

Stateless Load Balancing over Multiple MPLS Paths

B. Yang, C. Casetti, M. Gerla,
Computer Science Department - UCLA
Los Angeles, CA, 90024
{boyang,casetti,gerla}@cs.ucla.edu

Abstract—

The paper proposes a flow-independent approach to balance the load coming from several multimedia applications (i.e., IP Telephony) over multiple paths between a source and a destination. The proposed strategy exploits the advanced path-controlling and packet-forwarding features of the MPLS technology to set up the paths computed by the routing algorithm. Simulations reported in the paper show that the stateless approach increases the acceptance ratio (and thus the network utilization), while keeping the end-to-end delay and jitter within acceptable levels.

I. INTRODUCTION AND RELATED WORK

The general problem we will address in this paper is how to balance the load on an IP network using multiple paths over which packets are routed in a weighted round-robin fashion. Multiple-path load balancing has been the subject of several research works dating back to circuit-switched applications. However, the connectionless nature of the IP protocol, and the next-hop paradigm of IP routing have thwarted many attempts at applying the same techniques that were devised for connection-oriented networks. Although standard IP routing protocols such as OSPF support multipath routing through the computation of equal minimum-cost paths, the actual packet forwarding toward a destination is usually done on a single path. This choice is dictated by several concerns: packet resequencing, need to keep complex per-flow state, possible loops in the path computation.

The connectionless service model of the IP protocol does not guarantee that packets are received in the same order as they were initially forwarded. This problem is exacerbated if multiple paths are explicitly chosen, requiring that higher-level protocols take it upon themselves to resequence the packets. Packet resequencing can affect both connection-oriented data, such as TCP flows, and connectionless multimedia communications. In the case of TCP, too many out-of-sequence packets elicit the useless retransmission of packets following three duplicate ac-

knowledgements [8], adding to network congestion. In the case of multimedia flows, such as IP telephony, buffering techniques are required to smooth out the delay variations introduced by the different paths latencies and by the re-ordering overhead.

Although multipath information may be available to single routers, forwarding packets on multiple paths requires that the current next-hop approach be integrated with route-pinning capabilities. If connections are distributed over n paths at an upstream router, downstream routers are expected to make consistent routing choices: however, incoming IP packets do not carry path information, let alone any flow identification (at the IP level, at least), leaving fewer alternatives. Among them, source routing is the less appealing, due to its overhead and limitation on the maximum number of hops. In [1], Cao et al. have proposed traffic splitting algorithms that let routers hash the level 3 and 4 flow information on incoming packets to determine which path in the path set is to be selected, thus guaranteeing a consistent choice throughout the network, as well as per-flow packet ordering. A similar approach was followed in [9].

The route-pinning requirements, though, can be fulfilled using the explicit routing features of MPLS [10], [11]. Once multiple explicit paths (or Label-Switched Paths, LSPs, according to MPLS terminology) are established from an MPLS ingress router to an MPLS egress router, the traffic, i.e., from IP telephony applications, can be balanced across the path set. The advantages offered by MPLS are that, if the path set is carefully chosen so that every path in the set provides the same QoS guarantees, there is no need to keep per-flow information at the ingress router: incoming packets, regardless of the source, can be routed toward the destination egress router over each path in the set according to traffic shares assigned to each path.

Probably, one of the most important advantages of the flow-stateless mapping of voice connections to MPLS paths in the edge router is the ability to dynamically and transparently reconfigure the association between voice calls and paths without the need to maintain and modify

explicit maps or time out counters (as done in a "soft state" approach). This property comes in very handy when provisioning the voice service with a desired degree of redundancy. For example, for "hot standby" service, enough extra capacity is kept on alternate routed MPLS paths in order to permit instant recovery in case of trunk failure. With conventional per-flow mapping, the recovery requires the rerouting and even redistribution of connections from the failed path to the surviving paths – not an easy task if the edge router is supporting hundreds of thousands of calls to a given destination. With our proposed stateless association, this recovery is fully scalable in that it does not require any per-flow intervention whatsoever. Another obvious application where the stateless approach proves useful is the grooming of voice connections over multiple MPLS paths. As new paths are added or old paths are cleared in order to adjust to changing loads, the stateless scheme provides instant, even redistribution of the flows, again without per-flow intervention. Yet another benefit is the handling of mobility and handoffs. A user can dynamically move from one area to another, and thus be re-configured from one edge router to another, without need for extra signalling and map reconfiguration. In all, the stateless, packet-by-packet association between voice sessions and MPLS paths provides low latency, low overhead rerouting and greatly enhanced scalability.

The approach we propose extends the multiple-constraint QoS-routing algorithm described in [3] so that it can compute multiple QoS paths between a source and a destination. A novel path-selection strategy is also proposed.

The paper is organized as follows: Section II describes the overall architecture. The load balancing algorithms and path-selection strategies are discussed in Section III. Simulation results are presented in Section IV, and conclusions and hints on future work are in Section V.

II. ARCHITECTURE DESCRIPTION

The network architecture we have envisioned to support the implementation of our routing algorithms is similar to the one described in [5]. For the sake of simplicity, we will narrow our scope to an ISP-like network layout. We will refer to border routers (the routers connecting the ISP cloud to the outer network) as "ingress" or "egress" routers depending on the data flow, while we will refer to remaining routers as "intermediate". The IP network is supposed to be enhanced to support a simple IP integrated services framework. New IP phone call are set up through end-to-end RSVP messages that carry both the QoS requirements of the new call (such as the maximum tolerable end-to-end delay) as well as its traffic specifications (such as the av-

erage and peak bit rate of the call). When necessary to determine the QoS path for the new call, a router can use both the information in the RSVP message and link state information to perform admission control and proper routing.

Link state information in standard OSPF routing is flooded every 30 minutes [6], resulting in updates too far apart to allow reliable QoS routing. Therefore, we have implemented a threshold-based flooding policy, following the guidelines set forward in [9]. Every 15 seconds, each router checks the status of the links between itself and its immediate neighbors, computing the current load on those links, and determines which link needs to re-flood its own state. The flooding decision is based on three parameters: 1) the link load; 2) the change of the link load during the 15-second interval; 3) a timeout value. The larger the link equivalent load, the smaller the timeout is. Similarly, the larger the change of the link equivalent load during this 15 second interval, the smaller the timeout value is. The exact conditions are shown in Figure 1, in which $load$ is the current normalized load of the link, $diff$ is its change ratio during the latest period and $elapsed$ is the time in seconds since the last re-flood. To avoid any synchronized re-flooding behavior, a random jitter value between 0 and 5 seconds is inserted in the 15-second interval.

Each link state update is supposed to contain two pieces of information. For each outgoing link, a node advertises: 1) an estimate of the available bandwidth on that link; 2) the buffer occupancy in bytes at the outgoing interface (assuming each outgoing links has its own buffer). The latter comes in handy if the routing algorithm needs to compute end-to-end delays on a specific path.

III. ALGORITHMS FOR MULTIPATH QOS ROUTING

It is well known that the Bellman-Ford (BF) algorithm can potentially solve a two-metric routing problem in polynomial time, when one of the metrics is the hop count [4]. This fact favors the Bellman-Ford algorithm when compared with the Dijkstra algorithm, which lacks this capability, although the BF algorithm has a great complexity. This is because the Dijkstra algorithm does not search for a minimum path cost in ascending order of number of hops, as Bellman-Ford does. The use of a minimum hop count metric is of particular interest since it is likely to provide the lowest session rejection probability for a given network load, if no knowledge about the distribution of connections is assumed. The BF algorithm has the ability to solve multiple constrained routing problems, as further investigated in [3].

```

if (( (load > 1.00) & (diff > 0.05) & (elapsed >= 30)) ||
    ((load > 1.00) & (diff > 0.02) & (elapsed >= 60)) ||
    ((load > 1.00) & (diff > 0.01) & (elapsed >= 90)) ||
    ((load > 1.00) & (elapsed >= 180)) ||
    ((load > 0.90) & (diff > 0.05) & (elapsed >= 60)) ||
    ((load > 0.90) & (diff > 0.02) & (elapsed >= 240)) ||
    ((load > 0.90) & (diff > 0.01) & (elapsed >= 480)) ||
    ((load > 0.90) & (elapsed >= 600)) ||
    ((load > 0.70) & (diff > 0.10) & (elapsed >= 60)) ||
    ((load > 0.70) & (diff > 0.05) & (elapsed >= 120)) ||
    ((load > 0.70) & (diff > 0.02) & (elapsed >= 480)) ||
    ((load > 0.70) & (elapsed >= 900)) ||
    ((load > 0.50) & (diff > 0.10) & (elapsed >= 60)) ||
    ((load > 0.50) & (diff > 0.05) & (elapsed >= 300)) ||
    ((load > 0.25) & (diff > 0.25) & (elapsed >= 120)) ||
    ((load > 0.25) & (elapsed >= 1200)))
{
    refflood
}
    
```

Fig. 1. Pseudo-code for the refflooding algorithm

A. A single-path multiple-constraint routing algorithm

Given a network $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where \mathcal{V} is the set of vertices, or nodes, and \mathcal{E} is the set of edges, or links, a general QoS routing problem consists in finding a path $P(s, t)$ from source $s \in \mathcal{V}$ to destination $t \in \mathcal{V}$, which is suitable for applications with given end-to-end constraints. The constraints usually considered in IP telephony applications are delay and bandwidth. In particular, we are interested in the “best” path which satisfies the application end-to-end bandwidth and latency requirements.

“Best” means that such path should use the minimum amount of network resources. Therefore, an optimal path is generally defined as a path with minimum number of hops which still satisfies the bandwidth and latency constraints.

The Bellman-Ford (BF) routing algorithm computes shortest paths between a given source s and a set of destinations. It does not include multiple metrics. In [3], the BF algorithm is extended so as to handle multiple constraints such as delay and bandwidth. In particular, it is shown that the multiple-constraint BF algorithm can compute the minimum-hop path among all paths which satisfy a specific delay bound D . More generally, letting d be an additive cost metric (of which the delay is an example), the minimum-hop path with cost $d \leq D$ is found when for first time during the iterative execution of the BF algorithm the cost drops below D . This feature comes in handy when dealing with routing problems with multiple constraints, one of which is additive.

As for the bandwidth constraint, it is easily dealt with in an obvious way, i.e., by pruning the links without adequate bandwidth. A path is therefore acceptable if it satisfies the end-to-end delay constraint, and if, after accepting the new call, it has residual bandwidth $\geq BR$. The choice of the *bandwidth margin* BR is dictated by the need to guarantee adequate performance to the new coming connection and at the same time protect the performance of ongoing connections.

If the routing procedure does not succeed in finding enough resources to guarantee the QoS requirements both of the incoming call as well as of the calls already accepted on the network, the Call Admission Control rejects the call. More details on the Multiple-Constraints Bellman-Ford (MC-BF) routing scheme, including the use of other metrics of interest, such as packet loss and jitter, can be found in [3].

B. A Multiple-path extension of MC-BF

The MC-BF yields only a single minimum-hop, delay-constrained path. For our load-balancing purposes, we need to provide more than one such path, and it is therefore necessary to explore multipath capabilities of the MC-BF. Although a variety of solutions exist, we chose a simple algorithm that combines the MC-BF properties with a merge-sort approach.

In the following, we will indicate by $\mathcal{S}_{s,v}^h$ a set containing all paths from node s to node v with an hop count of h (where $h \leq H_{max} \leq |\mathcal{V}|$). We will assume that the paths in $\mathcal{S}_{s,v}^h$ are sorted according to their increasing estimated end-to-end delays (computed from the output buffer occupancy advertised through the link state updates).

The key idea is to compute, for each node v and for each hop number h , a subset $\mathcal{S}_{s,v}^h(m) \subseteq \mathcal{S}_{s,v}^h$ comprising at most the m paths that exhibit the lowest end-to-end delay among all paths in $\mathcal{S}_{s,v}^h$ and that, at the same time, satisfy the constraint $d \leq D$ (and the bandwidth constraint), as is done in the MC-BF case. We will refer to $\mathcal{S}_{s,v}^h(m)$ as “candidate path set for node s to node v in h hops”. It follows that H_{max} candidate path sets are kept at each node.

At the onset, all candidate path sets are initialized as empty sets. Then, for increasing hop count ($1 < h < H_{max}$), the set of neighboring nodes of node v (denoted by $\mathcal{N}(v)$) is scanned. For each $u \in \mathcal{N}(v)$, it concatenates the link $v \rightarrow u$ to each path in $\mathcal{S}_{s,u}^{h-1}(m)$ (i.e., the candidate path set for s to node u in $h-1$ hops), provided that the link $v \rightarrow u$ does not already belong to the path, and that v is not reached by that path (to avoid loops). The set of paths resulting from this concatenation and the current candidate path set $\mathcal{S}_{s,v}^h(m)$ are merged and sorted in increasing end-to-end delays and at most the first m paths

```

PROCEDURE ComputePathsFromSrcToDst( $s, t$ )
  for hops  $h = 1$  to  $H_{max}$  {
    for each node  $v$  in the network {
      for each node  $u, u \in \mathcal{N}(v), \{$ 
         $\mathcal{S}_{s,v}^h(m) = \text{merge-sort}(\mathcal{S}_{s,v}^h(m), \mathcal{S}_{s,u}^{h-1}(m))$ 
      }
    }
    if ( ( $v = t$ ) AND ( $\mathcal{S}_{s,v}^h(m)$  contains at least one path) )
      return  $\mathcal{S}_{s,v}^h(m)$ 
  }
End PROCEDURE

```

Fig. 2. Pseudo-code for the computation of the Candidate Path Set for a generic node v

are selected to become the *new* candidate path set for node s to node v in h hops. Notice that the set $\mathcal{S}_{s,u}^{h-1}(m)$ was either updated in the previous round, if $h > 1$, or it was empty, if $h = 1$.

The algorithm to compute the candidate path sets is summarized by the pseudocode in Figure 2.

Whenever a multiple paths are needed to a destination reachable through egress router t , the ingress router s executes the procedure `ComputePathsFromSrcToDst(s, t)`. By running the algorithm for increasing hop count, the final path computation is guaranteed to be also the one with the smallest possible number of hops.

In addition to the multi-path extension of MC-BF mentioned here, the path cost relaxation techniques mentioned in [9] can also be used to compute multiple paths to a destination. The latter approach can help reduce the computation overhead and will be studied in future work.

We will now outline two path-selection algorithms to route packets over multiple QoS paths. Both use the multipath version of MC-BF to compute feasible paths.

C. Flow-based Multipath

In this rather simplistic scheme, a path set containing at most M multiple paths is associated to each flow. This path set is computed during the call setup time using the procedure outlined in Subsection III-B and stored in the per-flow entry for this flow at the border router. When a voice packet arrives at the ingress router, its flow ID is determined first. If there is only one path in the path set associated to that flow, (i.e., only one path was found at setup), all packets of the flow are routed over that path. If there is more than one path in the path set, the packets (belonging to the same flow) are evenly distributed over those paths in a round-robin fashion.

Unquestionably, the Flow-based Multipath algorithm is ill-suited to handle a larger number of connections, due to the need to store a different path set for each ongoing connection. In this paper, it is only used as a term of comparison with the packet-based Stateless Multipath algorithm described next and no actual implementation is detailed.

D. Stateless Multipath

The Stateless Multipath algorithm was designed to overcome the limitations of the Flow-based Multipath. Although the path computation process is carried out using the multipath MC-BF algorithm, the Stateless Multipath differs from the Flow-based version under several important respects:

- the path set is an attribute of the ingress-egress router pair, rather than of the single flow;
- the load distribution is accomplished with packet-level granularity, rather than flow-level granularity;
- dynamic load balancing over the path set is made possible.

In this scheme, each ingress-egress router pair is assigned a path set of at most M paths over which flows with similar traffic specifications (i.e., bandwidth and end-to-end requirements) are routed. In this study, we have only considered a single traffic class per ingress-egress router pair, however the extension to multiple classes is possible with minor modifications and little additional complexity. Flows between the same ingress-egress router pair share the same path set and their packets are distributed over each path of the path set in a flow-independent fashion.

When a new flow entering at ingress router R_s and directed to a destination reachable through egress router R_t requests admission, the ingress router first checks whether there exists a path set between R_s and R_t . If no path set already exists, a new one, containing at most M QoS-aware paths, is computed following the procedure in Subsection III-B. If the path set was already computed, a standard MC-BF algorithm is run, in order to identify the current best QoS path from R_s to R_t . The new path is compared with the existing path set and, if it is new, it is added to the path set and an offloading procedure is initiated. Such procedure aims at gradually shifting the aggregate load (existing flows and new flow) onto the new member of the path set, until a new load balance is found over all paths in the set. The "steady-state" balance does not necessarily entail an evenly-split load. The details of the offloading procedure were taken from in [9].

The load of each path in the same path set (the traffic share) is also adjusted dynamically. This dynamic load balancing is done periodically. Two events can trigger an adjustment:

- Right after a router receives link state advertisement about a link, it checks to see if this link is the bottleneck of one or more of its path sets in use. If so, the traffic share for the path set that has this link as its bottleneck is adjusted. Otherwise, no adjustment is done.
- Periodically, a router scans the path sets in use and decides whether an adjustment of traffic shares should be performed for any of the path sets. The decision is based on the time elapsed since the last adjustment, the congestion level of the path set and the difference between the utilization of the most congested path and the idlest path.

The implementation of the adjustment strategy follows the detailed description in [9].

Underloaded paths (i.e., paths whose bottleneck link carries fewer traffic than a percentage of its capacity) should be removed from the path set and their load evenly split among the remaining paths.

As pointed out earlier, the establishment of multiple paths between an ingress router and an egress router can be accomplished using MPLS. Since MPLS is basically a forwarding technique, it is up to the routing algorithm to compute the feasible paths and select those that satisfy the QoS requirements. Once those paths are determined at the ingress router, the task of MPLS is that of setting up Label Switched Paths (LSPs) corresponding to the paths determined by the routing algorithm. These "explicit routes" allow packets to follow precise trajectories across the MPLS network, to an egress router. Upon receiving a packet, ingress router R_s looks up the egress router (say R_t) that is supposed to handle traffic for the packet's IP destination address *outside* the MPLS network. Next, the ingress router identifies the set of LSPs that are associated to R_t and are currently being used to carry packets from all connections entering at R_s and exiting the MPLS network through R_t . The packet is then forwarded on one of the LSPs of the set, according to the load-splitting ratio currently enforced at the R_s . To reduce complexity and maintain the stateless approach, a simple weighted round robin procedure can be used.

IV. SIMULATION RESULTS

A. Network and Traffic Models

In this section we describe the various models and features implemented in our simulator in order to recreate a realistic network environment. Both QoS traffic (voice) and signaling (OSPF messages) were handled using the same priority. We chose a highly connected topology, which offers potential for bottlenecks and provides several alternate routes (Fig. 3). Propagation delay varies from link to link, reflecting a regional ISP network scenario.

Router queues have a 100-Kbyte buffer and each link has a 15 Mb/s capacity. Propagation delays vary from link to link, averaging around 1ms per link, thus reflecting a regional ISP network scenario. The bandwidth margin BR on accepted calls is 0.5 Mb/s, a choice which was made according to the findings in [2].

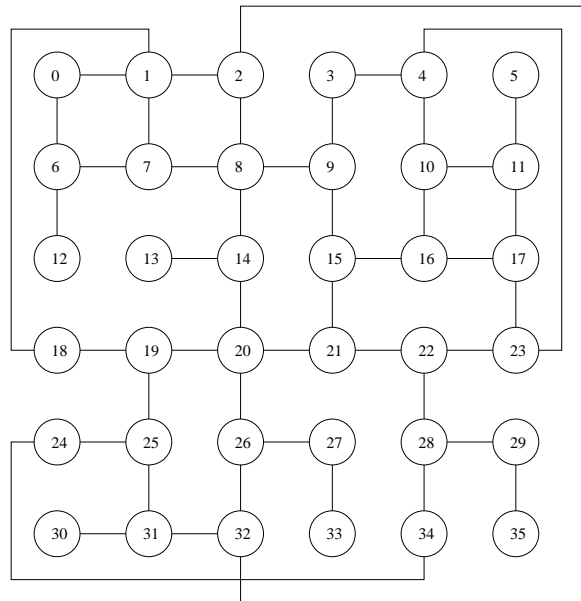


Fig. 3. Network Topology

New voice connection requests arrive according to a Poisson process; once a connection is accepted and established, the voice source is modeled as a two-state On-Off Markov chain. The "On" (talk spurt) state duration is an exponentially distributed random variable with average equal to 352 ms. The "Off" (silence) state is also exponentially distributed with average equal to 650 ms. Each voice call uses PCM encoding, transmitting data at speeds that range between 8 and 64 kb/s in talk spurt state, as will be detailed later on. The call duration is exponentially distributed with an average of 180 seconds.

New voice connection requests are generated at random between a predefined set of source and destination pairs. These nodes act as the ingress and egress routers. The candidate source/destination pairs are: (0,34), (3,32), (4,32), (5,32), (8,20), (10,32), (16,32). Each experiment duration is 10 minutes of simulated time. The first 3 minutes of the simulation experiment are not considered in the collection of results in order to allow the system to reach steady state. The QoS constraints for the voice sources are 100 ms end-to-end delay and a minimum residual bandwidth $BR = 0.5$ Mb/s on each link.

In the following, we will compare the performance of the two Multipath strategies outlined in Section III, and the "MC-BF single-path" approach. The latter routing strate-

gies uses the MC-BF algorithm to compute a QoS-aware single path between a source and a destination when a new call requests admission. The incoming call is routed over that path for the entire duration of the call. Clearly, the path choice is affected by changing traffic conditions and a different path might be selected if congestion arises.

B. Homogeneous traffic

In the first set of experiments, every source node requests 56-kb/s voice calls. The voice stream is packetized into 180-byte UDP packets, with a 160-byte payload and a 20-byte UDP+RTP overhead. Results are shown for decreasing values of the call interarrival times, which translate into increasing traffic load.

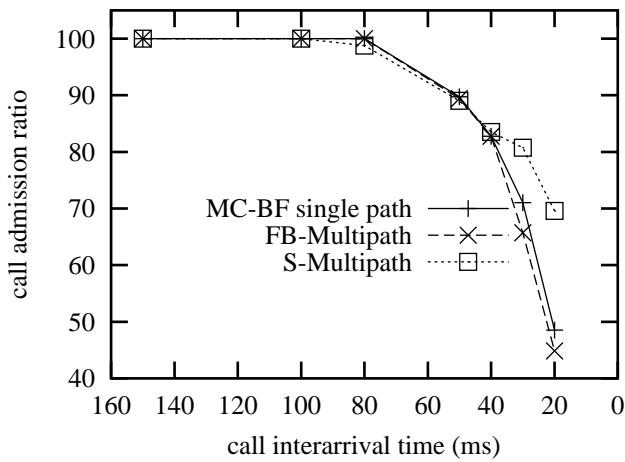


Fig. 4. Call admission percentage vs. offered load - homogeneous traffic

The call admission ratio is key to comparing the effectiveness of the algorithms in balancing the load over the network. As can be seen in Figure 4, the admission ratio for the three algorithms is similar at light and medium load, and diverges as the load increases. In particular, the Stateless Multipath (*S-Multipath* in the plots) outperforms both the Flow-based multipath (*FB-Multipath*) and the MC-BF single-path algorithms. It is a tenet of routing theory (see for example [7]) that, under heavy traffic load, choosing paths that stray from the shortest path leads to excessive resource consumptions on alternate paths. This observation explains why the Flow-based Multipath performs so poorly when compared to the MC-BF single-path for small call interarrival times. The Stateless Multipath, conversely, benefits from its ability to disperse the load from multiple connections in a seamless way across several paths, achieving a finer load balancing granularity. Moreover, by inspecting the load distributions we have observed that the Stateless Multipath reduces the number of

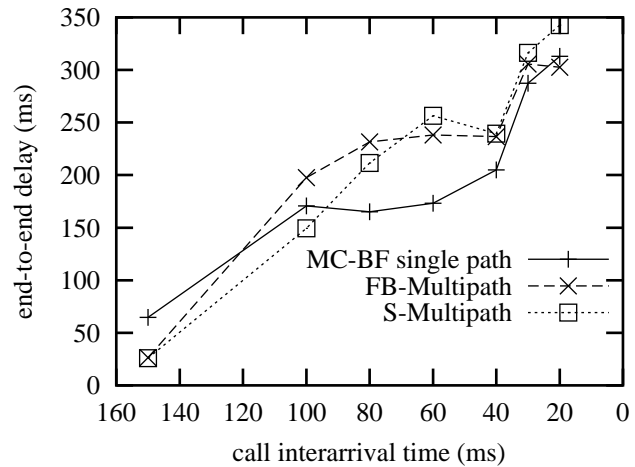


Fig. 5. Average end-to-end delay vs. offered load - homogeneous traffic

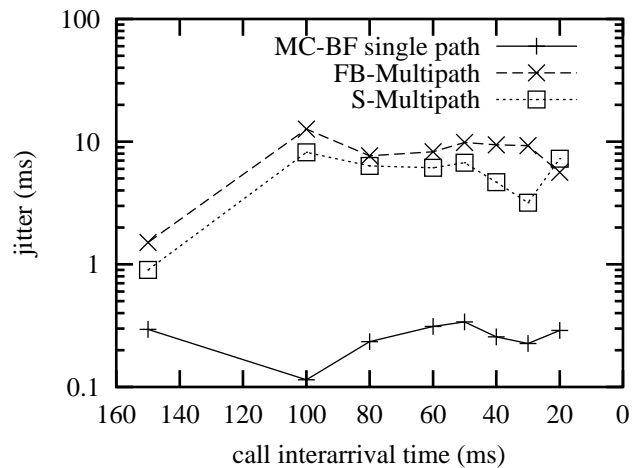


Fig. 6. Delay jitter vs. offered load - homogeneous traffic

links that have extremely high link utilizations (in other words, it has fewer highly congested links than the MC-BF single-path case and the Flow-based Multipath case).

In order to be a meaningful performance index in a multipath scenario, the average end-to-end delay needs to take into account the resequencing delay. Being scattered across different paths, packets can reach their destinations out of sequence, and it is necessary to wait for the missing intermediate packets before they can be played back. In Figure 5, we computed the end-to-end delay as the time between the packet transmission and the time it is *available* for playback, meaning that the packets with lower sequence numbers have either been correctly delivered, or declared "lost". In our simulations, in order to avoid increasing delays at the receiver, a threshold of 400 ms was imposed before a missing packet was declared "lost". Although the MC-BF single-path algorithm is not affected

by resequencing delays, its use results in longer queuing delays than the Multipath algorithms. It can be seen that, in comparison, the two delay components offset each other both at low and high loads, while at medium loads the single-path delays are noticeably smaller.

Dispersing packets over multiple paths is known to negatively affect the delay variance, which in turn has a serious impact on the perceptual quality of audio-visual streams. The estimate of the jitter, shown in Figure 6, is extremely low, as expected, for the single-path algorithm, and it is almost two orders of magnitude bigger for the Multipath algorithms (and still not exceedingly high at almost 10 ms). Also, the Stateless Multipath provides a better performance than the Flow-based Multipath.

C. Heterogeneous traffic

In the second set of experiments, every source node generates voice calls with different bandwidth requirements. Table I details the call rate distribution among source nodes. The traffic pattern was chosen so that the cluster of source nodes formed by nodes 3, 4 and 5, all sending to node 32 over potentially interfering paths, loaded the neighboring links with traffic exhibiting a different granularity (8 and 64 kb/s).

TABLE I
CALL BANDWIDTH DISTRIBUTION - HETEROGENEOUS
PEAK RATES

| Source Node | Call bandwidth |
|-------------|----------------|
| 0 | 64 kb/s |
| 3 | 8 kb/s |
| 4 | 64 kb/s |
| 5 | 8 kb/s |
| 8 | 8 kb/s |
| 10 | 64 kb/s |
| 16 | 64 kb/s |

The results, shown in Figures 7, 8 and 9, are not dissimilar from the homogeneous-traffic case, showing that the performance underscored above hold even with changing traffic conditions.

V. CONCLUSIONS

The usefulness of MPLS traffic engineering has been recognized since it was first proposed as a link-layer forwarding technique to replace the next-hop paradigm typical of IP networks. In this paper, we proposed a load balancing strategy called Stateless Multipath, that exploits MPLS features to route packets over multiple paths. The load balancing strategy is an extension of the so-called

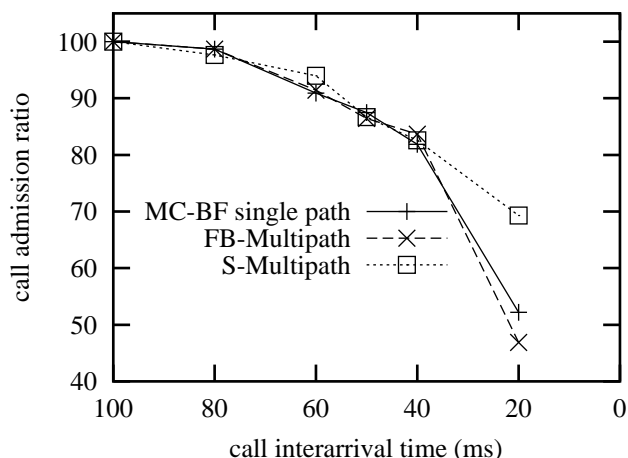


Fig. 7. Call admission percentage vs. offered load - heterogeneous traffic

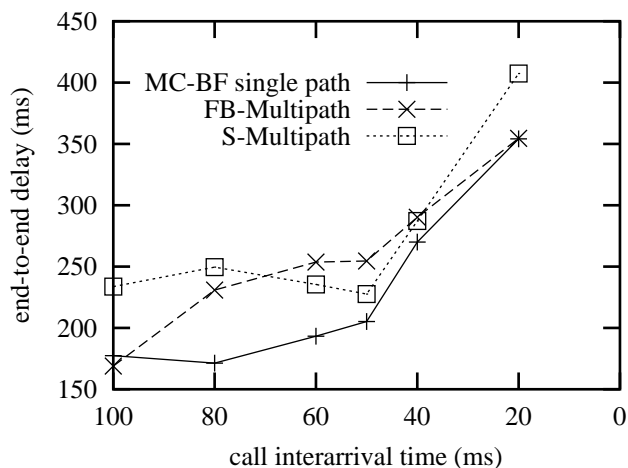


Fig. 8. Average end-to-end delay vs. offered load - heterogeneous traffic

Multiple-Constraint Bellman-Ford algorithm, and it identifies multiple paths between a source and a destination that can satisfy specific QoS requirements. This property makes the Stateless Multipath suitable for all those tasks where a soft-state reconfiguration of the association between calls and paths is needed.

Several aspects of the Stateless Multipath need further investigation, namely the determination of "optimum" traffic ratios among the paths computed by the routing algorithm, and how the dynamic adjustment of those ratios affects the traffic.

REFERENCES

- [1] Z. Cao, Z. Wang, and E. Zegura. Performance of Hashing-based Schemes for Internet Load Balancing. In *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, Israel, March 2000.
- [2] D. Cavendish, A. Dubrovsky, M. Gerla, G. Reali, and S.S. Lee.

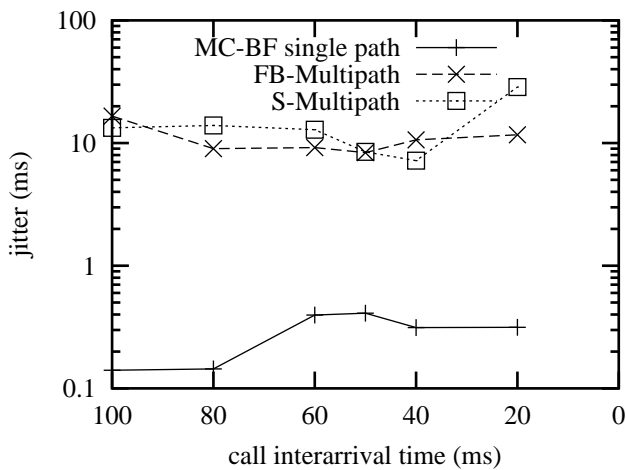


Fig. 9. Delay jitter vs. offered load - heterogeneous traffic

- Statistical Internet QoS Guarantees for IP Telephony. Technical Report CSD-990053, Computer Science Dept., UCLA, 1999.
- [3] D. Cavendish and M. Gerla. Internet QoS Routing Using the Bellman-Ford Algorithm. In *Proceedings of the Conference on High Performance Networking (HPN98)*, Austria, 1998. IFIP.
 - [4] R. Guerin, A. Orda, and D. Williams. QoS Routing Mechanisms and OSPF Extensions. In *Proceedings of IEEE GLOBECOM97*, Phoenix, AZ, USA, 1997.
 - [5] P.P. Mishra and H. Saran. Capacity Management and Routing Policies for Voice over IP Traffic. *IEEE Network Magazine*, pages 20–27, March 2000.
 - [6] J. Moy. OSPF, version 2. RFC 2328, April 1998.
 - [7] S. Sibal and A. Desimone. Controlling Alternate Routing in General-Mesh Packet Flow Networks. In *Proceedings of the ACM SIGCOMM'95*, pages 115–126, Cambridge, MA, USA, August 1995.
 - [8] W.R. Stevens. *TCP/IP Illustrated, vol. 1*. Addison Wesley, Reading, MA, USA, 1994.
 - [9] C. Villamizar. OSPF Optimized Multipath (OSPF-OMP). Internet-Draft, April 1999. Work in progress.
 - [10] A. Viswanathan, N. Feldman, Z. Wang, and R. Callon. Evolution of Multiprotocol Label Switching. *IEEE Communications Magazine*, pages 165–173, May 1998.
 - [11] X. Xiao, A. Hannan, B. Bailey, and L. Ni. Traffic Engineering with MPLS in the Internet. *IEEE Network Magazine*, pages 28–33, March 2000.