Generalized Load Sharing for Packet-Switching Networks

Ka-Cheong Leung and Victor O. K. Li
Department of Electrical and Electronic Engineering
The University of Hong Kong
Pokfulam Road, Hong Kong, China
{kcleung, vli}@eee.hku.hk

Abstract

In this paper, we propose a framework to study how to effectively perform load sharing in multipath communication networks. A generalized load sharing (GLS) model has been developed to conceptualize how traffic is split ideally on a set of active paths. A simple traffic splitting algorithm, called weighted fair routing (WFR), has been developed at two different granularity levels, namely, the packet level, and the call level, to approximate GLS with the given routing weight vector. The packet-by-packet WFR (PWFR) mimics GLS by transmitting each packet as a whole, whereas the call-by-call WFR (CWFR) imitates GLS so that all packets belonging to a single flow are sent on the same path. We have developed some performance bounds for PWFR and found that PWFR is a deterministically fair traffic splitting algorithm. This attractive property is useful in the provision of service with guaranteed performance when multiple paths can be used simultaneously to transmit packets which belong to the same flow. Our simulation studies, based on a collection of Internet backbone traces, reveal that WFR outperforms two other traffic splitting algorithms, namely, generalized round robin routing (GRR), and probabilistic routing (PRR). These promising results form a basis for designing future adaptive constraint-based multipath routing protocols.

1. Introduction

The convergence of the computer, communications, entertainment, and consumer electronics industry is driving an explosive growth in multimedia applications [18]. Nowadays, the Internet provides a convenient and cost-effective communication platform for electronic commerce, collaboration on research and development, education, and entertainment [3]. With the fulminating development of the Internet, Internet service providers (ISPs) face the challenge on how to effectively provision a variety of quality of service (QOS) guarantees for a large number of concurrent (connection-oriented or connectionless) communication activities.

One way for the ISPs to respond to this challenge is by capacity expansion. They can upgrade their backbones by replacing existing links with larger capacities. Besides, they can install new links in parallel with the existing ones to reduce the upgrade costs. If the second approach is used, there must be an effective mechanism to efficiently route or to distribute traffic to a set of parallel links.

Besides capacity expansion, traffic engineering is a powerful tool to overcome this challenge. Internet traffic engineering facilitates efficient and reliable network operations, and optimizes operational network utilization and performance, by the application of technological and scientific knowledge to the measurement, modelling, characterization, and control of Internet traffic, to achieve specific QOS requirements [2]. The popular routing approach for the Internet consists of finding a single shortest path from a source to a destination based on some link cost metrics, which are updated periodically. Although such unipath routing protocols can adapt very quickly to changing network conditions, they become unstable under heavy loads or bursty traffic as the link cost metrics used in the routing algorithms are related to delays or congestion experienced over the network links [4, 28]. Moreover, at a certain time instant, some subnets can be heavily congested, while other subnets along alternate paths are under-utilized. Thus, traffic engineering should be employed to balance the use of network resources, say link bandwidths. This advocates the use of multiple paths simultaneously for data transmission.

Load sharing is a channel aggregation approach that permits data traffic to be dispersed on multiple channels at the same time to reduce network load fluctuation. Given a set of active paths connecting a source to a destination, the key to multipath routing involves answering the following three basic questions:

- At which granularity should multipath routing be applied, i.e. at the byte level [9, 11], packet level [1, 6, 7, 13, 14, 15, 19, 20, 21], call level [5, 27], or otherwise?
- What is the optimal split of traffic along a set of active paths to achieve the best performance?
- How does one implement the traffic splitter? Or, given the optimal traffic split, what is the best way to distribute traffic?

As far as the allocation granularity for multipath routing is concerned, a finer allocation granularity, say at the byte level, gives a better balance on the use of network resources, but a stricter synchronization requirement is needed. For packet-switched networks, like the Internet, the smallest data switching unit is a packet. Moreover, both connection-oriented and connectionless traffics are allowed to co-exist, with a strict packet sequencing maintained within a connection-oriented call, such as a Transmission Control Protocol (TCP) connection. Thus, either packet level or call level multipath routing is the most suitable for packet-switched networks.

The determination of the optimal traffic split can be formulated as solving an optimal routing problem [4, 10, 12] with a specific objective function, such as the minimization of the expected delay within a network. An approximate iterative algorithm [28] has been suggested to improve the responsiveness and convergence of the routing solution. Besides, a heuristic based on loading and packet loss rate along a set of loop-free progressive paths[1] under the Open Shortest Path First (OSPF) protocol has been introduced in [27].

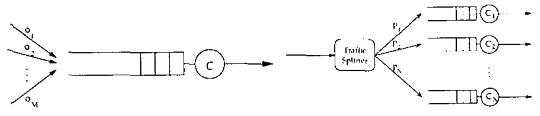
The most common form of traffic splitting distributes packets to a set of active paths in a round robin fashion [14] or its variants [1], or dispenses bursts of packets to all participating paths in a cyclic manner [7, 15]. Such algorithms are quite simple to implement. However, it can only support uniform traffic splitting or cyclic dispersion on these active paths. With heterogeneous paths, the best way to spread traffic along multiple paths may not be by cyclic dispersion, since it may not achieve the objective, such as minimizing the end-to-end path delay. In addition, it may induce substantial packet resequencing delay when the end-to-end delays are quite different among these paths. Such limitations can be alleviated by using a splitter which routes packets in a generalized round robin fashion, according to the given traffic split. Nevertheless, these routing approaches implement the desired traffic split in terms of the number of packets being routed, rather than the actual workload in bytes. Since packets are generally of different sizes in packet-switched networks, except Asynchronous Transfer Mode (ATM) networks, the actual workloads to paths may deviate unboundedly from the expected workloads.

Since a large volume of traffic in the Internet is TCP-based, an orderly delivery of packets within the same flow to a destination is critical to avoid triggering "false losses", resulting in a substantial degradation in protocol performance. Thus, all packets within the same flow should be delivered on the same path. Hashing-based traffic splitting approaches [5, 27] have been proposed so that all packets with the same key, such as the same origin-destination (O-D) pair, will be routed on the same path. These techniques are usually simple to implement and scalable in terms of the number of active paths and flows. However, the accuracy in implementing the requested traffic split depends greatly on the choices of both keys and hashing functions. Moreover, they cannot take the granularity of flows into account, as connection-oriented calls ordinarily demand different bandwidths and QOS requirements and they should not be treated equally.

The focus of this work is to propose a framework to study how to effectively perform load sharing in multipath communication networks. Section 2 develops a generalized load sharing model to conceptualize how traffic is split on a set of active paths. Section 3 presents both packetized and flow-based weighted fair routing algorithms to achieve the best load balancing according to the given traffic split. Section 4 outlines a set of traffic splitting algorithms for performance studies, defines the performance metric that will be used to compare various splitting algorithms, briefly describes the characteristics of the traffic traces used for the simulation, and examines the simulation results of the proposed traffic splitting algorithms. Section 5 concludes and discusses some possible extensions to our work. Lemma and theorem proofs are omitted here because of constraints in space. Interested readers are requested to contact the authors for the proofs.

[1] A loop-free progressive path is one in which any next hop is "smaller" in terms of cost than the current hop.

2. The Generalized Load Sharing Model



(a) Processor sharing. (b) Load sharing.

Figure 1. The roles of processor sharing and load sharing.

In [1, 5, 26], it has been observed that there is a close relationship between link sharing and load sharing. Processor sharing is a channel multiplexing technique that allows different traffic flows to share the same channel or link simultaneously to improve link utilization, whereas load sharing is a channel aggregation approach that permits data traffic to be dispersed on multiple channels at the same time to reduce network load fluctuation. The roles of these two techniques are diagrammed in Figure 1.

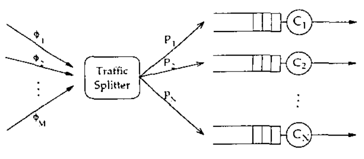


Figure 2. A generalized sharing model.

Both techniques are aimed at the efficient use of network resources to minimize network congestion and they can be applied in a complementary manner. For example, a set of traffic flows can be multiplexed to share a set of outgoing links, as shown in Figure 2.

To consider a fair allocation of network resources, a generalized processor sharing (GPS) approach [23, 24] has been proposed for link sharing. With GPS, each flow session α is assigned a fair share weight ϕ_α for sharing an outgoing network link with capacity C, such that, at any given time, the service rate of a backlogged session α, r_α, is given as:

$$r_\alpha = \frac{\phi_\alpha}{\sum_{i \in \mathcal{B}} \phi_i} C$$

where $\mathcal{B}$ is a set of backlogged sessions during that time.

A methodology that is similar to GPS can be applied for load sharing. Our load sharing model consists of a network node and a set of N (logical) outgoing links, namely, Link 1, Link 2, ..., Link N, as shown in Figure 1(b). Without loss of generality, we assume that there is a single class of traffic for traffic splitting. In other words, for ease of presentation, all incoming traffic to the captioned network node will be treated by the same traffic splitter, installed with a routing weight vector. Indeed, it is simple to generalize to cases with multiple classes, each of which can correspond to a destination with the same quality of service (QOS) class. Incoming traffic is fed into a traffic splitter for load sharing, according to its traffic class.

When the node receives an incoming packet destined to another node, a traffic splitter installed in the network node is invoked to decide which outgoing link the packet is forwarded to. Once an assignment is made, the packet will be dispatched to an output queue of the corresponding link to wait for transmission. Suppose p_i is the routing weight for Link i. It denotes the portion of dispersed traffic to be routed on Link i, where $\sum_{i=1}^{N} p_i = 1$. Denote the routing weight vector by $\mathbf{p} = (p_1 \quad p_2 \quad \ldots \quad p_N)$. Given the routing weight vector $\mathbf{p}$, the quality of a traffic splitter depends on how close to $\mathbf{p}$ the actual load distribution is at all times.

Let $L_i(\tau, t)$ be the amount of traffic (in bytes) prepared to be sent on Link i during the time interval $(\tau, t]$. Since departing traffic will be routed on one of the outgoing links, the total amount of incoming traffic to be forwarded to the destination during the time interval $(\tau, t]$, $T(\tau, t)$, is equal to the sum of all traffic scheduled to be routed on each outgoing link to the same destination during the same time interval. That is,

$$T(\tau, t) = \sum_{i=1}^{N} L_i(\tau, t)$$

for any period $(\tau, t]$.

A generalized load sharing (GLS) traffic splitter is one that can divide traffic to the set of outgoing links *exactly* according to the given routing weight vector $\mathbf{p}$. Thus, for any period $(\tau, t]$, a GLS traffic splitter[2] is defined [5] as one for which:

$$L_i(\tau, t) = p_i T(\tau, t)$$

or

$$\frac{L_i(\tau, t)}{L_j(\tau, t)} = \frac{p_i}{p_j}$$

for every $i, j = 1, 2, \ldots, N$.

If a traffic source is constrained by a leaky bucket, the following lemma states that all dispersed flows split by a GLS traffic splitter are also leaky bucket constrained.

Lemma 1 *Suppose a traffic source is constrained by a leaky bucket with parameters (σ, ρ), where σ and ρ represent the maximum size of the bucket token pool and the average token generation rate respectively. If it is fed into a GLS traffic splitter, a dispersed flow to be routed on Link i, where $i = 1, 2, \ldots, N$, is constrained by another leaky bucket with parameters $(p_i\,\sigma, p_i\,\rho)$.*

To summarize, there are two major attractive features for GLS. First, it distributes traffic *exactly* according to a set of prescribed routing weights. It is always fair deterministically, where the actual traffic routed to any link is equal to the expected one, for all time intervals. Second, if a GPS server works with a GLS traffic splitter, it is possible to make worst-case network queueing delay guarantees when the flows are constrained by leaky buckets. It follows directly from [23] that the worst-case network queueing delay of a leaky bucket constrained source is bounded under a GPS system.

[2]In [5], such kind of ideal load balancing is termed *ideal traffic splitting*.

3. Weighted Fair Routing

Like the generalized processor sharing (GPS) approach, the generalized load sharing (GLS) approach is an idealized scheme that assumes input traffic to be infinitely divisible for routing. In reality, the most common communication networks are packet-switched networks, such as the Internet, where the smallest possible data unit for routing is a packet. Therefore, it is necessary to propose a more practical scheme that can closely approximate GLS.

The proposed scheme should have the following objectives. First, it should be able to split traffic on multiple routes as fairly as possible. This means that, given the granularity constraints on routing, it should try to approximate GLS according to the routing weight vector as closely as possible, for every time period.

Second, its implementation should be simple and its applicability does not need a substantial modifications on existing protocols. For example, the two most commonly used Internet protocols are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). Each TCP connection requires each packet to arrive at the destination in order. If a TCP connection routes packets on multiple paths simultaneously, those packets sent on different paths may arrive at the destination out of order. "False losses" and a fast retransmit mechanism are then triggered. This results in a significant deterioration on protocol performance. This problem can be partially alleviated by re-scheduling the sending packet order so that all packets within the same flow are expected to arrive at the destination in order [17]. Nevertheless, to avoid modifications in TCP, all packets within the same call should be routed on the same path. Packet-based load sharing approaches may not work well for TCP flows and other connection-oriented flows that require packets to arrive at the destination in order. Yet, a coarser call-based multipath routing approach can be applied for load sharing. On the contrary, an UDP connection or any other connectionless traffic allows packets to arrive at the destination out of order, without affecting the protocol performance. Therefore, packet-based approaches can be utilized to realize a fine-grained load sharing on multiple routes for connectionless traffic.

To satisfy the captioned requirements, a load sharing approach, called weighted fair routing (WFR), is proposed. The main philosophy of the proposed scheme is to minimize the deviation of the actual load distribution from the given routing weight vector in making each routing decision. There are two levels of routing granularity, namely, at the packet level, and at the call level. The packet-by-packet WFR (PWFR) is a packet-level WFR in which a set of packets is split on a set of (logical) outgoing channels or links, whereas the call-by-call WFR (CWFR) is a call-connection level WFR in which a set of connections is split on a set of outgoing channels and all packets belonging to the same connection are routed on the same path.

The discussion will proceed as follows. Section 3.1 describes the PWFR algorithm, gives the performance bounds achieved by PWFR, and examines the application of PWFR to provide service with guaranteed performance. Section 3.2 outlines the CWFR algorithm and discusses how to incorporate it in multiprotocol label switching (MPLS) for traffic engineering. Section 3.3 describes how to couple the basic PWFR and CWFR algorithms so as to permit a traffic splitter to handle both connection-oriented and connectionless traffic simultaneously.

3.1. A Packet-Based WFR: PWFR

3.1.1. Algorithm

Suppose there is a sequence of packets, namely, Packet 1, Packet 2, ..., to be split on a set of N (logical) paths or channels. Denote the size of Packet k by $S(k)$ bytes. The routing weight for Path i is given as p_i, where $\sum_{i=1}^{N} p_i = 1$. Define the routing weight vector as $\mathbf{p} = (p_1 \quad p_2 \quad \ldots \quad p_N)$. Let $W_i^p(k)$ and $\hat{W}_i^p(k)$ respectively be the expected workload (in bytes), based on p_i, and the actual workload (in bytes) to be sent on Path i, just after the routing decision for the k^{th} packet has been made. Without loss of generality, we assume $p_i > 0$ and define:

$$W_i^p(0) = \hat{W}_i^p(0) = 0$$

for all $i = 1, 2, \ldots, N$.

For an idealized load sharing performed by a GLS traffic splitter,

$$W_i^p(k) = p_i \cdot \sum_{j=1}^{k} S(j)$$

where $i = 1, 2, \ldots, N$.

By the conservation of traffic, there is neither creation nor destruction of traffic when a routing decision is made by any traffic splitters. This means that the total traffic to be routed by all paths is independent of what traffic splitter is chosen. Therefore,

$$\sum_{i=1}^{N} W_i^p(k) = \sum_{i=1}^{N} \hat{W}_i^p(k) = \sum_{j=1}^{k} S(j)$$

The idea of PWFR is to simulate GLS as closely as possible, with the constraint that each packet is transmitted on a path as a whole. The assignment of a complete packet to a path may cause a *transient* load imbalance with respect to the targetted routing weight vector. That is, some paths may be fed more traffic than expected temporarily while other paths may have less, after the routing decision for a certain packet, say Packet $(k-1)$, is made. Those paths fed with more traffic than expected have the tendency of not having the next packet, i.e. Packet k, routed on them. However, if Packet k is large and an overloaded path has a large routing weight, it may still be preferable to send the packet on this overloaded path. Therefore, both the current level of load imbalance as well as the size of the next successive packet are required for the traffic splitter to make the next routing decision.

To quantify the above selection criterion, a metric is introduced to measure the traffic underload on a path. The residual workload of Path i, where $i = 1, 2, \ldots, N$, just *before* the routing decision for Packet k is made, $R_i^p(k)$, is defined as the amount of work (in bytes) that should be fed on Path i in order to achieve the expected workload $W_i^p(k)$. In other words,

```
procedure PWFR_Packet( Packet )
begin
    $S$ ← Packet.size
    for each path $i$ from 1 to N
        $R_i^p$ ← $R_i^p + p_i S$
    choose a path $j$ such that $R_j^p$ is maximized [a]
    place Packet to the output queue of Path $j$
    $R_j^p$ ← $R_j^p - S$
end.
```

[a] Ties are broken by a path with the largest routing weight, and, if still unresolved, the smallest path identification number.

Figure 3. The PWFR algorithm.

$$R_i^p(k) = \begin{cases} W_i^p(1) & \text{if } k = 1; \\ W_i^p(k) - \hat{W}_i^p(k-1) & \text{otherwise.} \end{cases}$$

and hence

$$\sum_{i=1}^{N} R_i^p(k) = S(k)$$

We use $R_i^p(k)$ to measure the traffic underload on Path i, just before the routing decision of Packet k is made. If $R_i^p(k) > 0$, Path i has been injected with less traffic than expected, and hence Packet k can be sent on this path. On the other hand, if $R_i^p(k) < 0$, Path i has too much traffic being routed on it and hence Packet k should not be transmitted on this path. In other words, $R_i^p(k)$ provides an indicator to the traffic splitter for deciding which path Packet k should be transmitted on. Packet k is to be sent on Path j when $R_j^p(k)$ is maximized along all participating paths. Ties are broken by a path with the largest routing weight, and, if still unresolved, the smallest path identification number. The complete PWFR algorithm is summarized in Figure 3.

3.1.2. Performance Bounds

A traffic splitting algorithm is *deterministically fair* if, for any sequences of packets to be dispersed, the absolute difference between the actual workload and the expected workload (in bytes) allocated to each path is always bounded by a finite constant. This means that any deterministically fair traffic splitter attempts to approximate GLS such that the workload deviation on every path is limited by a certain constant. In this section, we are going to show that, if all packets are bounded in sizes, PWFR is a deterministically fair traffic splitting algorithm.

Lemma 2 *For every positive integer k, there exists a positive residual workload on some active Path i just before the routing* *decision for a positive-size Packet k is made. That is,*

$$R_i^p(k) > 0$$

for some $i \in \{1, 2, \ldots, N\}$.

Lemma 3 *For every positive integer k, the actual workload allocated to Path i cannot exceed the expected workload by an amount equal to or more than the maximum packet size* $S\mathrm{max}$. *That is,*

$$\hat{W}_i^p(k) - W_i^p(k) < S\mathrm{max}$$

where $i = 1, 2, \ldots, N$.

Lemma 4 *For every positive integer k, the actual workload allocated to Path i cannot lag behind the expected workload by an amount equal to or more than* $(N - 1) \cdot S\mathrm{max}$, *where N is the number of paths and* $S\mathrm{max}$ *is the maximum packet size. That is,*

$$W_i^p(k) - \hat{W}_i^p(k) < (N-1) \cdot S\mathrm{max}$$

where $i = 1, 2, \ldots, N$.

Theorem 1 *Given that all packets are bounded in sizes, PWFR is a deterministically fair traffic splitting algorithm.*

In practice, the routing weight vector may change with time. This means that the routing weight vector when Packet k arrives may be different from that when Packet $(k+1)$ arrives, for some positive integer k. It is easy to show that the lemmas and theorem in this section still hold when the routing weight vector changes between any two successive packet arrivals as their proofs do not require any specific relationship among routing weights.

3.1.3. Provision of Guaranteed Performance Service

Traditionally, the Internet can only provide a best-effort delivery service. To remedy this deficiency, a guaranteed quality of service (QOS) architecture [25] has been proposed. Weighted fair queueing (WFQ) [8], also known as packet-by-packet GPS (PGPS) [23], can be employed, as a queueing discipline, in every participating network router to provide firm bounds on end-to-end packet queueing delays, when packets within the same flow are all sent on a single path. In this section, we show that, with the application of PWFR, the provision of guaranteed QOS can be extended to cases where multiple paths are utilized simultaneously to transmit packets which belong to the same flow.

The following lemma shows that, when a leaky bucket constrained traffic source is divided on a set of paths by a PWFR traffic splitter, the offered workload to each path is also leaky bucket constrained.

Lemma 5 *Suppose a traffic source is constrained by a leaky bucket with parameters* (σ, ρ), *where* σ *and* ρ *represent the maximum size of the bucket token pool and the average token generation rate respectively. Let* $S\mathrm{max}$ *be the maximum packet size. If the traffic source is fed into a PWFR traffic splitter, a dispersed sub-flow to be routed on Link i, where* $i = 1, 2, \ldots, N$, *is constrained by another leaky bucket with parameters* (σ_i, ρ_i), *where*

$$\begin{cases} \sigma_i = p_i\,\sigma + S\mathrm{max} \\ \rho_i = p_i\,\rho \end{cases}$$

Compared with Lemma 1, a PWFR traffic splitter requires an expansion of the bucket token pool by the maximum packet size to constrain a dispersed sub-flow. This means that a PWFR traffic splitter may give a dispersed sub-flow with a larger burstiness than the one outputted from the idealized GLS traffic splitter. Yet, the difference can become insignificant if a traffic source is very bursty in nature (with a large σ). This is generally true for multimedia traffic found in the current Internet.

Based on the result from Lemma 5, we can derive performance bounds for the provision of guaranteed QOS service using multiple paths for a single flow, which is leaky bucket constrained by the parameters (σ, ρ). The performance bounds are useful as they can be adopted to routing policed traffic with end-to-end QOS guarantees, such as in [22], with the extension of supporting the concurrent use of multiple routes. Further research is needed to devise an efficient and practical algorithm for QOS multipath routing.

Denote by g_i (where $g_i \geq \rho_i$), h_i, C_{ij}, and ζ_{ij}, respectively, the reserved bandwidth on Path i, the number of hops for Path i, the link capacity at the j^{th} hop for Path i, and the propagation delay at the j^{th} hop for Path i. It can be shown [29] that the delay bound for Path i, $D_i(p_i, g_i)$, the delay jitter bound for Path i, $J_i(p_i, g_i)$, and the buffer space requirement at the j^{th} hop router for Path i, $B_{ij}(p_i)$, respectively, can be written as:

$$\begin{cases} D_i(p_i, g_i) = \frac{\sigma_i + h_i\,S\mathrm{max}}{g_i} + \sum_{j=1}^{h_i}\left(\frac{S\mathrm{max}}{C_{ij}} + \zeta_{ij}\right) \\ J_i(p_i, g_i) = \frac{\sigma_i + h_i\,S\mathrm{max}}{g_i} \\ B_{ij}(p_i) = \sigma_i + j\,S\mathrm{max} \end{cases}$$

Denote the reserved bandwidth vector by $\mathbf{g} = (g_1 \quad g_2 \quad \ldots \quad g_N)$. The end-to-end delay bound can be computed as:

$$D(\mathbf{p}, \mathbf{g}) = \max_{i=1}^{N}(D_i(p_i, g_i))$$

and the end-to-end delay jitter bound can be calculated as:

$$\begin{aligned} J(\mathbf{p}, \mathbf{g}) &= D(\mathbf{p}, \mathbf{g}) - \min_{i=1}^{N}(D_i(p_i, g_i) - J_i(p_i, g_i)) \\ &= D(\mathbf{p}, \mathbf{g}) - \min_{i=1}^{N}\left(\sum_{j=1}^{h_i}\left(\frac{S\mathrm{max}}{C_{ij}} + \zeta_{ij}\right)\right) \end{aligned}$$

3.2. A Call-Based WFR: CWFR

PWFR can closely imitate GLS so that the actual workload allocated to any one path will exceed the expected workload allocated to that path by no more than the maximum packet size. However, the major drawback of using multiple paths for transmitting a single flow of packets is that these packets may arrive at the destination out of order, thereby causing a substantial performance deterioration for some protocols, such as TCP which is commonly deployed in the Internet. Thus, it is necessary to devise a simple flow-based load sharing algorithm, known as CWFR, to allow using only one path to send packets of the same flow, as well as to mimic GLS as closely as possible.

3.2.1. Algorithm

The idea of CWFR is to distribute a set of connections to a set of paths such that the workload allocated to each path is as close to the expected workload as possible. Suppose there is a set of connections, namely Connection 1, Connection 2, ..., to be distributed on a set of N (logical) paths or channels. Connections may have different characteristics. For example, Connection 1 may be established at a time different from that of Connection 2. Besides, Connection 1 may have a longer holding time or larger bandwidth requirement than that of Connection 2. To facilitate a fair distribution of connections to the set of paths, we assume that the bandwidth requirement of each connection, such as its equivalent bandwidth or average bandwidth needed, is known a priori, or can be estimated (such as according to what application it is to support), during the call establishment phase.

The algorithm works as follows. Suppose each call has a finite bandwidth requirement. Connection k needs a bandwidth requirement of $Q(k)$ units from the captioned network node. If a call is routed on a certain path or channel away from that node, the needed bandwidth for this call is reserved on that path. When the node receives a call establishment request from an upstream node, it needs to decide which of the N outgoing channels is to be used for routing the call. The decision is based on the reserved bandwidth on each channel during the time the request is made, as well as the bandwidth requirement for the incoming call. When a node receives a call termination request, it will release all bandwidth reservations corresponding to that call.

Define $\hat{W}_i^c(k)$ as the reserved bandwidth on Path i just *before* Connection k is established. The total reserved bandwidth for all calls, including the incoming one, on all outgoing channels at the time when Connection k is established can be computed as:

$$A(k) = \sum_{i=1}^{N} \hat{W}_i^c(k) + Q(k)$$

To enjoy the desired call-level load sharing, we hope that the total reserved bandwidth can be split according to the routing weight vector $\mathbf{p} = (p_1 \quad p_2 \quad \ldots \quad p_N)$, where p_i is the routing weight for Path i, and $\sum_{i=1}^{N} p_i = 1$. Denote by $W_i^c(k)$ the expected reserved bandwidth on Path i at the time when Connection k is made. This means that, for every positive integer k and $i = 1, 2, \ldots, N$,

$$W_i^c(k) = p_i \, A(k)$$

The bandwidth deviation on Path i, where $i = 1, 2, \ldots, N$, just *before* the routing decision for Connection k is established, $R_i^c(k)$, is defined as the amount of bandwidth that should be reserved on Path i in order to have a reserved bandwidth equal to the expected reserved bandwidth on Path i. Thus, the bandwidth deviation on Path i can be written as:

$$R_i^c(k) = W_i^c(k) - \hat{W}_i^c(k)$$

for all $k = 1, 2, \ldots$.

We use the value of $R_i^c(k)$ to measure the level of load imbalance on Path i, just before the routing decision of Connection k is made. Similar to the argument made for the PWFR algorithm, if $R_i^c(k) > 0$, Path i is underloaded and Connection k can be routed on this path. On the contrary, if $R_i^c(k) < 0$, Path i is already overloaded and Connection k should not be carried on Path i. Connection k is to be routed on Path j when $R_j^c(k)$ is maximized along all active paths. Ties are broken by a path with the largest routing weight, and, if still unresolved, the smallest path identification number. The complete CWFR algorithm is shown in Figure 4.

```
procedure CWFR_Establishment(Call)
begin
    $Q \leftarrow$ Call.bandwidth
    $A \leftarrow \sum_{i=1}^{N} \hat{W}_i^c + Q$
    for each path $i$ from 1 to N
        $W_i^c \leftarrow p_i \, A$
        $R_i^c \leftarrow W_i^c - \hat{W}_i^c$
    choose a path $j$ such that $R_j^c$ is maximized [b]
    Call.path $\leftarrow j$
end.
```

[b] Ties are broken by a path with the largest routing weight, and, if still unresolved, the smallest path identification number.

```
procedure CWFR_Termination(Call)
begin
    $Q \leftarrow$ Call.bandwidth
    $i \leftarrow$ Call.path
    $\hat{W}_i^c \leftarrow \hat{W}_i^c - Q$
end.
```

```
procedure CWFR_Packet(Packet)
begin
    $i \leftarrow$ Packet.call.path
    place Packet to the output queue of Path $i$
end.
```

Figure 4. The CWFR algorithm.

3.2.2. Incorporation of MPLS for Traffic Engineering

A potential drawback of the CWFR algorithm is that it may be necessary to maintain states so that a network node knows how to forward incoming packets corresponding to each call, since calls are routed individually at each node. It seems that it is not scalable with the growing number of flows in the network. However, with the introduction of label switching techniques like MPLS, the scalability problem can be alleviated. All flows sharing the same path segment can be assigned the same flow label so that they can be switched together. Specific information with respect to a flow can be referenced from the label stack, which is a last-in, first-out stack to store a set of label stack entries. A label stack entry contains a flow label which can be employed by a node to choose the path a packet should be forwarded to. When a packet arrives at a node at the end of a path segment, the top-most label stack entry will be used to determine the next hop label forwarding entry, which contains information to determine the packet's next hop and the operation to perform on the packet's label stack.

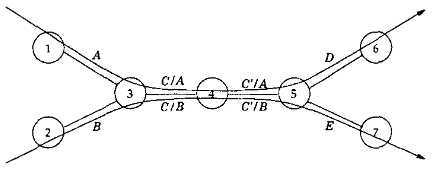


Figure 5. A seven-node network.

The forwarding process can be illustrated by a seven-node network, as shown in Figure 5. Consider two group of flows. The path taken by the first and second groups of flows, respectively, are: Node 1 $\rightarrow$ Node 3 $\rightarrow$ Node 4 $\rightarrow$ Node 5 $\rightarrow$ Node 6, and Node 2 $\rightarrow$ Node 3 $\rightarrow$ Node 4 $\rightarrow$ Node 5 $\rightarrow$ Node 7. With the use of MPLS, all packets belonging to the first group of flows carry A as the flow label when they are sent from Node 1 to Node 3. Similarly, all packets belonging to the second group of flows carry B as the flow label when they are sent from Node 2 to Node 3. At Node 3, the label A or B is pushed into the label stack and the flow label is replaced by C, since the two groups of flows begin to share the same path segment. At Node 4, packets within the two flow groups are switched and forwarded to Node 5, and the flow label is taken over by C'. When such a packet arrives at Node 5, if it carries a flow label C', another flow label is popped from the label stack to determine which path and flow label are to be used next. If the label A is popped, the packet is assigned with the flow label D and forwarded to Node 6. If the label B is popped, the packet is assigned with the flow label E and forwarded to Node 7.

With the application of flow label and label stack, the traffic splitter does not need to maintain states for every active flow. Instead, it needs to maintain states for each participating flow label. Since a flow label may be shared by many flow groups, the number of flow labels maintained at each network node can then be dramatically reduced. Thus, the system is as scalable as other label-switching networks, such as Asynchronous Transfer Mode (ATM) which employs virtual path/virtual channel concept for performance enhancement.

3.3. Combining PWFR and CWFR

Generally, packet-switching networks need to handle both connection-oriented and connectionless traffic. We may need to couple the basic PWFR and CWFR algorithms to obtain the combined WFR algorithm.

When a packet is to be forwarded to another node, it is necessary to determine whether it belongs to connectionless traffic or connection-oriented traffic. If the former applies, the PWFR algorithm can be used directly. Otherwise, the route for that packet has been pre-determined and this information should be retrieved accordingly. However, routing a packet on a pre-determined path may cause even more load imbalance among a set of paths. To alleviate the impact to load distribution, the residual workloads on all participating paths are incremented by their expected workloads contributed from the packet. The residual workload on the pre-determined path is then reduced by the size of that packet. The combined WFR algorithm is depicted in Figure 6.

```
procedure MWFR_Packet(Packet )
begin
    if ( Packet.class is connectionless )
        PWFR_Packet( Packet )
    else { Packet.class is connection-oriented }
        $S \leftarrow$ Packet.size
        $i \leftarrow$ Packet.call.path
        for each path j from 1 to N
            $R_j^p \leftarrow R_j^p + p_j S$
        CWFR_Packet( Packet )
        $R_i^p \leftarrow R_i^p - S$
end.
```

Figure 6. The combined WFR algorithm.

4. Performance Evaluation

In this section, we first outline a set of traffic splitting algorithms for performance studies. A performance metric is then defined for comparing various splitting algorithms. Afterwards, we briefly describe the characteristics of the traffic traces used for the simulation, and finally, we examine the simulation results of the proposed traffic splitting algorithms.

4.1. Traffic Splitting Algorithms

We study three basic traffic splitting algorithms, namely, weighted fair routing (WFR), generalized round robin routing (GRR), and probabilistic routing (PRR). The WFR approach, which is described in Section 3, simulates generalized load sharing (GLS) as closely as possible, subject to the granularity constraint imposed. The packet-by-packet WFR (PWFR) simulates GLS with the limitation that packets are routed as a whole, while the call-by-call WFR (CWFR) simulates GLS in such a way that all packets of a single flow follow the same transmission path.

GRR distributes packets or calls to each route such that the number of packets or calls allocated to each path relative to the sum on all paths is as close to its routing weight as possible. The packet-by-packet GRR (PGRR) splits packets to each path in a cyclical fashion so that the arrival instants of any two packets to each path is as uniform as possible. The call-by-call GRR (CGRR) dispatches an incoming call to a path such that the resulting load distribution, in terms of the number of active calls, to a set of paths is closest to the routing weight vector. For dealing with mixed traffic, GRR performs adjustments on PGRR just as PWFR does. Indeed, a way to implement GRR is to apply WFR by setting all packet sizes and bandwidth requirements of all calls to one unit, or any other constant.

PRR spreads out packets or calls to each route in random such that the chance for routing a packet or a call to a specific path is equal to the routing weight for that path. According to the routing weight vector, the packet-by-packet PRR (PPRR) divides packets at random, whereas the call-by-call PRR (CPRR) routes calls at random. Indeed, PRR is equivalent to a perfect hashing routing scheme, which provides a performance benchmark to all other hashing-based traffic splitting algorithms. A list of direct hashing and table-based hashing schemes for call-based traffic splitting can be found in [5].

4.2. Performance Metric

Our performance metric for a traffic splitter is the mean squared workload deviation, which measures the variation of the actual workloads allocated by the traffic splitter to a set of N paths from the expected ones distributed under GLS. A good traffic splitting algorithm should divide traffic according to the given routing weight vector, and hence it should be able to keep the value of the mean squared workload deviation as small as possible.

Suppose a set of M packets, namely, Packet 1, Packet 2, ..., Packet M, which belong to either connectionless or connection-oriented traffic, are split on a set of paths. Let $W_i^p(k)$ and $\hat{W}_i^p(k)$ respectively be the expected and actual workloads (in bytes) allocated to Path i, just after the routing decision for the k^{th} packet has been made. The mean squared workload deviation for the set of packets is defined as:

$$E[(\hat{W}^p - W^p)^2] = \frac{\sum_{k=1}^{M} \sum_{i=1}^{N} (\hat{W}_i^p(k) - W_i^p(k))^2}{MN}$$

4.3. Traffic Traces

The simulation is based on a set of real packet traces collected on the MCI Internet Backbone over two trunks, namely, the FIX-West facility at NASA-Ames (FDDI interface), and the SDSC OC12mon vBNS connection. A total of six traces, from July 1994 to December 1999, are employed for our simulation studies. The characteristics of these traffic traces are summarized in Figure 7.

Trace File	Duration in Sec.	# TCP Calls	# TCP Packets	# non-TCP Packets	# TCP Bytes	# non-TCP Bytes
FXW-19940714	3 624	121 628	7 680 715	2 245 811	1 874 271 141	674 803 625
FXW-19960918	299	217 590	3 491 749	1 529 446	1 431 367 056	435 232 031
12SDC-19991115a	89	5 388	435 936	1 196 674	312 679 995	1 573 644 381
12SDC-19991115b	89	6 806	451 451	1 250 882	281 740 032	1 611 799 302
12SDC-19991215a	89	4 862	418 442	235 128	294 629 822	116 560 362
12SDC-19991215b	90	4 876	509 902	290 316	360 136 635	140 277 214

Figure 7. The characteristics of traffic traces.

4.4. Simulation Results

Our simulation experiments are based on a network node which routes incoming traffic on three outgoing paths. It is sufficient to only consider three paths as it has been shown [16], under a wide range of scenarios, that multipath routing is effective in using a small number of paths, say up to three. The routing weight vector, $\mathbf{p} = (p_1 \quad p_2 \quad p_3)$, has been set such that $p_3 = 0$ or 0.35. When $p_3 = 0$, p_1 varies between 0.001 and 0.5, with a total of eleven data points. Due to symmetry, it is not necessary to perform duplicate experiments when p_1 is greater than 0.5. For instance, the result of $p_1 = 0.7$ is the same as that of $p_1 = 0.3$ since, for both cases, the routing weight of one path is 0.3 and that of another is 0.7, and thus the results should be the same. Similarly, when $p_3 = 0.35$, p_1 varies between 0.001 and 0.3, with a total of seven data points.

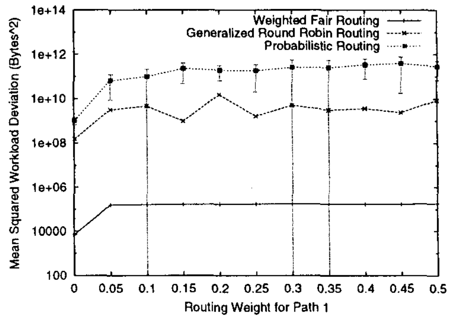


Figure 8. Workload deviation plot for connectionless traffic with $p_3 = 0$.

Each simulation run is filled with a complete traffic trace, and the statistics for computing the performance metric are collected only when the routes for the first 10000 packets have been determined. For WFR and GRR, a single run is sufficient to determine the mean squared workload deviation for a certain setting. However, since PRR involves the use of random numbers, a total of ten runs have been done to find an average value of the performance metric, and a 95% confidence interval for each average value of the metric is also computed. Because of constraints in space, plots showing similar results are left out and hence only plots corresponding to the trace file "12SDC-19991115a" are presented here.

The results compare the effectiveness of different traffic splitters under three different traffic types, namely, connectionless traffic, connection-oriented traffic, and mixed traffic. Connection-oriented traffic consists of traffic from Transmission Control Protocol (TCP) connections only, while connectionless traffic comes from non-TCP connections. Mixed traffic is composed of both connectionless and connection-oriented traffic. Figure 8 shows the mean squared workload deviation for connectionless traffic when the routing weight for Path 3 is 0, i.e. no traffic will be routed on Path 3. The routing weight for Path 1 varies between 0.001 and 0.5. We see that the mean squared workload deviation when WFR is employed is significantly lower than when GRR or PRR is used. Obviously, it is due to the fact that WFR aims to minimize workload deviation each time the route of each packet is assigned. Though GRR tries to distribute packets to paths as closely to the routing weight vector as possible, it does not take the dynamics of packet size into account. PRR routes packets at random and hence it can be expected that it can split the workload approximately equal to the expected load distribution over a very long time period. This means that, over short time periods, the workload may be very different from the expected one. Thus, this approach cannot minimize workload deviation in general.

Figure 9 exhibits the mean squared workload deviation for connection-oriented traffic when the routing weight for Path 3 is

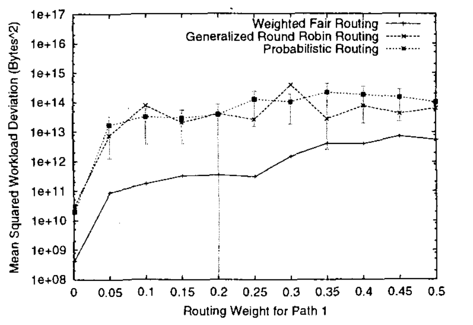


Figure 9. Workload deviation plot for connection-oriented traffic with $p_3 = 0$.

0. We find the mean squared workload deviation when WFR is applied is generally lower than when GRR or PRR is used. Comparing with the cases for connectionless traffic, its superiority fades as all traffic within a single call must be transmitted on the same path once determined. This limits the power of WFR to balance traffic splitting in proportion to the set of routing weights. Moreover, the average call bandwidth consumed, instead of equivalent bandwidth, for simplification purposes, is taken as the bandwidth requirement. This may not be able to properly identify various call requirements where traffic characteristics vary substantially for different calls.

Figure 10 gives the mean squared workload deviation for mixed traffic when the routing weight for Path 3 is 0. As for connectionless traffic, the mean squared workload deviation is always the lowest value when WFR is used. Nevertheless, the level of improvement differs with the proportion of connectionless traffic. A larger proportion of connectionless traffic drives a greater performance improvement.

The mean squared workload deviation for connectionless traffic, connection-oriented traffic, and mixed traffic when the routing weight for Path 3 is 0.35 is similar to cases when the routing weight for Path 3 is 0, and thus the corresponding plots are omitted to conserve space.

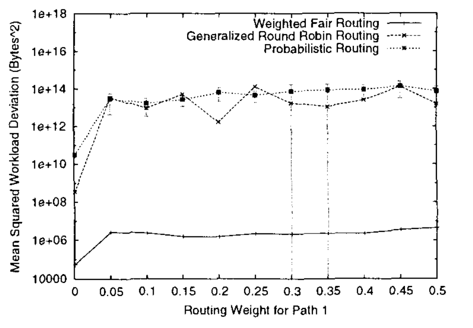


Figure 10. Workload deviation plot for mixed traffic with $p_3 = 0$.

5. Conclusions

In this paper, we have proposed a framework to study how to effectively perform load sharing in multipath communication networks. A generalized load sharing (GLS) model has been developed to conceptualize how traffic is split ideally on a set of active paths. A simple traffic splitting algorithm, called weighted fair routing (WFR), has been devised at two different granularity levels, namely, the packet level, and the call level, to approximate GLS with the given routing weight vector. The packet-by-packet WFR (PWFR) mimics GLS by transmitting each packet as a whole, whereas the call-by-call WFR (CWFR) imitates GLS so that all packets belonging to a single flow are sent on the same path. We have developed some performance bounds for PWFR and found that PWFR is a deterministically fair traffic splitting algorithm. This attractive property is useful in the provision of service with guaranteed performance when multiple paths can be used simultaneously to transmit packets which belong to the same flow.

A total of six historical Internet backbone traces have been used in our simulation studies. For each of the traffic trace, we investigated three different scenarios: connectionless traffic, connection-oriented traffic, and mixed traffic. Connection-oriented traffic consists of traffic from Transmission Control Protocol (TCP) connections only, while connectionless traffic comes from non-TCP connections. Mixed traffic is composed of both connectionless and connection-oriented traffic. Our simulation studies, based on these traffic traces, reveal that WFR outperforms two other traffic splitting algorithms, namely, generalized round robin routing (GRR), and probabilistic routing (PRR), in all three scenarios. These promising results can form a basis for designing future adaptive quality of service (QOS) or constraint-based multipath routing protocols.

There are several possible extensions to our work, some of which are listed below:

- devise an efficient algorithm for constraint-based multipath routing;
- develop an adaptive load sharing architecture for packet-switching networks.

Acknowledgement

This research is supported in part by the University Grants Committee, Hong Kong, Area of Excellence in Information Technology, Grant No. AOE98/99.EG01. We would also like to thank the United States National Science Foundation Cooperative Agreement No. ANI-9807479 and the National Laboratory for Applied Network Research for making their real Internet backbone packet traces available for our research. Last, but not least, we are grateful to three anonymous referees for their valuable comments which assist us in improving the quality of this paper.

References

[1] H. Adiseshu, G. Varghese, and G. Parulkar. An Architecture for Packet-Striping Protocols. *ACM Transactions on Computer Systems*, Vol. 17, No. 4, pp. 249-287, November 1999.

[2] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for Traffic Engineering Over MPLS. *Request for Comments*, RFC 2702, Network Working Group, Internet Engineering Task Force, September 1999.

[3] D. O. Awduche. MPLS and Traffic Engineering in IP Networks. *IEEE Communications Magazine*, Vol. 37, No. 12, pp. 42-47, December 1999.

[4] D. Bertsekas and R. Gallager. *Data Networks*. Second Edition. Prentice Hall, 1992.

[5] Z. Cao, Z. Wang, and E. Zegura. Performance of Hashing-Based Schemes for Internet Load Balancing. *Proceedings of IEEE INFOCOM 2000*, Vol. 1, pp. 332-341, Tel Aviv, Israel, 26-30 March 2000.

[6] S. N. Chiou and V. O. K. Li. Diversity Transmissions in a Communication Network with Unreliable Components. *Proceedings of IEEE ICC '87*, Vol. 2, pp. 968-973, Seattle, WA, USA, 7-10 June 1987.

[7] J. H. Déjean, L. Dittmann, and C. N. Lorenzen. String Mode – A New Concept for Performance Improvement of ATM Networks. *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 9, pp. 1452-1460, December 1991.

[8] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. *Internetworking: Research and Experience*, Vol. 1, No. 1, pp. 3-26, September 1990.

[9] J. Duncanson. Inverse Multiplexing. *IEEE Communications Magazine*, Vol. 32, No. 4, pp. 34-41, April 1994.

[10] L. Fratta, M. Gerla, and L. Kleinrock. The Flow Deviation Method: An Approach to Store-and-Forward Communication Network Design. *Networks*, Vol. 3, pp. 97-133, 1973.

[11] P. H. Fredette. The Past, Present, and Future of Inverse Multiplexing. *IEEE Communications Magazine*, Vol. 32, No. 4, pp. 42-46, April 1994.

[12] R. G. Gallager. A Minimum Delay Routing Algorithm Using Distributed Computation. *IEEE Transactions on Communications*, Vol. COM-25, No. 1, pp. 73-85, January 1977.

[13] E. Gustafsson. *Traffic Dispersion in ATM Networks*. Ph.D. Dissertation, Royal Institute of Technology, TRITA-IT R 97:03, Kista, Sweden, June 1997.

[14] J. van der Lande. Inverse Multiplexing Over ATM: Broadband Access For Less. *Computer Technology Review*, Vol. 18, No. 12, pp. 24,26, December 1998.

[15] T. T. Lee, S. C. Liew, and Q.-L. Ding. Parallel Communications for ATM Network Control and Management. *Performance Evaluation*, Vol. 30, No. 4, pp. 243-264, October 1997.

[16] K.-C. Leung and V. O. K. Li. A Resequencing Model for High Speed Networks. *Proceedings of IEEE ICC '99*, Vol. 2, pp. 1239-1243, Vancouver, BC, Canada, 6-10 June 1999.

[17] K.-C. Leung and V. O. K. Li. Flow Assignment and Packet Scheduling for Multipath Networks. *Proceedings of IEEE GLOBECOM '99*, Vol. 1 (V01a), pp. 246-250, Rio de Janerio, RJ, Brazil, 5-9 December 1999.

[18] V. O. K. Li and W. Liao. Distributed Multimedia Systems. *Proceedings of the IEEE*, Vol. 85, No. 7, pp. 1063-1108, July 1997.

[19] N. F. Maxemchuk. Dispersity Routing. *Proceedings of IEEE ICC '75*, pp. 41-10 - 41-13, San Francisco, CA, USA, June 1975.

[20] N. F. Maxemchuk. Dispersity Routing in High-Speed Networks. *Computer Networks and ISDN Systems*, Vol. 25, No. 6, pp. 645-661, January 1993.

[21] N. F. Maxemchuk. Dispersity Routing on ATM Networks. *Proceedings of IEEE INFOCOM '93*, Vol. 1, pp. 347-357, San Francisco, CA, USA, 28 March - 1 April 1993.

[22] A. Orda. Routing with End-to-End QoS Guarantees in Broadband Networks. *IEEE/ACM Transactions on Networking*, Vol. 7, No. 3, pp. 365-374, June 1999.

[23] A. K. Parekh and R. G. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case. *IEEE/ACM Transactions on Networking*, Vol. 1, No. 3, pp. 344-357, June 1993.

[24] A. K. Parekh and R. G. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple Node Case. *IEEE/ACM Transactions on Networking*, Vol. 2, No. 2, pp. 137-150, April 1994.

[25] S. Shenker, C. Partridge, and R. Guerin. Specification of Guaranteed Quality of Service. *Request for Comments*, RFC 2212, Network Working Group, Internet Engineering Task Force, September 1997.

[26] C. B. S. Traw and J. M. Smith. Striping Within the Network Subsystem. *IEEE Network*, Vol. 9, No. 4, pp. 22-32, July-August 1995.

[27] C. Villamizar. OSPF Optimized Multipath (OSPF-OMP). *Internet Draft*, Internet Engineering Task Force, Work in Progress, 18 August 1999.

[28] S. Vutukury and J. J. Garcia-Luna-Aceves. A Simple Approximation to Minimum-Delay Routing. *Computer Communication Review*, Vol. 29, No. 4, pp. 227-238, October 1999.

[29] H. Zhang. Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks. *Proceedings of the IEEE*, Vol. 83, No. 10, pp. 1374-1396, October 1995.