The initial rates on all paths for a source are set to 0. For each source, the initial path selection probability is set to $1/k$.

We first look at the impact of the initial value of the rate increment term, $\beta_{sj}(0)$. Recall that when $\beta_{sj}(0) > 0$, our rate adjustment is essentially MIMD with adaptive rate

k	$b=0$				$b=0.5$				$b=1$			
	$f=0.6$	$f=1.0$	$f=1.6$	$f=3.0$	$f=0.6$	$f=1.0$	$f=1.6$	$f=3.0$	$f=0.6$	$f=1.0$	$f=1.6$	$f=3.0$
2	20	60	80	20	20	20	60	20	20	20	20	20
3	20	60	80	20	40	50	80	20	50	50	50	30
4	20	60	80	20	50	70	90	20	60	60	100	60

TABLE I

CONVERGENCE TIME (IN SECONDS) UNDER DIFFERENT SETTINGS, $m_s = 1.2$ MBPS, $\alpha = 1.1$, $\gamma = 2.0$.

increment terms; when $\beta_{sj}(0) = 0$, the rate adjustment is AIAD by simply adding or substracting the small constant, ϵ. Fig. 3 plots the normalized aggregate source rate using $\beta_{sj}(0) = 0.1$ and $\beta_{sj}(0) = 0$, where $k = 2$, $m_s = 1.2$ Mbps, $\alpha = 1.1$ and $\gamma = 2.0$ (the results under $\alpha = \gamma = 1.1$ are similar). We have proved that the rate adjustment under $\beta_{sj}(0) = 0$ converges to maximize the aggregate source rate. This is confirmed by the simulation results: we observe that the normalized aggregate source rate converges to a value close to 0.8, the optimal value obtained from *cplex* [28]. The slight difference between the simulation result and the optimal value maybe due to packetized network flows, network delays, and bursty packet transmission. When $\beta_{sj}(0) = 0.1$, we observe that the normalized aggregate source rate also converges to a value close to 0.8. Furthermore, the convergence rate under $\beta_{sj}(0) = 0.1$ is much faster (almost seven times faster) than that under $\beta_{sj}(0) = 0$.

In Fig. 3, we also plot the result when $\beta_{sj}(n) \equiv 1$, which leads to an MIMD rate adjustment with a constant multiplicative increment/decrement term of 2. We observe that this type of rate adjustment leads to more fluctuations and furthermore does not maximize the aggregate source rate. Therefore, it is important to adjust the rate increment terms to achieve convergence and maximize the aggregate source rate.

For each setting that we explore, our algorithm under $\beta_{sj}(0) = 0.1$ converges to obtain a normalized aggregate source rate close to that obtained under $\beta_{sj}(0) = 0$ at a much faster convergence rate. Table I lists the convergence time under $\beta_{sj}(0) = 0.1$ for various values of skew factor, b, and the ratio of network bandwidth over the aggregate source demands, f, when $m_s = 1.2$ Mbps, $\alpha = 1.1$ and $\gamma = 2.0$ (the results under $\alpha = \gamma = 1.1$ are similar). We observe that all of the convergence times are within 2 minutes. The convergence time is very short when the network bandwidth is much lower than the source requirement (e.g., when $f = 0.6$) or when there is a large amount of extra bandwidth in the network (e.g., when $f = 3.0$). Using more paths may lead to a slower convergence under certain settings (e.g., when $b = 0.5$ and $b = 1$, i.e., the relay-receiver bandwidths are skewed). Last, the convergence speed when $m_s = 6$ Mbps is similar to that when $m_s = 1.2$ Mbps, indicating that our scheme can be used for applications with high bandwidth demands.

V. TESTBED EXPERIMENTS

To demonstrate the practicality of our application-level controller, we have implemented it on top of Linux. We next briefly describe our implementation and preliminary results in a local testbed. We stress that the purpose of this section is to demonstrate that our scheme is easy to deploy not to to present an extensive evaluation of our scheme in a testbed.

In our implementation, a TCP connection is established on each logical hop from a source to a receiver. The receiver reassembles data over multiple paths from a source according to application-level sequence numbers that are embedded in the packets. Each packet is 1008 bytes. A relay has an application-level buffer to hold 5 or 10 packets. Furthermore, the TCP sender and receiver socket buffers at the relay are set to hold 5 or 10 packets. The small buffers (at both the application and transport level) are to avoid excessive buffering at the relays. Data coming into the relay are buffered and then forwarded to the next hop. A full buffer at the relay suppresses the sending rate of the previous hop. Refinement of our implementation (e.g., how to set the size of the TCP socket buffers and the application-level buffer) is left as future work.

Our local testbed contains two sources, three relays and a receiver, as shown in Fig. 4. These hosts are connected by routers. Source s_1 sends data to relays r_1 and r_2, which forward incoming data to the receiver. Similarly, source s_2 sends data via relays r_2 and r_3. The routers are configured so that the source rates are only constrained on the second

logical hop, i.e., from the relays to the receiver. This bandwidth limitation is through serial ports connecting two routers. We do not emulate network delays in our testbed. Instead, we use relatively low link bandwidths so that the round trip time of the TCP connections from the relay to the receiver ranges from tens to hundreds of milliseconds.

We have performed a set of preliminary experiments in our testbed. The demands of sources s_1 and s_2 are 250 and 300 Kbps, respectively. The bandwidths from the relays to the receiver are varied to create different settings. In all of the settings, our controller obtains rate allocations as we expected. In the interest of space, we only describe the results in one setting in detail. In this setting, the bandwidths from relays r_1, r_2 and r_3 to the receiver are 256, 256 and 56 Kbps, respectively. The length of the control interval for a source is the duration to send 20 packets at the maximum source rate (i.e., 0.645 and 0.538 second for sources s_1 and s_2 respectively); the small constant, ϵ, is 1 packet and the threshold to decide whether a rate increment is successful, δ, is 0.1. The initial rate increment term is 0 (i.e., $\beta_{sj}(0) = 0$, $s = s_1, s_2$, $j = 1, 2$). For each source, the initial path selection probability is 0.5 for each path. The TCP socket buffers and the application-level buffer of the relays are set to hold 10 packets. Initially, source s_1 sets the application-level path rates on the two paths to be both half of its demand; source s_2 sets application-level path rates to be 10 and 6 packets per control interval. Fig. 5 plots the throughput of each source measured at the receiver versus time. Each data point is averaged over 2 seconds. We observe that source s_1 gradually moves data from the path via relay r_2 to that via relay r_1. Consequently, source s_2 increases its path rate on the path via relay r_2 and obtains the maximum sending rate in approximately 10 seconds. This demonstrates that our scheme can effectively discover spare network bandwidth to improve the aggregate source rate.

VI. Conclusions and Future Work

In this paper, we developed an application-level multipath rate controller via TCP. Our controller utilizes the bandwidth probing mechanisms embedded in TCP and does not require explicit network knowledge (e.g., topology, available bandwidth). We theoretically prove the convergence of our algorithm in certain settings. Furthermore, using a combination of simulation and testbed experiments, we demonstrate that our algorithm provides efficient multipath data transfer and is easy to deploy.

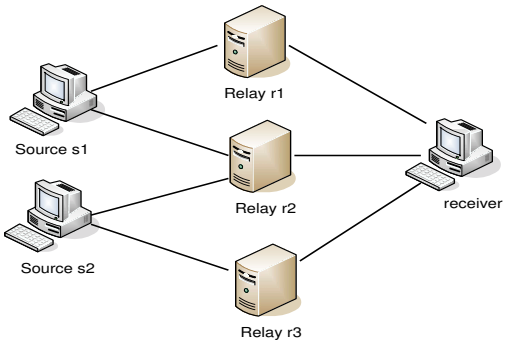

Fig. 4. Illustration of the testbed.

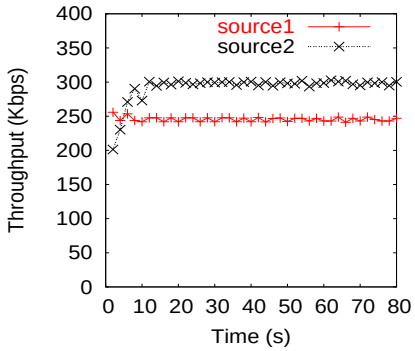

Fig. 5. Throughput measured at the receiver from a testbed experiment.

As future work, we are pursuing the following directions: (1) performance evaluation in more general settings (with multiple receivers and/or bandwidths constrained on the first logical hop); (2) more systematic study of our scheme in a larger testbed under more realistic conditions.

References

[1] *Engineering Research Center for Collaborative Adaptive Sensing of the Atmosphere*. http://www.casa.umass.edu.

[2] http://www.sdss.org/.

[3] J. Gray and A. S. Szalay, "The world-wide telescope, an archetype for online science," Tech. Rep. MSR-TR-2002-75, Microsoft Research, June 2002.

[4] A. Akella, S. Seshan, and A. Shaikh, "An empirical evaluation of wide-area internet bottlenecks," in *IMC*, (Miami, Florida), 2003.

[5] A. Akella, B. Maggs, S. Seshan, A. Shaikh, and R. Sitaraman, "A measurement-based analysis of multihoming," in *Proc. ACM SIGCOMM*, August 2003.

[6] A. Akella, J. Pang, B. Maggs, S. Seshan, and A. Shaikh, "A comparison of overlay routing and multihoming route control," in *Proc. ACM SIGCOMM*, 2004.

[7] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Trans. Networking*, 1999.

[8] J. Chen, P. Druschel, and D. Subramanian, "An efficient multipath forwarding method," in *Proc. IEEE INFOCOM*, 1998.

[9] S. Vutukury and J. Garcia-Luna-Aceves, "MPATH: a loop-free multipath routing algorithm," *Elsevier Journal of Microprocessors and Microsystems*, pp. 319–327, 2000.

[10] W. T. Zaumen and J. J. Garcia-Luna-Aceves, "Loop-free multipath routing using generalized diffusing computations," in *INFOCOM (3)*, pp. 1408–1417, 1998.

[11] B. Cheng, C. Chou, L. Golubchik, S. Khuller, and Y.-C. Wan, "Large scale data collection: a coordinated approach," in *Proc. IEEE INFOCOM*, (San Francisco, CA), March 2003.

[12] S. Ganguly, A. Saxena, S. Bhatnagar, S. Banerjee, and R. Izmailov, "Fast replication in content distribution overlays," in *Proc. IEEE INFOCOM*, (Miami, FL), March 2005.

[13] F. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability," in *Journal of the Operational Research Society*, vol. 49, 1998.

[14] W.-H. Wang, M. Palaniswami, and S. H. Low, "Optimal flow control and routing in multi-path networks," *Performance Evaluation*, vol. 52, pp. 119–132, 2003.

[15] X. Lin and N. B. Shroff, "The multi-path utility maximization problem," in *41st Annual Allerton Conference on Communication, Control, and Computing*, (Monticello, IL), October 2003.

[16] S. H. Low, "Optimization flow control with on-line measurement," in *Proceedings of the 16th International Teletraffic Congress*, (Edinburgh, U.K.), June 1999.

[17] B. A. Movsichoff and C. M. L. H. Che, "Decentralized optimal traffic engineering in the Internet," *IEEE Journal on Selected Areas in Communications*, 2005.

[18] K. Kar, S. Sarkar, and L. Tassiulas, "Optimization based rate control for multipath sessions," in *Proceedings of Seventeenth International Teletraffic Congress (ITC)*, (Salvador da Bahia, Brazil), December 2001.

[19] R. Srikant, *The Mathematics of Internet Congestion Control*. Springer-Verlag, March 2004.

[20] H. Han, S. Shakkottai, C. Hollot, R. Srikant, and D. Towsley, "Overlay TCP for multi-path routing and congestion control," in *IMA Workshop on Measurements and Modeling of the Internet*, JAN 2004.

[21] F. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, pp. 33–37, January 1997.

[22] B. Wang, J. Kurose, D. Towsley, and W. Wei, "Multipath overlay data transfer," Tech. Rep. 05-45, Department of Computer Science, University of Massachusetts, Amherst, 2005.

[23] A. Dhamdhere and C. Dovrolis, "ISP and egress path selection for multihomed networks," in *Proc. IEEE INFOCOM*, (Barcelona, Spain), April 2006.

[24] J. Han, D. Watson, and F. Jahanian, "Topology aware overlay networks," in *Proc. IEEE INFOCOM*, March 2005.

[25] K. P. Gummadi, H. V. Madhyastha, S. D. Gribble, H. M. Levy, and 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.

[26] H. Pucha and Y. C. Hu, "Overlay TCP: Ending end-to-end transport for higher throughput," in *Proc. ACM SIGCOMM*, August 2005.

[27] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 2nd ed., 1999.

[28] http://www.ilog.com/products/cplex/.

[29] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. The MIT press, 1990.

APPENDIX
PROOF OF THEOREM 1

Proof: When $\beta_{sj}(0) = 0$, we refer to our application-level rate controller as *A-AIAD* since the rate increment/decrement under this condition is simply by adding/substracting the small constant, ϵ. We prove this theorem by first transforming the rate control problem into a network flow problem [29]. We construct a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to represent the network we consider as follows. The vertex set $\mathcal{V}$ contains the set of multipath sources S, the set of relays R, the destination d and an additional vertex b, referred to as the *origin*. We use (u, v) to represent a directed edge from u to v, $\forall u, v \in \mathcal{V}$. Furthermore, let c_{uv} denote the capacity on the directed edge (u, v). The origin b and each source $s \in S$ is connected by a directed edge (b, s) with the capacity as the demand of the source, that is, $c_{bs} = m_s$. If source $s \in S$ selects a relay $r \in R$, then s is connected to r by a directed edge (s, r). The capacity of the edge (s, r), c_{sr}, is the available bandwidth on the path from s to r. A relay $r \in R$ is connected to the destination d by a directed edge (r, d). The capacity of the edge (r, d), c_{rd}, is the available bandwidth on the path from the relay to the destination. In the directed graph $\mathcal{G}$, if two vertices u and v are not connected, i.e., $(u, v) \notin \mathcal{E}$, then $c_{uv} = 0$.

Let $f(u, v)$ be the network flow from vertix u to v. We next describe the how to set the initial value of the flow between two vertices u and v, $\forall u, v \in \mathcal{V}$. Let x^0_{sj} denote the initial rate allocation on the jth path of source s, $s \in S, j = 1, 2, \ldots, k$. Then $f(b, s) = \sum_{j=1}^{k} x^0_{sj}$. That is, the flow from the origin to source s is the source rate of this source. Let r_{sj} denote the relay used by the jth overlay path of source s. Then $f(s, r_{sj}) = x^0_{sj}$. That is, the flow from a source to its selected relay is the sending rate on that overlay path. On the edge from a relay r to the destination d, the flow $f(r, d) = \sum_s \sum_{j=1}^{k} \mathbf{1}(r_{sj} = r) x^0_{sj}$, where $\mathbf{1}(\cdot)$ is the indicator

function. That is, the flow from a relay to the destination is the aggregate sending rate over all sources that uses that relay to the destination. The flows of all other edges are 0.

We next show that A-AIAD has positive probability to find *augmenting paths* in the *residual network* [29]. For completeness, we briefly describe residual network and augmenting path. Given a flow network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and the flows between two vertices, a residual network $\mathcal{G}_f$ induced by these flows is $\mathcal{G}_f = (\mathcal{V}, \mathcal{E}_f)$, where $\mathcal{E}_f = \{(u, v \in \mathcal{V} \times \mathcal{V} : c_f(u, v) > 0\}$, where $c_f(u, v)$ is the *residual capacity* of edge (u, v), i.e., $c_f(u, v) = c_{uv} - f(u, v)$. Given the residual network $\mathcal{G}_f$, an augmenting path is a path from the origin b to the destination d in $\mathcal{G}_f$. By the definition of residual network, each edge along an augmenting path can admit positive flow without violating the capacity of this edge.

We represent an augmenting path by the sequence of vertices along the path. Let $\mathcal{P} = (b, s_{i_1}, r_{i_1}, \ldots, s_{i_n}, r_{i_n}, d)$ be an arbitrary augmenting path in the residual network, where $n \geq 1$. Since edge (b, s_{i_1}) can admit positive flow, source s_{i_1} is not satisfied. Let $c_f(\mathcal{P})$ be the minimum residual capacity along this path. That is, $c_f(\mathcal{P}) = \min\{c_f(u, v), (u, v) \in \mathcal{P}\}$. Under perfect detection of network congestion, a source increases its sending rate on a path iff there is spare bandwidth on that path. We next prove that, with perfect congestion detection, there is a positive probability for A-AIAD to find this augmenting path and increase the flow on the path by $c_f(\mathcal{P})$. We prove this by induction on n.

- Case 1 ($n = 1$). In this case, when using A-AIAD, there is a positive probability that source s_{i_1} increases its sending rate on the path from s_{i_1} to the destination via relay r_{i_1} by the amount of $c_f(\mathcal{P})$.
- Case 2 ($n > 1$). We first show that it is sufficient to consider augmenting paths in which sources $s_{i_n}, s_{i_{n-1}}, \ldots, s_{i_2}$ are all satisfied. Suppose source s_{i_n} is not satisfied. Then there is an augmenting path of (b, s_{i_n}, r_{i_n}, d). From Case 1, when using A-AIAD, there is a positive probability for s_{i_n} to increase its path rate on (s_{i_n}, r_{i_n}, d) until it is satisfied or the path rate cannot be increased any more (i.e., either path (s_{i_n}, r_{i_n}) or path (r_{i_n}, d) is saturated). The former case is desired. In the latter case, path $\mathcal{P}$ is not an augmenting path any more (so we do not need to consider path $\mathcal{P}$ any more). Similarly, we only need to consider augmenting paths in which sources $s_{i_{n-1}}, \ldots, s_{i_2}$ are all satisfied. When sources $s_{i_n}, s_{i_{n-1}}, \ldots, s_{i_2}$ are all satisfied, the aggregate source rate can be increased by $c_f(\mathcal{P})$ when A-AIAD adjusts the sending rates in the following manner: source s_{i_n} gradually shifts its data from the path $(s_{i_n}, r_{i_{n-1}}, d)$ to path (s_{i_n}, r_{i_n}, d), thus leaving spare bandwidth on the path of $(r_{i_{n-1}}, d)$ and allowing source $s_{i_{n-1}}$ to shift its data from $(s_{i_{n-1}}, r_{i_{n-2}}, d)$ to $(s_{i_{n-1}}, r_{i_{n-1}}, d)$, ..., and allowing source s_{i_1} to increases its sending rate on the path of (s_{i_1}, r_{i_1}, d). This sequence of rate adjustment leads to a rate increment of $c_f(\mathcal{P})$ in the aggregate source rate.

Since path $\mathcal{P}$ is arbitrary, we have proved that A-AIAD can find any augmenting path in the residual network. UC-maxmin continues the process of finding an augmenting path and adjusting rate along that augmenting paths until no augmenting path can be found. This is equivalent to the Ford-Fulkerson algorithm in maximum network flow [29]. Suppose at time T, no augmenting path can be found. Then the maximum aggregate source rate is reached [29]. We prove that later rate changes of A-AIAD does not lower the aggregate source rate (hence the rate allocation converges) by considering the following two cases:

- Case 1: all relay bandwidths are fully utilized at time T. The rate allocation does not change in this case, and hence A-AIAD converges.
- Case 2: not all relay bandwidths are fully utilized at time T. If a relay is not selected by any source, it can be removed without affecting the rate allocation. Therefore, without loss of generality, we assume each relay is selected by at least one source. Consider an arbitrary relay r with spare bandwidth and an arbitrary source s that selects relay r. If source s is satisfied, it may shift its data from other paths to path (s, r, d). However, by the assumption, the shifting occurs iff there is still spare bandwidth on path (s, r, d), which does not affect the sending rate of any other source, and hence does not reduce the aggregate source rate. If source s is not satisfied, then there is no spare bandwidth on the path of (s, r). Otherwise, the sending rate of source s can be increased, which contradicts with that the maximum aggregate source rate has been reached. Under

the assumption of perfect bandwidth detection, source s does not increase the rate on path (s, r, d) and hence does not affect the aggregate source rate.

■