Application Level Relay for High-Bandwidth Data Transport *

Yong Liu[a], Yu Gu[a], Honggang Zhang[a], Weibo Gong[b] and Don Towsley[a]
[a]Computer Science,
University of Massachusetts,
Amherst, MA 01003
{yongliu, yugu, honggang, towsley}@cs.umass.edu

[b]Electrical & Computer Engineering,
University of Massachussetts,
Amherst, MA 01003
gong@ecs.umass.edu

September 19, 2004

Abstract

Data intensive applications require massive amount of data to be transported over shared wide area networks. Traditional network congestion control and routing schemes have proven inadequate for fully utilizing available network resource for high-bandwidth data transport. In this work, we explore the flexibility of control at the application layer and propose various application level data relay schemes to largely improve the data throughput by optimally integrating application level routing and transport layer control. The proposed algorithms can be easily implemented in overlay networks. Preliminary experiments have shown that our relay schemes are efficient in utilizing network resource for high-bandwidth data transport. The impact of application level relays on the underlay network is also discussed.

1 Introduction

High-bandwidth data transport has become increasingly important with the emergence of data intensive network applications, such as peer-peer applications and grid computing. At the same time, advances in communication technology have made available physical network infrastructure that can support point to point data transmission at the speed of tens of Gigabits per second. How to fully utilize network capacity for high-bandwidth end-end data transport has generated considerable research on network resource allocation. Network resource allocation is carried out by two mechanisms, namely *routing* and *congestion control*, operating at different time scales. At a coarse time scale, a routing algorithm aims at choosing routes for all data transport pairs so as to optimize the overall network performance [1, 2]. Largely due to the complexity of the topology and variability in traffic demand, the capability of traffic engineering in the Internet is still very limited. At a finer time scale, congestion control schemes adjust sending rates of data sources to avoid congestion along the paths provided by a routing algorithm. The objective of network congestion control mechanisms, including transport control at the edge and queue management inside the network core, is to maintain fair bandwidth sharing among competing users and to ensure the stability of the network as a whole [3, 4]. Most existing transport control protocols work in the so called "end-end" fashion. The end-end principle keeps the network core *transparent* to

*This work is supported in part by NSF under grants ANI0240487, ANI0085848 and EIA-0080119, and DARPA under contract DOD F30602-00-0554. Any opinions, findings, and conclusions of the authors do not necessarily reflect the views of NSF and DARPA.

end users. However, traditional transport control protocols, such as TCP, have proven to be inefficient at obtaining available bandwidth in high bandwidth-delay product networks. New end-end congestion control protocols, such as HighSpeed TCP [5], FAST [6], XCP [7], have been proposed to achieve high speed data transport.

Recently, there is an increasing trend to employ application level control to improve users' performance over shared wide area networks. The flexibility of control at the application layer helps to overcome inefficiencies in underlay network routing and congestion control. For example, application level overlay networks have emerged to provide services which cannot be embedded in the Internet [8, 9, 10, 11, 12, 13]. In overlay networks, all participating nodes collaborate at the application level and relay traffic for each other. Application level routing enables end users to deviate from the end-end semantics and data can be relayed from a source to a destination. In this paper, we study how to achieve high-bandwidth data transport through application level relays.

Instead of setting up an end-end connection between the source and destination, an *application relay* employs multiple *relay nodes* to sequentially relay data to its destination. Each relay node buffers packets upon receipt and is responsible for reliable delivery of packets to the downstream relay node (or the final destination). Between two neighboring relay nodes, data transmission is managed by a transport control protocol. We choose the universally deployed TCP for data transport between two relay nodes. However, our application relay framework is independent of the choice of TCP. Any transport control protocol can be used as a primitive to build up the relay. We refer to a path along which data is relayed as a *relay path*. A relay path may or may not follow the default route between its source and destination provided by the underlay network routing. For example, in an overlay network, a source node can specify a sequence of overlay nodes to relay traffic to a destination. Our focus is on how to organize application relays to maximize the achieved end-end throughput. We start with the assumption that application relays have negligible impact on the underlay network and demonstrate how network users can greatly increase their throughput by employing relays of TCP connections for their data transmission. Application level relays' impact on the underlay network is discussed at the end of the paper.

The paper is organized as follows: In Section 2, we briefly describe related work on utilizing application level control to improve end users' performance, such as Parallel TCP, overlay routing and multicast overlay, etc. In Section 3, we study the application relay along the data path provided by the underlay network. We show that by optimally organizing TCP connections, both sequentially and in parallel, one can significantly improve the throughput of end-end data transport. How to organize TCP relay in general overlay networks is investigated in Section 4. Specifically, the optimal TCP relay path problem is studied as a multi-metric application level routing problem. Various algorithms are proposed to find relay paths optimized for data transport performance metrics, such as throughput and hop-count, as well as overlay network performance measures, such as the maximal workload on all relay nodes. Preliminary experimental results are presented in Section 5 to demonstrate the efficiency of proposed relay schemes. Section 6 is devoted to general discussions on application level relays' impact on underlay networks, especially the performance of regular users not employing application layer control. The paper is concluded with future works in Section 7.

2 Related Work

There have been many efforts on improving data transport throughput through application level control. Concurrent downloads have been widely used in web applications, such as FlashGet [14], to significantly speedup HTTP downloads. These applications *parallelize* the download of a web object by opening multiple connections per object and downloading a different portion of the object on each

connection. Concurrent downloads are also employed by popular peer-peer applications, such as BitTorrent [15]. A peer-peer user sets up parallel TCP connections to multiple peers to concurrently download different portions of requested data object. Concurrent downloads are also employed by a variant of FTP, GridFTP [16], which has been proposed to guarantee high bandwidth data transfer for grid applications. Studies in [17] have shown that concurrent downloads violate, at the application level, the fairness that TCP tries to maintain at a connection level. A recent study [18] on parallel TCP describes an approach to preserve the effectiveness when the network is under-utilized and prioritizes the fairness with competing traffic when the network is fully utilized. Our work studies the performance improvement of application level relay formed by *sequential* TCP connections.

Overlay routing allows end hosts to choose application level routes by themselves. It is shown that overlay routing schemes are effective in dealing with some of the deficiencies in today's IP routing. For example, measurements from overlay routing projects RON [9] and Detour ([8], [19]) have shown that a large percentage of Internet flows can find better alternative paths by relaying among overlay nodes, thereby improving their performance. It was shown in [10] that overlay networks can enhance Quality-of-Service(QoS) perceived by users without any support from the underlying Internet. Recently, application overlays have been proposed to support group communications [11, 12, 13]. It is shown in [12] that overlay based group communication using TCP scales in both the obtained throughput and the buffer required as the group size gets large. Optimal tree construction algorithms are discussed to maximize the throughput of the multicast group. [13] proposes an overlay multicast architecture which incorporates the use of loosely coupled TCP connections to deliver a scalable solution that better accommodates a set of heterogeneous receivers. It is also pointed out that a chain of TCP connections can achieve higher throughput than a single end-end TCP connection.

Our work focuses on how to set up application level relay paths optimized for data transport performance metrics, such as throughput and hop-count, as well as overlay network performance measures, such as the maximal workload on all relay nodes. We study this as an application level multi-metric routing problem. There is a rich literature on QoS routing to satisfy multiple performance constraints [20, 21, 22, 23, 24]. Various algorithms have been proposed to find paths satisfying end-end performance constraints, such as loss, delay and bandwidth, etc. It is not a surprise that our problem has much in common with QoS routing. The unique nature of data relay in an overlay network makes our work different from previous studies. Application level control provides us more flexibility in the design and implementation of the proposed data relay schemes.

3 TCP Relay for Long-haul Data Transport

The performance of end-end transport protocols degrades in long-haul data transmission over lossy links. Recent studies have shown TCP cannot obtain available bandwidth in high bandwidth-delay product networks [5, 6, 7]. The fairness and throughput of TCP suffer when it is used in mobile ad hoc networks (MANETs) [25]. One solution is to deviate from the end-end semantics and use multiple relay nodes between a source and destination to relay traffic. By doing so, one breaks a long end-end connection into multiple shorter connections to improve efficiency and fairness. In this section, we limit ourselves to the case that the relay path aligns with the default route. We study how to optimally place relay nodes along the default data path to improve end-end throughput. We start with the sequential relay case, where TCP connections are concatenated as a *pipeline* to relay data. We show that TCP pipelines greatly improve the throughput of data transport. Then we demonstrate how parallel TCP can be employed efficiently to increase the bottleneck bandwidth of a TCP pipeline. The construction of an optimal TCP relay is formulated and solved as dynamic programming problems.

3.1 TCP Pipeline and Throughput Improvement

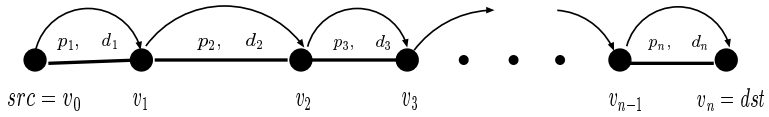


Figure 1: TCP Relay for Long-haul Data Transport

Motivating example: Consider a path on which $n-1$ nodes lie between a sender src and receiver dst (Figure 1). Denote by v_i the ith node on the path, with $src = v_0$ and $dst = v_n$. Let p_i denote the packet loss probability on link $E_i =< v_{i-1}, v_i >$, and d_i the two way packet delay between node v_{i-1} and v_i (two way propagation delay, plus possible queueing delays due to congestion). When $\{p_i\}$ are small, the throughput of a TCP connection between node i and node $j(> i)$ can be approximately characterized by [26]

$$B(i,j) = \frac{C}{\sum_{k=i+1}^{j} d_k \sqrt{\sum_{k=i+1}^{j} p_k}} \tag{1}$$

In the simplest case, all links are homogeneous: $p_i = p$ and $d_i = d$. Then the throughput of a end-end TCP connection is $B(0,n) = \frac{C}{nd\sqrt{np}}$. On the other hand, if all the nodes commit to application level packet forwarding, the user can set up n sequential TCP connections, one between each adjacent node pair (v_{i-1}, v_i), to relay packets from src to dst. Each such relay TCP connection will achieve throughput $B(i,i+1) = \frac{C}{d\sqrt{p}}$. The end-end throughput is the minimum of all relay TCP connections. In this case, it is still $\frac{C}{d\sqrt{p}}$, which is $n\sqrt{n}$ times larger than the throughput of the single end-end TCP connection and is $\sqrt{n}$ times larger than the aggregate throughput of n parallel end-end TCP connections between src and dst.

We refer to the set of sequential TCP connections used to relay packets from the source to the destination as a *TCP pipeline*. Formally, a TCP pipeline with m sequential connections is a m-partition, A, of a n-hop path $[0,n]$ such that $0 = A(0) < A(1) < A(2) < \cdots < A(m-1) < A(m) = n$ and the ith TCP connection runs from node $v_{A(i-1)}$ to $v_{A(i)}$. According to (1), the throughput on the ith segment of the pipeline is $B(A(i-1), A(i))$. The end-end throughput of a TCP pipeline A, $T(A)$, is the minimal throughput of all constituent segments:

$$T(A) = \min_{1 \le i \le m} B(A(i-1), A(i))$$

Since A is a partition of $[0,n]$, it is easy to verify that

$$T(A) \le \min_{1 \le i \le n} B(i-1, i),$$

That is to say the throughput of a pipeline of sequential TCP connections is bounded from above by the "slowest" link between src and dst. Obviously, the upper bound will be achieved when the pipeline consists of n sequential connections and $A(i) = i, 0 \le i \le n$.

3.2 How to Set Up a Pipeline Optimally?

Application relays alleviate the discrimination against long-haul connections and the achieved data rate scales with the number of links along a path. It also improves the efficiency of data retransmissions. A packet lost on one relay segment is retransmitted locally on just that segment. It avoids unnecessary retransmission on all links along the path and lost packets are recovered faster. At the same time, data

relay incurs overhead in application level operations. Multiple TCP connections have to be established for one unicast data transport. Along the relay path, all packets are sent in a store-forward manner and experience additional processing delays at intermediate relay nodes. Due to the rate mismatch between TCP connections on different relay segments, packet buffers coupled with back-pressure schemes are necessary to avoid flooding the slowest relay segment. When setting up the pipeline, one has to trade off the throughput gain against memory and computation consumption.

To bound these overhead, we can limit the number of segments on a relay path. Then the question becomes: given at most m TCP connections, how to setup the pipeline to maximize end-end throughput? This is equivalent to the problem of optimally placing $m-1$ nodes to relay traffic from src to dst. For example in Figure 2, if the third link is the slowest link and we are allowed to use only 2 relay nodes, we may want to place the two relay nodes on both ends of the slowest link. Formally, we want to find an optimal m-partition A of $[0, n]$ such that $T(A)$ is maximized:

$$T^* = \max_{|A|=m} \min_{1 \le i \le m} B(A(i-1), A(i))$$
$$A^* = \operatorname*{argmax}_{|A|=m} \min_{1 \le i \le m} B(A(i-1), A(i))$$

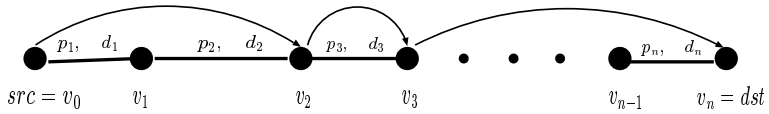


Figure 2: Optimal TCP Pipeline with 3 Segments

This optimal pipeline problem can be solved by dynamic programming. Let $A^*[k, m]$ be an optimal m-connection pipeline between v_0 and v_k, and $T^*[k, m]$ be its throughput. We have

$$T^*[n, m] = \max_{\mathbf{m-1 \le k \le n-1}} \min\left\{T^*[k, m-1], B(k, n)\right\}$$
$$k^* = \operatorname*{argmax}_{\mathbf{m-1 \le k \le n-1}} \min\left\{T^*[k, m-1], B(k, n)\right\}$$
$$A^*[n, m] = A^*[k^*, m-1] \oplus \{n\}$$

The algorithm to find the optimal *m-pipeline* between v_0 and v_n is

for $i = 1$ upto n **do**
  $T^*[i, 1] = B(0, i); A^*[i, 1] = \{0, i\};$
end for

for $j = 2$ upto m **do**
  for $i = j$ upto n **do**
    $T^*[i, j] = \min\left\{T^*[j-1, j-1], B(j-1, i)\right\};$
    $k^* = j - 1;$
    for $k = j$ upto $i - 1$ **do**
      if $T^*[i, j] > \min\left\{T^*[k, j-1], B(k, i)\right\}$ **then**
        $T^*[i, j] = \min\left\{T^*[k, j-1], B(k, i)\right\};$
        $k^* = k;$
      end if

end for
$A^*[i,j] = A^*[k^*, j-1] \oplus \{i\}$
end for
end for

The complexity of solving this dynamic programming problem is $O(n^2 m)$. $T^*[n,m]$ is a non-decreasing function of m and is bounded from above by $T[n,n] = \min_{1 \le i \le n} B(i-1,i)$. When solving the dynamic problem in a bottom-up way, we can find the smallest m^* which achieves $T^*[n,m^*] = T[n,n]$. If $m^* < n$, adding more than m^* sequential TCP connections won't improve the end-end throughput, which is throttled by the slowest link. At the same time, multiple parallel TCP connections helps achieve high throughput across congested link. This motivates us to think about how to optimally combine sequential and parallel TCP connections to achieve higher throughput. For the network in Figure 2, we can use multiple parallel connections to cover the slowest link as shown in Figure 3.

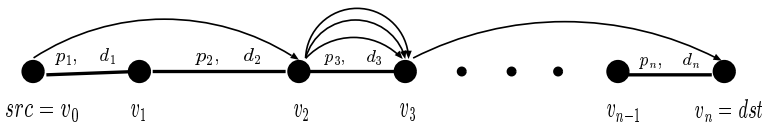


Figure 3: TCP Relay with Sequential and Parallel Connections

3.3 Combining TCP Pipelining and TCP Striping

For very congested relay segments, TCP striping can be employed to increase the bandwidth of the bottleneck on the whole relay path. Data is striped by a sending relay node over parallel TCP connections to a receiving relay node. The receiver reassembles data segments received from different connections and relays them to its downstream neighbor. Data striping and reassembly also incurs memory and computation overhead and introduces additional packet delay. Too many parallel connections can cause more congestion to those already congested links and can degrade the network performance as a whole [27]. In this section, we limit the number of parallel and sequential TCP connections that users can possibly use and study how to organize them optimally to achieve the maximal bandwidth.
As in the previous section, we still partition the path from src to dst into m segments as $0 = Q(0) < Q(1) < Q(2) < \cdots < Q(m-1) < Q(m) = n$. Between nodes $v_{Q(i-1)}$ and $v_{Q(i)}$, there are $s(i) \ge 1$ parallel connections and with the constraint $\sum_{i=1}^{m} s(i) = N$. The optimization problem is:

$$\max_{\{\mathbf{m},\mathbf{Q},\mathbf{s}\}} \min_{1 \le i \le m} s(i) B(Q(i-1), Q(i)),$$

Given m and Q, it becomes an integer programming problem:

$$\max_{|\mathbf{s}|=\mathbf{m}} \min_{1 \le i \le m} s(i) B(Q(i-1), Q(i)),$$

subject to $s(i) \ge 1$ and $\sum_{i=1}^{m} s(i) = N$
If m and Q are not fixed, it can still be solved as the following dynamic programming problem. Let $\{Q^*[k,l], s^*[k,l]\}$ be an optimal organization of l TCP connections between v_0 and v_k, $T^*[k,l]$ be its

throughput. It satisfies

$$T^*[n,N] = \max_{\substack{1\le k\le n-1\\ 1\le j\le N-1}} \min\left\{T^*[k,j],(N-j)B(k,n)\right\}$$

$$\{k^*,j^*\} = \operatorname*{argmax}_{\substack{m-1\le k\le n-1\\ 1\le j\le N-1}} \min\left\{T^*[k,j],(N-j)B(k,n)\right\}$$

$$Q^*[n,N] = Q^*[k^*,j^*]\oplus\{n\}$$

$$s^*[n,N] = s^*[k^*,j^*]\oplus\{N-j^*\}$$

The complexity of solving this problem is $O(N^2n^2)$. The detailed algorithm is:

for $i = 1$ upto n **do**
  $T^*[i,1] = B(0,i); Q^*[i,1] = \{0,i\}; S^*[i,1] = \{1\};$
end for

for $j = 2$ upto N **do**
  for $i = 1$ upto n **do**
    $T^*[i,j] = T^*[i,j-1]$
    for $k = 1$ upto $i-1$ **do**
      for $l = 1$ upto $j-1$ **do**
        if $T^*[i,j] > \min\left\{T^*[k,l],(j-l)B(k,i)\right\}$ **then**
          $T^*[i,j] = \min\left\{T^*[k,l],(j-l)B(k,i)\right\};$
          $k^* = k;\ l^* = l;$
        end if
      end for
    end for
    $Q^*[i,j] = Q^*[k^*,l^*]\oplus\{i\};$
    $s^*[i,j] = s^*[k^*,l^*]\oplus\{j-l^*\};$
  end for
end for

4 TCP Relay in Overlay Networks

In the previous section, we studied how to optimally place relay nodes along the default end-end data path provided by the underlay network. However, underlay network routes are not optimized for the throughput of application level relays. With the flexibility of application level routing in an overlay network, it is possible to find better paths to relay data at higher speeds. In this section, we study the optimal TCP relay problem in an overlay network. We formulate it as a multi-metric application level routing problem. Various algorithms are proposed to find relay paths optimized for data transport performance metrics, such as throughput and hop-count, as well as overlay network performance measures, such as the maximal workload on all overlay nodes.

As illustrated in Figure 4, in an underlay network, a subset of network nodes form an application level overlay network and forward traffic for each other. An overlay network can be represented as a directed graph $G = (V, E)$, where V is the set of overlay nodes and E is the set of all the *logical links* between overlay nodes. Each logical link corresponds to a physical path in the underlay network. (Note: in most cases, the graph is fully connected, meaning each node can reach each other; however, if the overlay runs across multiple ASes, due to the policy based inter-domain routing, some overlay nodes may not

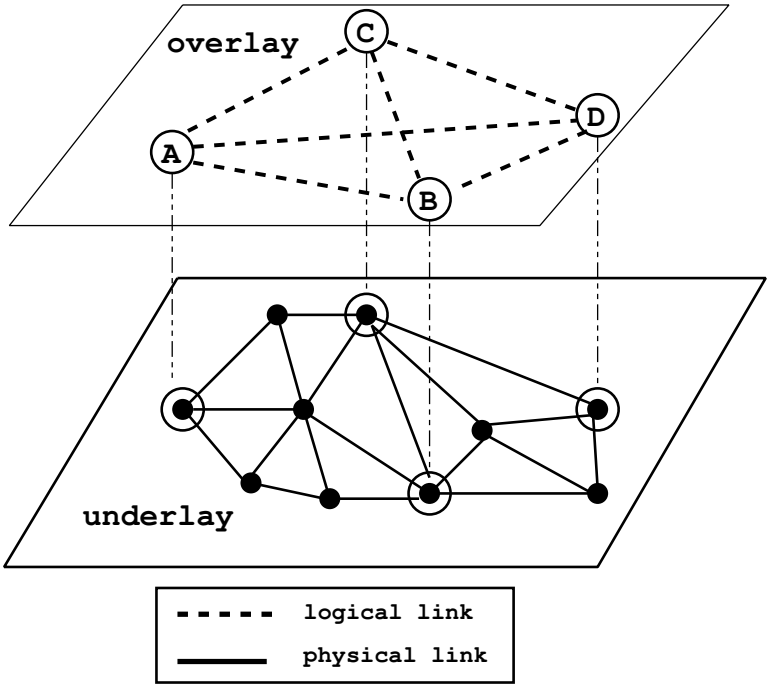


Figure 4: Application Overlay Built upon Underlay Network

able to talk directly to each other. Hence we only assume G is connected.) Through application level routing, an overlay node can specify a path in the logical graph G to reach another overlay node. Data is relayed to its destination on all logical links along the specified path. We refer to a path used to relay data between two overlay nodes as a *relay path*. Most previous work on application level routing focused on finding paths with high resilience against failures and best end-end path performance metrics, such as delay and loss rate, which can translate into the throughput of end-end TCP connections along those paths. Our objective here is to find relay paths optimized for the performance of TCP relays. Throughout this section, we assume that underlay network routes are fixed.

4.1 Widest TCP Relay Path

Our first objective is to find a relay path with the highest throughput. The throughput of a TCP relay path is the minimal TCP throughput on all constituent logical links. We refer to the throughput of a single TCP connection over a logical link $< v_i, v_j >$ as the *local* TCP throughput between v_i and v_j, denoted by $B(i, j)$. $B(i, j)$ can be calculated by the end-end packet loss probability and round trip time from v_i to v_j as in (1). In a directed overlay graph $G = (V, E)$, if we assign weight $B(i, j)$ to link $< v_i, v_j >$, and define the *width* of a path as the minimum weight of all constituent links, an optimal relay path from v_i to v_j is a *widest* path from v_i to v_j.

The widest path problem has been studied in [13, 20]. The widest path tree rooted at a source s can be constructed using a simple variant of Dijkstra's algorithm, which is typically used to construct single-source shortest path trees. The algorithm keeps two sets of vertices: Z, the set of vertices whose widest paths from the source have already been determined and $V - Z$ the remaining vertices. Let $d(v)$ be the best estimate of the widest path for vertex v and $\pi(v)$ be the predecessor on the widest path. The basic mode of operation is:

1. Initialize $d(v) = 0$, $d(s) = \infty$, $Z = \emptyset$,
2. While there are still vertices in $V - Z$,
   (a) Sort the vertices in $V - Z$ according to $d(v)$,
   (b) Add u, the vertex with the largest $d(v)$ in $V - Z$, to Z,
   (c) Relaxation: for v connected to u updates $d(v)$: if $\min(d(u), w(u, v)) > d(v)$, then $\pi(v) = u$ and $d(v) = \min(d(u), w(u, v))$.

Similar to the Dijkstra's shortest-path algorithm, the running time of the entire algorithm is $O(|V|^2)$. It can be implemented as a link-state algorithm. Each overlay node broadcasts its connectivity and link widths to all other nodes. After collecting all the link state information, an overlay node can compute the widest paths to all other nodes. In a real network environment, the local TCP throughput on a logical link varies over time. Therefore the link weight $B(i, j)$ should be calculated from some representative statistics. In most cases, $B(i, j)$ should be quantized to avoid unnecessary complexity introduced by small differences in link width. Quantization is also important when other path metrics, in addition to the width, are also considered. We will discuss this issue later.
If local TCP throughput on all logical links are symmetric, i.e., $B(i, j) = B(j, i)$, we can use an undirected graph to represent an overlay network. The widest relay paths between all node pairs can be efficiently constructed by finding a Minimum Spanning Tree (MST) for the associated undirected graph G. For example, Figure 5 illustrates an overlay network with symmetric links, let $B(i, j) = B(j, i) = 1$, if $i + 1 = j$, $2 \leq j \leq 5$ and $B(i, j) = 0.1$ otherwise. Then the best path connecting v_1 and v_5 is $< v_1, v_2, v_3, v_4, v_5 >$ instead of the direct link between v_1 and v_5.

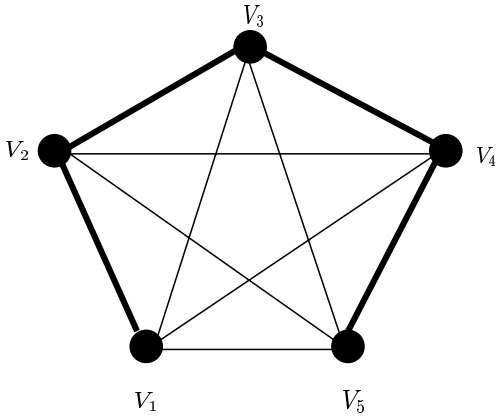


Figure 5: Overlay Network with Symmetric Logical Links

Theorem 1 *In the undirected graph G, assign weight $\frac{1}{B(i,j)}$ to link $< v_i, v_j >$ and construct a Minimum Spanning Tree S, then the path in S between any two nodes v_i and v_j is an widest relay path from node v_i and v_j.*

Proof: We prove it by contradiction. Let S be a Minimum Spanning Tree of G, $p = \{i, \cdots, j\}$ be the path in S connecting v_i to v_j, and $< v_{k_1}, v_{k_2} >$ be the link along p with the maximal weight $\frac{1}{B(k_1,k_2)}$, then the throughput of relay path p equals to $B(k_1, k_2)$. Suppose there is another path $\hat{p}$ in G between v_i and v_j with throughput higher than $T(p) = B(k_1, k_2)$, then all the links on $\hat{p}$ will have local TCP throughput higher than $B(k_1, k_2)$. Removing link $< v_{k_1}, v_{k_2} >$ from S will partite S into

two trees S_1, S_2. Suppose v_i is in S_1 and v_j is in S_2, since $\hat{p}$ connects v_i and v_j, there must be one link $< v_{h_1}, v_{h_2} >$ connects S_1 and S_2. Adding $< v_{h_1}, v_{h_2} >$ to S_1 and S_2 will form a new spanning tree $\hat{S}$. Since $B(h_1, h_2) > B(k_1, k_2)$, $\hat{S}$ will have smaller cost than S, which leads to contradiction. ■

For the example in Figure 5, all the links will have weight 10 except for those links on the path $S =< v_1, v_2, v_3, v_4, v_5 >$. Therefore S is the MST and optimal path between node i and $j(> i)$ is always $L = \{i, i+1, \cdots, j\}$. Widest paths constructed from the Minimum Spanning Tree are naturally loop free. Minimum Spanning Trees can also be used to construct the optimal overlay multicast trees [12]

4.2 Widest TCP Relay Path with Limited Hops

The number of logical links on a relay path corresponds to the number of overlay nodes participating in the data relay. As discussed in Section 3, data relay involves computation and memory overhead at the intermediate relay nodes. Within an overlay network, buffer and computation power for relaying traffic are important resources, which are contended for by competing overlay users. The number of relay nodes on a relay path is an important measure of resource consumption of the corresponding data transport. Therefore, it is important to enforce a limit on the number of overlay nodes on a relay path. The optimal relay problem with this path length constraint is to find a path with no more than m hops to achieve the highest throughput.

To find optimal relay paths with no more than m hops from a single overlay node s to all other overlay nodes, we develop a variant of the *Bellman-Ford* (**BF**) algorithm, which is designed to find the shortest-paths from a single source to all other nodes for all hop counts. The **BF** algorithm is an iterative algorithm. At the h-th iteration, it finds the shortest path between the source and all the destinations with at most h hops. If we want to find an optimal relay path with no more than m hops, we run the following BF-like iterative algorithm up to m iterations. Again we assign $B(i, j)$, the local TCP throughput between node v_i and v_j, as the weight for the link $< v_i, v_j >$ in the directed overlay graph $G = (V, E)$. Let $W[v, k]$ be the width of the optimal path from the source s to the node v with no more than k hops. The algorithm is

$W[v, k] = 0, W[s, k] = \infty$, $0 \leq k \leq m$;
for $k = 1$ upto m **do**
  for $u \in V$ **do**
    W[u,k]=W[u,k-1];
  end for
  for $< u, v > \in E$ **do**
    if $\min(W[u, k-1], B(u, v)) > W[v, k]$ **then**
      $W[v, k] = \min(W[u, k-1], B(u, v))$
    end if
  end for
end for

If we set $m = |V|$, after the completion of the algorithm, we can identify the smallest possible number of relay hops to achieve the highest throughput for all destinations. The running time for the algorithm is $O(m|E|)$. It can be implemented as a distributed distance-vector algorithm. Each node communicates only with directly-attached neighbors about the widest paths to other destinations with hop constraints. It is has been shown in [24] that the basic **BF** algorithm is not the most efficient in finding the hop-constrained *widest* paths when the network is densely connected. An Improved Bellman-Ford (**IBF**) algorithm is proposed with lower asymptotic complexity $O(|E|log|V| + m(|V|^2/log|V|))$. Interested readers are referred to [24] for details. It deserves study to compare **IBF** with **BF** on different size overlay networks.

To find optimal relay paths with no more than m hops between all pairs of overlay nodes, we can

develop an algorithm similar to the recursive all-pairs shortest-paths algorithm. Let $N(v)$ be the set of neighbors of node v; $W[s,t,k]$ be the width of the optimal path from s to t with no more than k hops. The algorithm is in the bottom-up order of k:

for $s,t \in V$, **do**
  if $(s,t) \in E$ **then**
    $W[s,t,1] \leftarrow B(s,t)$
  else
    $W[s,t,1] \leftarrow 0$
  end if
end for
for $k = 2$ upto m **do**
  for $s,t \in V$ **do**
    $WP(s') = \min\{W[s',t,k-1], B(s,s')\}$, $\forall s' \in N(s)$;
    $W[s,t,k] = \max\{W[s,t,k-1], \max_{s' \in N(s)} WP(s')\}$
  end for
end for

4.3 TCP Relay Path Optimized for Width and Length

So far, we have focused on finding the highest throughput TCP relay paths with or without a constraint on the number of relay hops. In this section, we introduce *path length* as an additional metric to search for optimal TCP relay paths. When the path length is used as a secondary optimization criterion, an optimal relay path is a *shortest-widest path* from a source to destination. Under certain situations, the strict preference of path width over path length should be relaxed. One way to trade off width against length is to find the *shortest path satisfying width requirement*.

Since an end-end path is bounded just by its "narrowest" link, all other links on the path have no effect on the path's width. Therefore, there are normally many widest paths with equal width between two nodes and a path with loops may also qualify as a widest path. This problem is more serious in an overlay network, where two different logical links will have the same weights if they physically share the same bottleneck link and have similar round-trip delays. Therefore, in an overlay network, it is easy to have many logical paths, which have the same (or very close) width. In Figure 6, all possible logical paths from A to D are physically constrained by A's access link, which is the only bottleneck link in the network. If the delays on logical links $A-C$ and $A-B$ are equal, $A-C-D$, $A-B-D$, $A-C-B-D$ and $A-B-C-D$ are all widest paths from A to D. The *path length* can be used as the

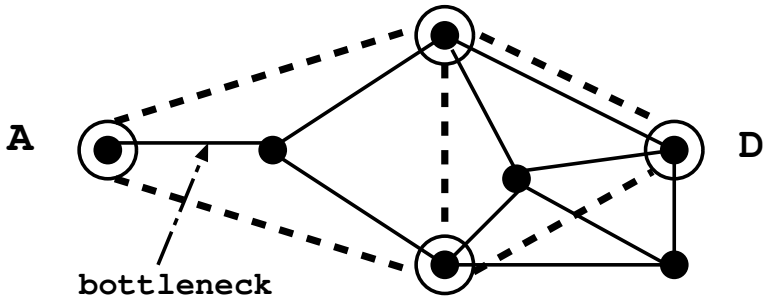


Figure 6: Ambiguity of Widest Path in Overlay Networks

secondary optimization criterion when there is a tie in path width. That is to say we want to find the so called *shortest-widest path* (*SWP*), i.e., the shortest one among all the widest paths. As studied in the previous section, *logical hop count* can be used to measure the length of a relay path. *Physical hop*

count, the number of physical links on a path, is also an important length measure. The more physical links on the relay path, the more network resources consumed by the corresponding data transport. When two relay path are equally wide, one should always choose the relay path with smaller *physical hop count* to minimize network resource consumption by one data transfer, which in turn maximizes the overall network utilization. However, end-end physical length, or equivalently propagation delay, is not an important metric for relay path. The latency of transmitting a large file is dominated by the transmission delay, which is determined only by the relay path's throughput.
In our study, we use either *logical hop count* or *physical hop count* as the second relay path metric. In an overlay graph, if we assign 1 as the length of each logical link, the length of a relay path is its logical hop count; if we assign the number of physical links on a logical link as its length, the length of a relay path is its physical hop count. Although multi-metric path optimization in general is an intractable problem [28, 20], the choice of bandwidth and length as path metrics makes the complexity of the SWP algorithm similar to that of a standard shortest path algorithm. Both *Dijkstra* like link state algorithms and *Bellman-Ford* like distance vector algorithms have been proposed to solve the SWP problem in the context of Quality-of-Service routing [20, 21, 22]. Extra care has to be taken when extending shortest/widest path algorithms due to the following properties of SWP:

1. If a path from A to Z through B, $A \overset{p_1}{\rightsquigarrow} B \overset{p_2}{\rightsquigarrow} Z$, is a SWP from A to Z, it doesn't mean that the path $B \overset{p_2}{\rightsquigarrow} Z$ is a SWP from B to Z; It is true if we replace *SWP* with *shortest path*. This is an important property for the correctness of Dijkstra shortest path algorithms.

2. If $A \overset{p_1}{\rightsquigarrow} B \overset{p_2}{\rightsquigarrow} Z$ is a SWP from A to Z, and $B \overset{p_3}{\rightsquigarrow} Z$ is a SWP from B to Z, it is not always true that $A \overset{p_1}{\rightsquigarrow} B \overset{p_3}{\rightsquigarrow} Z$ is a SWP from A to Z; It is true if we replace *SWP* with *widest path*.

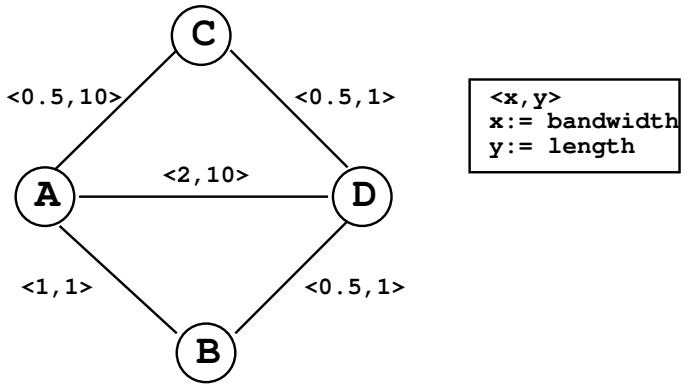


Figure 7: Special Properties of Slowest-widest Paths

For the network in Figure 7, the SWP from C to A should be the path $C-D-B-A$, which has a width of 0.5 and a length of 3. However, $D-B-A$ is narrower than D's SWP $D-A$ (due to 1)) and $C-D-A$ has a length of 11, which is larger than that of C's SWP (due to 2)). Shortest-widest relay paths can be found in the following two-phase search:

1. In the first phase, assign local TCP throughput to each logical link in the overlay graph, find the widest paths from the source to all other nodes using Dijkstra like algorithm as described in Section 4.1

2. Let $T(i)$ be the width of a widest path to node i. Sort all the nodes into a list such that $T(i)$ is in increasing order. Assign link lengths as link weights in the graph

3. Start with the node with the smallest $T(i)$, prune all the links in E which have bandwidth less than $T(i)$.

4. Find the shortest path from the source to node i in the pruned graph, that is the shortest-widest path from the source to node i in the original graph, take node i out of the sorted list

5. Go back to step 3), until the list is empty.

In an overlay network, the choice of metric for optimal relay path is application dependent. So far, we give the width of a relay path strict priority over the its length. The length metric is used only when there is a tie in the width metric. This strict priority may be too rigid in a real network environment. Given the dynamics in the underlay network traffic, both the achieved local TCP throughput and packet delay over one logical link vary over time. In other words, all those link weights are not accurate. Therefore link-weight-based search algorithms should not be too rigid, especially regarding the preference among different path metrics. It doesn't make sense to rule out a path that is much "shorter", while being a little bit "narrower", than the shortest-widest path constructed based on a one-time or some mean statistics. The strict priority of bandwidth can be relaxed while still keeping the preference for it. To practically do it, threshold-based searching algorithms can be employed. One can specify a lower bound for the width of a path, which is calculated based on the result of the widest path search and some relaxation margin, and then choose *the shortest path satisfying the width requirement*. This can be done by using the previous prune-search procedure as for the shortest-widest path. The only difference is that the width threshold, instead of the maximal path width, is used to prune links.

4.4 Contention on Relay Nodes

Limiting the length of relay path can alleviate possible contention on some relay nodes. However, it doesn't explicitly control the workload on relay nodes. To avoid overloading some relay nodes, we have to take the contention on relay nodes explicitly into consideration when constructing the application relays for all demand pairs. This is somewhat similar to the network traffic engineering problem, which is to find optimal routes for all demand pairs such that overall link contention (translated into link utilization, or link delay) is minimized. The general load balancing routing can be formulated as the unsplittable flow problem which tries to minimize the maximum workload on one node. It is a well-known NP-hard problem [29]. However, for the purpose of setting up a relay path, if we only use workload on overlay nodes as the objective function to optimize, the solution is trivially that every node sends data directly to its destination. What we really want is a solution which gives us both high aggregate throughput for all data transport pairs and a low contention level on all overlay nodes. This is again a difficult multi-metrics routing problem. Here we propose some heuristic algorithms to solve a relaxed problem. Instead of solving the multi-objective optimization problem, we ask the following question: how to optimally set up relay paths for all data transport pairs so as to maximize the networks' aggregate throughput while each relay node relays traffic for at most m data transport pairs? To exactly solve this problem is also difficult. We developed some heuristics to solve it approximately:

1. Find the shortest-widest path (SWP)s between all demand pairs

2. Sort the demand pairs in decreasing order of path widths; set up a relay workload counter for each overlay node.

3. Take a demand pair out from the head of the sorted list and increase the counter of all the nodes on its SWP by one, and remove the demand pair from the sorted list

4. repeat the previous step until some overlay node's counter reach the upper limit m, then remove it from the relay topology (it can still be used as source or destination for some demand pair, but not relaying nodes)

5. based on the reduced topology, go back to step 1), until all demand pairs have been routed.

If during the execution of this algorithm all relay nodes reach their relaying capacities, remaining demand pairs can only be transmitted directly without any relay.

5 Experiments

We show some experimental results in this section to demonstrate that application level relays effectively increase the throughput of data transport. The experiments were carried in ns-2 simulator. In order to simulate the receiver TCP buffer occupancy and the back-pressure in the TCP relay path, we modified the FullTCP module and the TCP packet header to implement the TCP flow control mechanisms. By adding the advertisement window to the TCP packet header, sender side TCP can adjust the maximum window size and stop sending data when the receiver's TCP buffer is full. We also developed a specific application module that explicitly pulls data from the receiver TCP buffer. This simulates the read operation that real applications do when receiving data from a socket connection. This application module is used to simulate TCP relay nodes.
Two sets of experiments are shown in this section. In the first set, we demonstrate throughput improvement by pipelining TCP connections along the data path provided by the underlying networks. In the second set, we study TCP relays in overlay networks. We show performance improvement by using the widest TCP relay paths with limited hops and shortest relay paths satisfying the width requirement. We performed these experiments several times and got similar results each time.

5.1 TCP Pipeline and Throughput Improvement

In this subsection, we show the performance of TCP pipeline in a linear network topology. As in figure 8, we use a chain topology which consists of 6 duplex links and study the throughput between node $n1$ and $n7$. Each link has a capacity of $20Mbps$ and a $10ms$ propagation delay. On each link we add

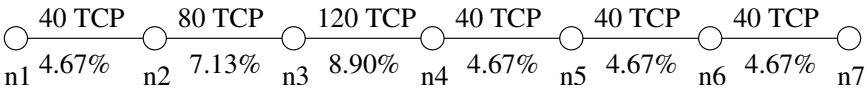


Figure 8: Linear Network Topology

40, 80 or 120 background TCP flows from one end of the link to the other as indicated in the figure. The figure also indicates the loss rate incurred on each link. Links with 120 background TCP flows are the most congested links and those with 40 background TCP flows are the least congested. The number of the background TCP flows is large enough to ignore TCP pipeline's impact on link loss rates. In each experiment, we fix the number of relay nodes and use the algorithm in Section 3.2 to optimally set-up TCP pipeline. In the first case, data is transferred from source to destination via a single TCP connection. In the second case, data is transferred using an optimally set-up TCP pipeline and only 1 relay node is used. In the third case, data is transferred using an optimally set-up TCP pipeline with 2 relay nodes in the middle. Similarly for the 4th case and the 5th case. In the 6th case, data is transferred using a hop-by-hop TCP pipeline. Figure 9 shows the achieved throughput of TCP pipelines optimally established as in Section 3. From this set of experiments, we see that

the throughput increases as the number of relay nodes increases, which shows the performance boost brought by TCP pipelining.

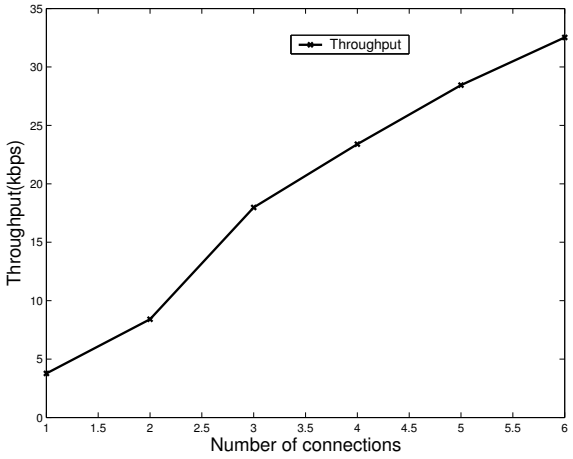


Figure 9: Chain Topology Experimental Results

5.2 TCP Relay in Overlay Networks

We now demonstrate the throughput boost enabled by application relay in an overlay network. Our underlying network consists of 50 nodes and 217 edges, created by the Georgia Tech's topology generator GT-ITM [30]. All links have the same capacity of $10Mbps$ and the delays are set proportional to the distance between two nodes in the GT-ITM generated scenario. Any two links between two nodes have the same loss rate, and the loss rate is randomly set between 0 and 0.02. Among the 50 nodes, we randomly choose 20 nodes to form an overlay network. On this overlay network, we perform two sets of experiments. In one set of experiments, we use the Bellman-Ford algorithm proposed in Section 4.2 to find the widest TCP relay path with different logical hop constraints. In the other set of experiments, we use the algorithm discussed at the end of Section 4.3 to demonstrate the trade-off between the width and length of a relay path.

The first set of experiments in this subsection uses the widest TCP relay path with limited hops algorithm to compute optimal TCP relay paths. We randomly choose two pairs of overlay senders and receivers in the overlay network and observe the throughput achieved using or without using TCP relay. We compare the changes of throughput for the sender/receiver pairs as the maximum number of hops used in TCP relay increases. In these cases, the widest paths do not change every time the upper limit increases and keep the same when the upper limit is larger than 7. Thus we set the maximum hops used in the overlay TCP relay from 1 to 8 and observe the throughput reached in each case. When the maximum hops is set to 1, the underlying path between the sender and the receiver is used. Figure 10 shows the throughput achieved for the three sender/receiver pairs with different TCP relay paths. We see that, when the limit on the maximum relay hops used is increased, wider relay paths can be found and the achieved throughput shows an obvious increasing trend.

In the second set of experiments, we use the algorithm proposed in Section 4.3 to set up the shortest TCP relay path satisfying the width requirement in this 20 node overlay network. The purpose is to illustrate the trade-off between the width and length of a relay path. We choose three source destination pairs. Based on the previous discussions on the trade-off between width and length (logical or physical hop counts) of a path, we relax the width requirement gradually and solve for shortest length (measured in logical or physical hops) at each relaxation level. Experiment results are shown in Figure 11 and 12.

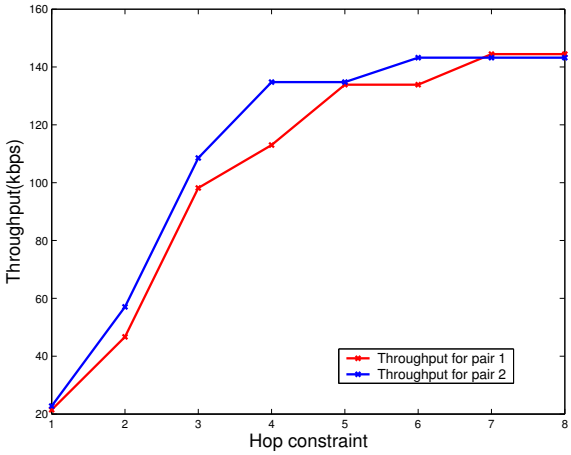


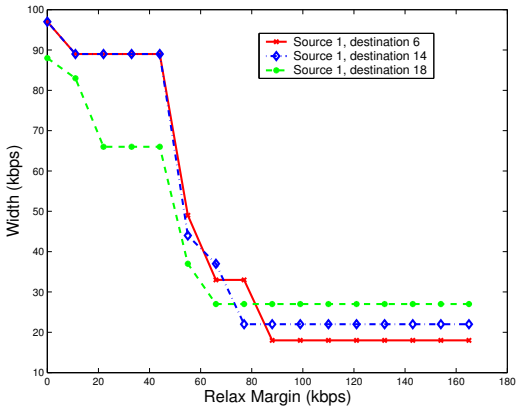
Figure 10: TCP Relay using Widest Path with Limited Hops

The level of path width relaxation is represented by the relax margin, which is the gap between the width of the widest relay path and the width requirement. Figure 11 shows that as we relax the width requirement, we can find shorter path in terms of relay segments at the expense of smaller width. We observe the similar results when we use physical hop counts as the metric for the path length, as shown in Figure 12.

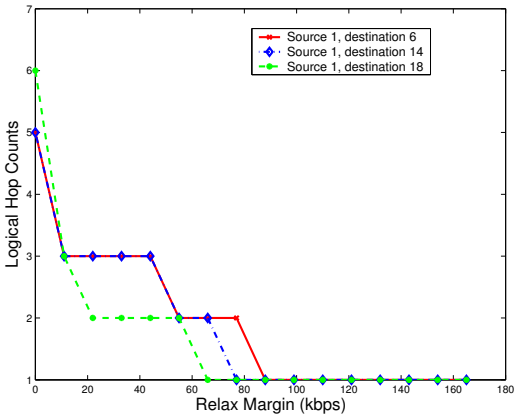
6 Discussions: Impact on the Underlay Network

In previous sections, we have studied various application relay schemes to increase the bandwidth of data transport. We have assumed that an application level relay has negligible impact on the underlay network. This can be justified when the amount of traffic employing application relay constitutes a small portion of the total network traffic. However, end users have a strong motivation to employ application relays to improve their performance. Application relays are becoming more and more popular and they are generating an increasing fraction of network traffic. The fast growth in overlay traffic draws more and more attentions. In this section, we discuss application relays' impact on the underlay network and the performance of normal users. Specifically, our discussions focus on application relays' impact on efficiency and fairness.

Application relays overcome deficiencies in underlay network congestion control and routing. They improve network efficiency in the following ways. Application relay paths are set to maximize the throughout on their "narrowest" relay hops. By doing so, traffic is routed away from the most congested links. Therefore, application level relays in fact help achieve network load balancing. At the network layer, traffic engineering also tries to balance workload on all links. Due to the cost of changing routing tables on routers, the time interval between network routing updates is on the scale of days. On the other hand, application relay paths can be updated much more frequently so that it can keep up with changes in network traffic. Along a relay path, relay nodes are responsible for the reliability of packets relayed by them. Lost packets, either due to congestion or corruption, are recovered locally. The *local recovery* eliminates unnecessary packet retransmissions on all links along an end-end path triggered by packet losses on just one link. This saving is significant especially when there are wireless links on the path [25]. Large feedback delay is a major hurdle for the stability and efficiency of congestion control schemes [31, 32, 33]. TCP connections on relay segments experience much smaller feedback delays and reacts faster than end-end connections. *Local rate control* is more efficient in obtaining

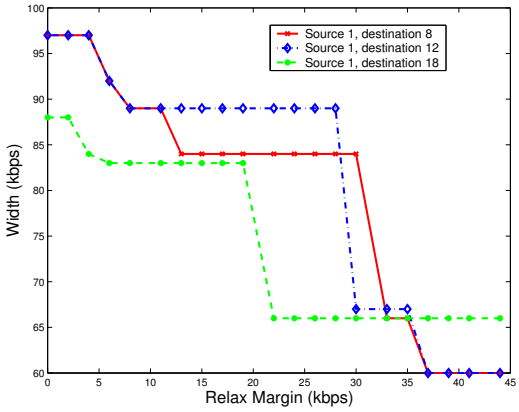


(a) Width of a shortest width-constrained path decreases as we relax the width constraint.

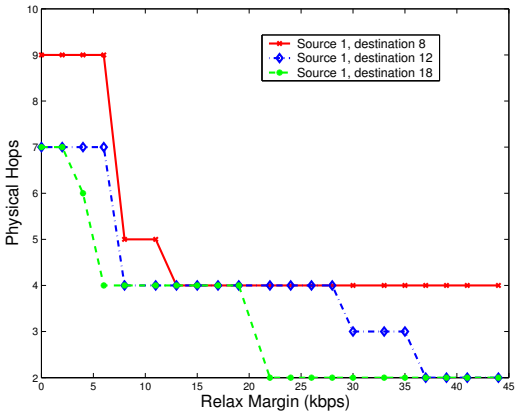


(b) Logical hop count of a shortest width-constrained path decreases as we relax the width constraint.

Figure 11: Shortest Width-constrained Path: logical hop count as the path length



(a) Width of a shortest width-constrained path decreases as we relax the width constraint.



(b) Physical hop count of a shortest width-constrained path decreases as we relax the width constraint.

Figure 12: Shortest Width-constrained Path: physical hop count as the path length

available bandwidth and at the same time helps to maintain network stability. Application level data relay in the store-and-forward manner naturally supports local caching of redundant data. For some data intensive applications, such as multicast and web server caching, an application relay can greatly reduce the amount the traffic crossing the network.

Application relay users can achieve better performance than regular users. They raise the fairness issues between different applications. However, it is *not fair* to criticize application relay users for being unfair to regular users just because they achieve higher throughput. As argued in [25], splitting a long connection into multiple smaller connections actually provides better fairness in a MANET. By moving traffic to less congested path, application relays in fact help to improve the performance of those regular users that have to use the congested path. Most previous studies of fair sharing of network resource focused on the transport layer and below [3, 34, 35, 36]. Little attention has been given to application layer fairness. Fairness issues of multi-path routing was investigated in [23]. Recently it has been studied in [17, 18] that Parallel TCP breaks, at the application layer, the fairness that TCP tries to maintain at the transport layer. The resources consumed by an application can be measured by the amount of *congestion* it generates in the network. Utility-price based framework was proposed to study the fairness and stability of congestion control schemes [35, 36, 37, 38]. Each transport layer connection has a utility function and incurs cost from all links along its path with congestion based price. The rate control problem is studied as a distributed optimization problem. This utility-price based approach can potentially be used to investigate the fairness issue at application layer. Since one application relay employs multiple transport layer connections to relay data, how to combine utility functions of those relay connections to study the application level fairness is a problem deserves further study. The goal of application layer fairness study is to guide the design of application level relays which trade-off efficiency with fairness among competing applications.

7 Conclusions and Future Works

In this paper, we investigate application level relay schemes for high-bandwidth data transport over shared wide area networks. We have shown that, by optimally combining TCP connections, both sequentially and in parallel, TCP pipelining greatly improves the throughput of long-haul data transport over lossy links. It also improves network efficiency and alleviates discrimination against long connections. The proposed optimal relay algorithms can be easily adopted in wireless/sensors and special purpose wired-line networks to relay data between nodes far away from each other. The optimal relay problem in overlay networks has been formulated as a multi-metric application level routing problem. Various algorithms have been investigated to optimize for multiple performance measures of individual users and the network as a whole. Proposed algorithms are readily to be implemented in operating overlay networks. We also discuss application level relays' impact on the fairness and efficiency of network resource allocation.

Future works can be pursued in several directions:

1. Extensive experiments have to be conducted in real network environment, such as the wide area overlay testbed PlanetLab [39], to test the performance of proposed relay schemes, and more importantly gain more understanding on the trade-offs between multiple performance metrics of relay paths.

2. The implementation of relay schemes in overlay networks remains to be studied. What is the right way to collect statistics regarding the underlay network? It was pointed in [40] that independent probes by overlay nodes generate considerable ping traffic. A *routing underlay* was proposed to collect information from the underlying Internet and answer the queries of overlay nodes.

Another question to be answered is when and how to compute optimal relay paths? Each node can calculate its optimal relay paths using distributed algorithms or a centralized unit is set up to do the optimization for all the nodes. Depending on the variability in the network condition, relay paths can be *pre-computed* and updated occasionally when the state of the network changes; or relay paths are calculated *on-demand*.

3. While the major objective of application level relay is to efficiently utilize network resource, the fairness issue at *application level* deserves more study. How to trade-off efficiency with fairness in application level relays is an interesting problem to look into.

References

[1] *Open Shortest Path First (OSPF)*, `http://www.ietf.org/html.charters/ospf-charter.html`.

[2] *Mulitprotocol Label Switching (MPLS)*, `http://www.ietf.org/html.charters/mpls-charter.html`.

[3] D.M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. 17, June 1989.

[4] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the internet," *IEEE/ACM Transactions on Networking*, August 1999.

[5] Sally Floyd, "Rfc 3649: Highspeed TCP for large congestion windows," Tech. Rep., IETF, 2003.

[6] Cheng Jin, David X. Wei, and Steven H. Low, "Fast tcp: motivation, architecture, algorithms, performance," in *Proceedings of IEEE/INFOCOM*, March 2004.

[7] Dina Katabi, Mark Handley, and Charlie Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proceedings of the ACM/SIGCOMM*, 2002.

[8] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan, "Detour: a case for informed internet routing and transport," Tech. Rep. TR-98-10-05, 1998.

[9] David Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris, "Resilient overlay networks," in *Proc. 18th ACM Symposium on Operating Systems Principles*, Banff, Canada, October 2001.

[10] Lakshminarayanan Subramanian, Ion Stoica, Hari Balakrishnan, and Randy H. Katz, "OverQoS: An overlay based architecture for enhancing internet QoS," in *Proceedings of HotNets-I*, October 2002.

[11] Suman Banerjee, Bobby Bhattacharjee, and Christopher Kommareddy, "Scalable application layer multicast," in *Proceedings of ACM/SIGCOMM*, 2002.

[12] Francois Baccelli, Augustin Chaintreau, Zhen Liu, Anton Riabov, and Sambit Sahu, "Scalability of reliable group communication using overlays," in *IEEE INFOCOM*, 2004.

[13] Gu-In Kwon and John W. Byers, "Roma: Reliable overlay multicast with loosely coupled tcp connections," in *IEEE INFOCOM*, 2004.

[14] Unknown, *FlashGet Web Page*, Amazesoft Company, 2001, `http://www.amazesoft.com/index.htm`.

[15] B. Cohen, *Bittorrent Web Site*, 2004, `http://bitconjurer.org/BitTorrent/`.

[16] Globus Alliance, *GridFtp Web Page*, 2004, `http://www.globus.org/datagrid/gridftp.html`.

[17] Y. Liu, W.B. Gong, and P. Shenoy, "On the impact of concurrent downloads," in *Proceedings of Winter Simulation Conference*, December 2001, pp. 1300–1305.

[18] Thomas J. Hacker, Brian D. Noble, and Brian D. Athey, "Improving throughput and maintaining fairness using parallel tcp," in *Proceedings of the IEEE INFOCOM*, 2004.

[19] Stefan Savage, Andy Collins, Eric Hoffman, John Snell, and Thomas E. Anderson, "The end-to-end effects of internet path selection," in *Proceedings of SIGCOMM*, Boston, MA, August-September 1999.

[20] Zheng Wang and Jon Crowcroft, "Quality-of-Service routing for supporting multimedia application," *IEEE Journal on Selected areas in communications*, vol. 14, no. 7, pp. 1228–1234, September 1994.

[21] Roch Guerin, Ariel Orda, and Douglas Williams, "QoS routing mechanisms and OSPF extensions," in *Proceedings of the second Global Internet Miniconference (joint with Globecom)*, 1997.

[22] G. Apostolopoulos, D. Williams, S. Kamat, R. Guerin, A. Orda, and T. Przygienda, "rfc2676: QoS routing mechanisms and OSPF extensions," Tech. Rep., IETF, 1999.

[23] Qingming Ma, Peter Steenkiste, and Hui Zhang, "Routing high-bandwidth traffic in max-min fair share networks," in *Proceedings of ACM/SIGCOMM*, 1996.

[24] Roch Guerin and Ariel Orda, "Computing shortest paths for any number of hops," *IEEE/ACM Transactions on Networking*, vol. 10, no. 5, pp. 613–620, 2002.

[25] Swastik Kopparty, Srikanth V. Krishnamurthy, Michalis Faloutsos, and Satish K. Tripathi, "Split tcp for mobile ad hoc networks," in *Symposium on Ad-Hoc Wireless Networks*, 2002.

[26] J.Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling tcp throughput: A simple model and its empirical," in *Proceedings of ACM/SIGCOMM*, 1998.

[27] Y. Liu and W.B. Gong, "Challenges to congestion control posed by concurrent downloads," in *Proceedings of 41th IEEE Conference on Decision and Control*, December 2002.

[28] M. R. Garey and D. S. Johnson, *Computers and Intractability*, San Francisco, CA: Freeman, 1979.

[29] P. Raghavan and C. D. Thompson, "Provably good routing in graphs: regular arrays," in *Proceedings of the 17th annual ACM Symposium on Theory of Computing*, 1985.

[30] Ken Calvert and Ellen Zegura, *GT Internetwork Topology Models (GT-ITM)*, `http://www.cc.gatech.edu/projects/gtitm/`.

[31] Laurent Massoulie, "Stability of distributed congestion control with heterogeneous feedback delays," *IEEE Transactions on Automatic Control*, vol. 47, no. 6, pp. 895–902, 2002.

[32] R. Johari and D. Tan, "End-to-end congestion control for the internet: Delays and stability," *IEEE/ACM Transactions on Networking*, December 2001.

[33] C.V. Hollot, V. Misra, D. Towsley, and W.B. Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *Proceedings of IEEE/INFOCOM*, 2001.

[34] J. M. Jaffe, "Bottleneck flow control," *IEEE Transactions on Communications*, vol. COM-29, no. 7, pp. 954–962, July 1981.

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

[36] M. Vojnovi'c, J. Boudec, and C. Boutremans, "Global fairness of Additive Increase and Multiplicative Decrease with heterogeneous roundtrip times," in *Proceedings of IEEE/INFOCOM*, 2000.

[37] S. Kunniyur and R. Srikant, "End-to-end congestion control schemes: Utility functions, random losses and ecn marks," in *Proceedings of IEEE INFOCOM'2000*, 2000, pp. 1323–1332.

[38] Steven H. Low, "A duality model of tcp and queue management algorithms," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, August 2003.

[39] *PlanetLab Web Page*, `http://www.planet-lab.org/`.

[40] Akihiro Nakao, Larry Peterson, and Andy Bavier, "A routing underlay for overlay networks," in *Proceedings of the ACM SIGCOMM*, Karlsruhe, Germany, august 2003.