Generalized Load Sharing for Packet-Switching Networks II: Flow-Based Algorithms

Ka-Cheong Leung, *Member*, *IEEE*, and Victor O.K. Li, *Fellow*, *IEEE*

Abstract—In this paper, we extend the load sharing framework to study how to effectively perform flow-based traffic splitting in multipath communication networks. The generalized load sharing (GLS) model is employed to conceptualize how traffic is split ideally on a set of active paths. A simple flow-based weighted fair routing (WFR) algorithm, called call-by-call WFR (CWFR), has been developed to imitate GLS so that all packets belonging to a single flow are sent on the same path. We have investigated how to couple the proposed basic packet-by-packet WFR (PWFR) and CWFR algorithms so as to permit a traffic splitter to handle both connection-oriented and connectionless traffic simultaneously. 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.

Index Terms—Computer communications, dispersity routing, high speed networks, inverse multiplexing, load sharing, multipath routing, multiprotocol label switching, network striping, performance modeling, traffic dispersion, traffic engineering.

1 Introduction

In a companion paper [6], we have proposed a generalized load sharing (GLS) model to conceptualize how traffic is split ideally on a set of active paths. A simple traffic splitting algorithm, called packet-by-packet weighted fair routing (PWFR), has been devised to approximate GLS with the given routing weight vector by transmitting each packet as a whole. We have analyzed and developed some performance bounds for PWFR and found that PWFR is a deterministically fair traffic splitting algorithm. In this paper, we will extend this load sharing framework to investigate how to effectively perform flow-based traffic splitting in multipath communication networks.

Since a large volume of traffic in the Internet is Transmission Control Protocol (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 [1], [7] 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 quality of service (QoS) requirements and they should not be treated equally.

• *The authors are with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong, China. E-mail: kcleung@ieee.org, vli@eee.hku.hk.*

Manuscript received 2 Feb. 2004; revised 17 Feb. 2005; accepted 2 June 2005; published online 25 May 2006.
Recommended for acceptance by J. Wu.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0039-0204.

1.1 Our Contributions

The focus of this work, first described in [5], is to extend the load sharing framework to study how to effectively perform flow-based traffic splitting in multipath communication networks. 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 investigated how to couple the proposed basic packet-by-packet WFR (PWFR) and CWFR algorithms so as to permit a traffic splitter to handle both connection-oriented and connectionless traffic simultaneously.

A scalable architecture for providing deterministic guarantees has been developed in [8] to support flow splitting and aggregation over multiple paths. During the call establishment process, the authors have suggested three different methods to route the connection by, namely, picking the widest outgoing link at each hop, picking the shortest path, and dividing the request into k smaller equal-size flows and picking k widest outgoing links at each hop. All these methods do not consider a targeted routing weight vector for load balancing. Given an optimal traffic split, our proposed WFR assumes a given targeted routing vector so as to achieve a better load balancing of traffic over a set of active paths.

Our proposed framework differs from the one developed in [9] in three ways. First, their datagram forwarding does not discuss which path is chosen when there are two or more underloaded paths, whereas our PWFR algorithm picks the most underutilized path with the same time and storage complexities. Second, the route of each TCP traffic flow is determined by hashing in [9]. Its optimal performance is the same as the call-by-call probabilistic routing

(CPRR). Our CWFR algorithm attempts to consider the bandwidth requirement of a flow before its route is determined. Our simulation results in Section 3 demonstrate that CWFR outperforms CPRR for all configurations considered. Third, both their datagram forwarding algorithm and the hybrid forwarding algorithm allow their working variables to grow unboundedly, while the working variables of CWFR and MWFR have their ranges limited by the level of load imbalance. Indeed, we have proved in our companion paper [6] that, if all packets are bounded in sizes and the number of paths is fixed, the workload deviation on every path for PWFR is confined by a certain constant.

1.2 Organization of the Paper

This paper is organized as follows. Section 2 presents both packetized and flow-based weighted fair routing algorithms to achieve the best load balancing according to the given traffic split. Section 3 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 4 concludes and discusses some possible extensions to our work.

2 Weighted Fair Routing

Like the generalized processor sharing (GPS) approach, the generalized load sharing (GLS) approach proposed in the companion paper [6] 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. 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, a User Datagram Protocol (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.

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

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

Fig. 1. The PWFR algorithm.

The discussion will proceed as follows. Section 2.1 briefly describes the PWFR algorithm proposed in the companion paper [6]. Section 2.2 outlines the CWFR algorithm and discusses how to incorporate it in multiprotocol label switching (MPLS) for traffic engineering. Section 2.3 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.

2.1 A Packet-Based WFR: PWFR

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 $\boldsymbol{p} = (p_1 \quad p_2 \quad \cdots \quad p_N)$. 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. Packet k is to be sent on Path j when $R_j^p(k)$ is 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 Fig. 1.

2.2 A Call-Based WFR: CWFR

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.

2.2.1 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. $\hat{W}_i^c(k)$ can be obtained by maintaining the counter $\hat{W}_i^c$ as exhibited in Fig. 2. Alternatively, $\hat{W}_i^c(k)$, which represents the measured aggregate effective bandwidth for all existing flows on Path i, can be evaluated by, say taking estimates over a set of measurement windows during the call establishment phase [2], [3]. 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

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

To enjoy the desired call-level load sharing, we hope that the total reserved bandwidth can be split according to the routing weight vector $\boldsymbol{p} = (p_1 \quad p_2 \quad \ldots \quad p_N)$, 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 $i = 1, 2, \ldots, N$,

$$W_i^c(k) = p_i\, A(k). \tag{2}$$

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

$$R_i^c(k) = W_i^c(k) - \hat{W}_i^c(k) \tag{3}$$

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, Connection k is to be routed on Path j when 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 Fig. 2.

As far as the time and space complexities are concerned, CWFR takes $O(N)$ time for each router to determine a routing decision during call establishment, and $O(1)$ time to either relinquish the reserved bandwidth during call termination or forward incoming packets of a call. CWFR requires $O(N)$ counters to store its working variables. Because the number of paths N is generally small and fixed,

```
procedure CWFR_Establishment( Call )
begin
    Q ← Call.bandwidth
    A ← Σ_{i=1}^{N} Ŵ_i^c + Q
    for each path i from 1 to N
        W_i^c ← p_i A
        R_i^c ← W_i^c − Ŵ_i^c
    choose a path j such that R_j^c is maximized [b]
    Call.path ← j
    Ŵ_j^c ← Ŵ_j^c + Q
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 ← Call.bandwidth
    i ← Call.path
    Ŵ_i^c ← Ŵ_i^c − Q
end.

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

Fig. 2. The CWFR algorithm.

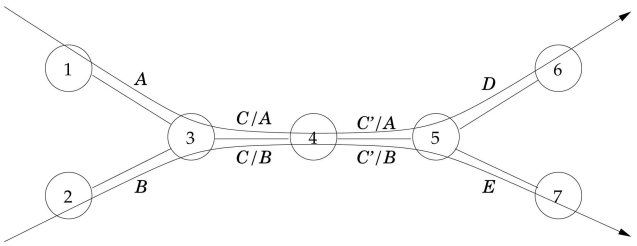

Fig. 3. A seven-node network.

we expect that the computational and storage costs are minimal with respect to the number of traffic flows maintained at each router, nowadays.

2.2.2 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 a label switching technique 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 topmost 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.

The forwarding process can be illustrated by a seven-node network, as shown in Fig. 3. Consider two groups 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 the virtual path/virtual channel concept for performance enhancement.

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

Fig. 4. The combined WFR algorithm.

2.3 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 predetermined and this information should be retrieved accordingly. However, routing a packet on a predetermined 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 predetermined path is then reduced by the size of that packet. The combined WFR algorithm is depicted in Fig. 4.

As far as the time and space complexities are concerned, MWFR calls either PWFR or CWFR once to forward an incoming packet. If the packet belongs to connection-oriented traffic, MWFR updates all R_j^p counters for any Path j to keep track of the traffic underloads on all N paths. Thus, MWFR takes $O(N)$ time and $O(N)$ counters for traffic splitting.

3 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. Afterward, 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.

Trace File	Duration in Sec.	# TCP Calls	# TCP Packets	# non-TCP Packets	# TCP Bytes	# non-TCP Bytes	$S_{\max}^{\text{TCP}}$	$S_{\max}^{\text{non-TCP}}$
SDZ-19940714	*3 622.845238*	*121 628*	*7 680 715*	*2 245 811*	*1 874 271 141*	*674 803 625*	*2 184*	*1 500*
FXW-19960918	*298.712016*	*217 590*	*3 491 749*	*1 529 446*	*1 431 367 056*	*435 232 031*	*4 352*	*4 348*
12SDC-19991115a	*88.921181*	*5 388*	*435 936*	*1 196 674*	*312 679 995*	*1 573 644 381*	*4 136*	*4 458*
12SDC-19991115b	*88.839599*	*6 806*	*451 451*	*1 250 882*	*281 740 032*	*1 611 799 302*	*4 136*	*4 458*
12SDC-19991215a	*88.879599*	*4 862*	*418 442*	*235 128*	*294 629 822*	*116 560 362*	*4 136*	*9 180*
12SDC-19991215b	*89.287843*	*4 876*	*509 902*	*290 316*	*360 136 635*	*140 277 214*	*4 136*	*9 180*
IND-20010815	*90.094056*	*87 353*	*1 321 518*	*1 525 490*	*729 404 800*	*537 911 885*	*9 180*	*4 426*

Fig. 5. The characteristics of traffic traces.

3.1 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 2, 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 are 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 at 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 [1].

3.2 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 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 kth packet has been made. The mean squared workload deviation for the set of packets is defined as

$$E[(\hat{W}^p - W^p)^2] = \frac{\sum_{k=1}^{M}\sum_{i=1}^{N}(\hat{W}_i^p(k) - W_i^p(k))^2}{MN}. \quad (4)$$

3.3 Traffic Traces

The simulation is based on a set of real packet traces collected on the Internet backbones for the Passive Measurement and Analysis (PMA) Project over four trunks, namely, the SDSC FDDI DMZ, the FIX-West facility at NASA Ames (FDDI interface), the SDSC OC12mon vBNS connection, and the Indiana University GigaPOP. A total of seven traces, from July 1994 to August 2001, are employed for our simulation studies. Denote the maximum packet sizes in bytes for Transmission Control Protocol (TCP) connections and non-TCP connections respectively by $S_{\max}^{\text{TCP}}$ and $S_{\max}^{\text{non-TCP}}$. The characteristics and the packet size distributions of these traffic traces are summarized in Figs. 5 and 6, respectively.

3.4 Simulation Results

Our simulation experiments are based on a network node which routes incoming traffic on a set of outgoing paths. Each simulation run is executed with a complete traffic trace, and the statistics for computing the performance metric are collected only when the routes for the first 10,000 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 10 runs have been done to find an average value of the performance metric, and a 95 percent 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 "IND-20010815" are presented here.

The results are provided in three sets. The first set examines the performance of different traffic splitters under two different traffic types, connection-oriented traffic and mixed traffic, for multipath routing with different numbers of homogeneous paths used, where the routing weight for

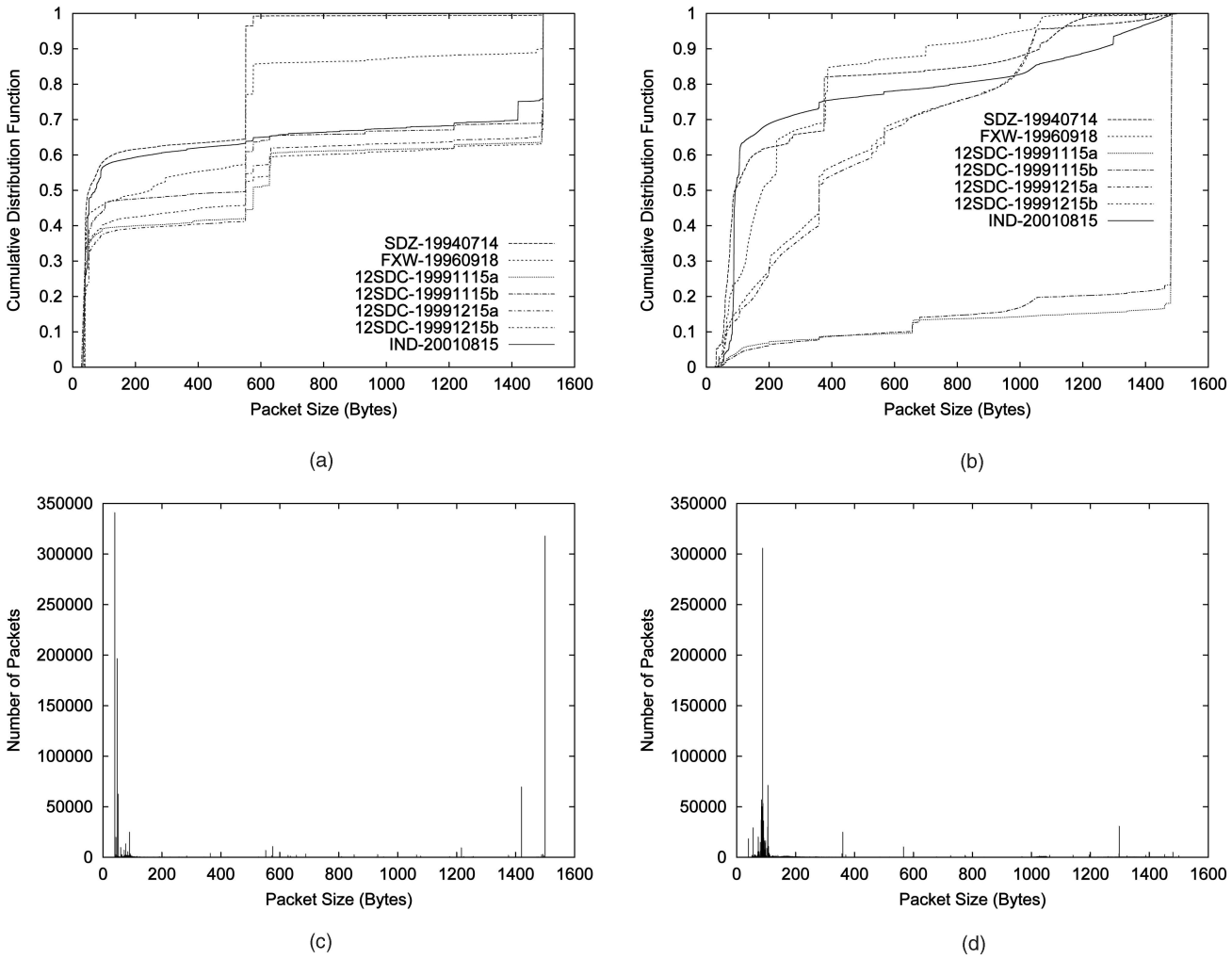

Fig. 6. Packet size distribution plot. (a) All seven traffic traces for TCP connections. (b) All seven traffic traces for non-TCP connections. (c) Trace file "IND-20010815" for TCP connections. (d) Trace file "IND-20010815" for non-TCP connections.

each path is the same. Connection-oriented traffic consists of traffic from TCP connections only, while connectionless traffic comes from non-TCP connections. Mixed traffic is composed of both connectionless and connection-oriented traffic. Results for connectionless traffic only have been reported in the companion paper [6] and they are not presented here.

Fig. 7 displays the mean squared workload deviation for connection-oriented traffic and mixed traffic when the number of paths used varies from one to 10. When there is only one outgoing path, no traffic splitting is needed and, thus, the mean squared workload deviation for all traffic splitters considered is always zero. By using two or more paths, we find that the mean squared workload deviation when WFR is employed is often substantially smaller than when GRR or PRR is used. The performance of WFR is more or less the same regardless of the number of paths used.

For connection-oriented traffic, 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.

For mixed 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 second set of results compares the effectiveness of different traffic splitters for heterogeneous multipath routing, where the routing weight for one path can differ from that for another path. It is sufficient to consider only three paths for the extensive study as it has been shown [4], 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, $\boldsymbol{p} = (\, p_1 \quad p_2 \quad p_3)$, 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 11 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

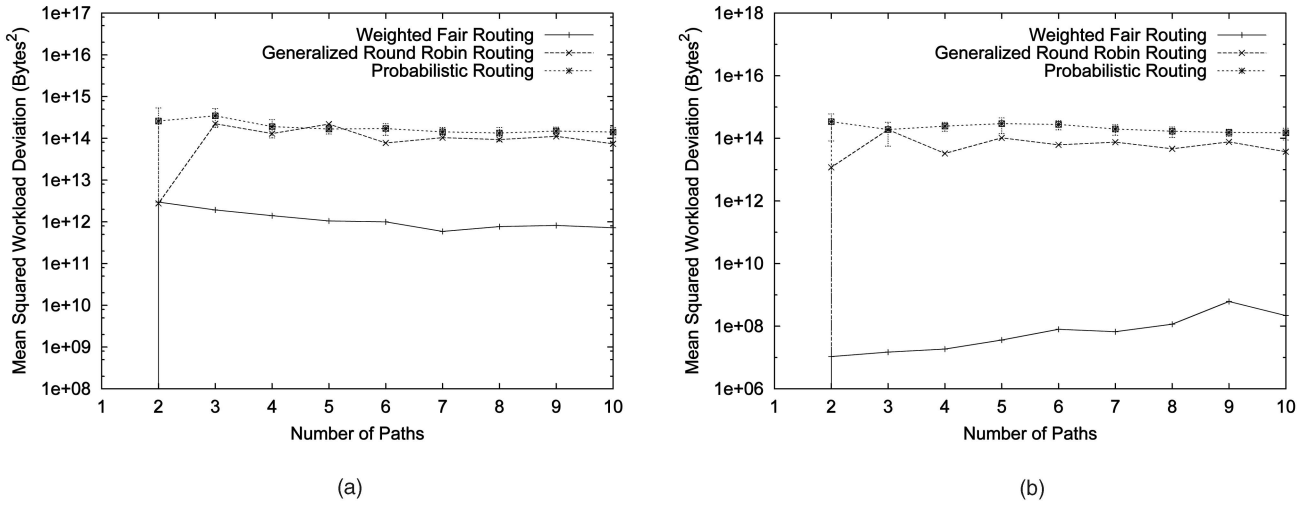

Fig. 7. Workload deviation plot for homogenous multipath routing. (a) Connection-oriented traffic. (b) Mixed traffic.

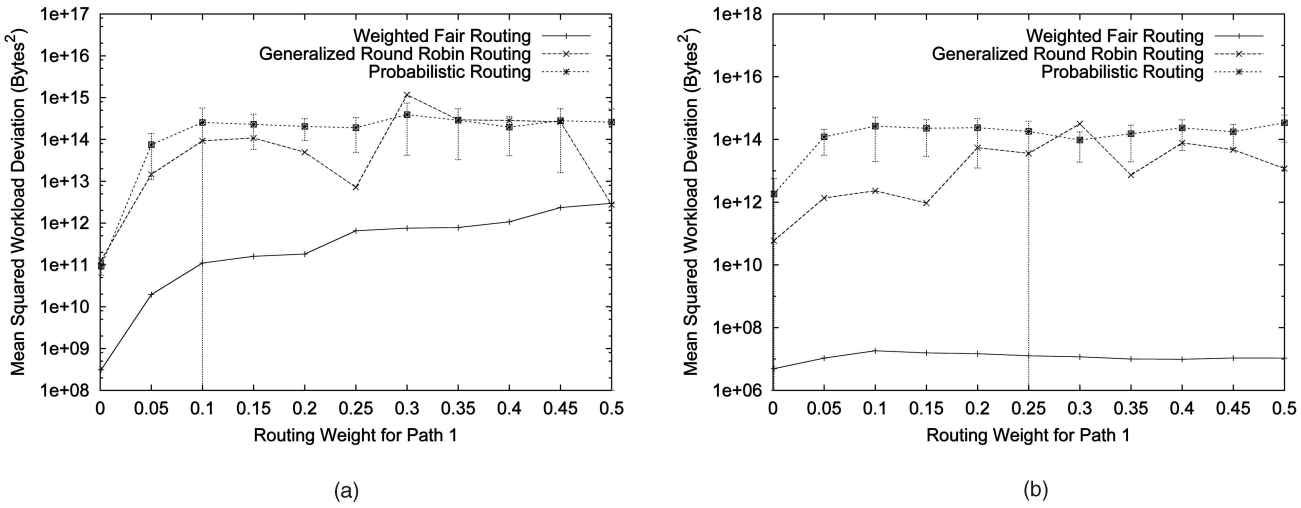

Fig. 8. Workload deviation plot with $p_3 = 0$. (a) Connection-oriented traffic. (b) Mixed traffic.

weight of one path is 0.3 and that of another is 0.7. Similarly, when $p_3 = 0.35$, p_1 varies between 0.001 and 0.3, with a total of seven data points.

Fig. 8 shows the mean squared workload deviation for connection-oriented traffic and mixed 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. Similar to our first set of results, we see that the mean squared workload deviation when WFR is employed is significantly lower than when GRR or PRR is used.

Fig. 9 shows the mean squared workload deviation for connection-oriented traffic and mixed traffic when the routing weight for Path 3 is 0.35, where the routing weight for Path 1 varies between 0.001 and 0.3. The results are similar to cases when the routing weight for Path 3 is 0.

The third set of results demonstrates the effectiveness of our proposed algorithms on load balancing by means of plotting a set of sample load vectors. Each sample load vector consists of two sample loads, each corresponding to a path. A sample load is taken at intervals of 0.001 second. The routing weight vector is $(\,0.35 \quad 0.65 \quad 0\,)$. Fig. 10 exhibits the distribution of sample load vectors for connection-oriented traffic. There are similar patterns of "cloudy" distributions on sample load vectors for WFR, GRR, and PRR. Yet, it is still clear that the vectors are slightly more localized when using WFR. This implies that WFR can provide a better load balancing of traffic, but its improvement is not as significant as for connectionless traffic.

Fig. 11 shows the distribution of sample load vectors for mixed traffic. It can be inferred that, using WFR, the sample load vectors are much less scattered around the line with the slope of 1.857, which is equal to the ratio of the routing weights between Path 1 and Path 2, than for those using GRR and PRR. This means that WFR gives better performance on load sharing. However, their shapes can differ significantly when different traffic traces are used. Actually, it stems from the fact that there is a better balancing on workload for traces with a larger component of connectionless traffic. Nevertheless, the improvement is still obvious although connectionless traffic constitutes only 28.3 percent of total traffic.

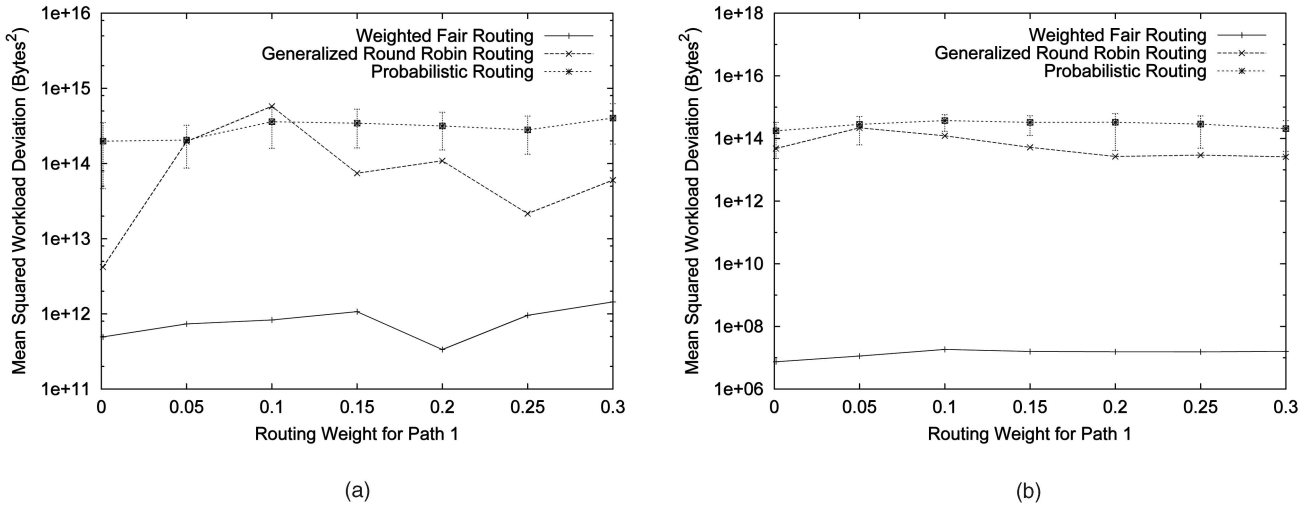

Fig. 9. Workload deviation plot with $p_3 = 0.35$. (a) Connection-oriented traffic. (b) Mixed traffic.

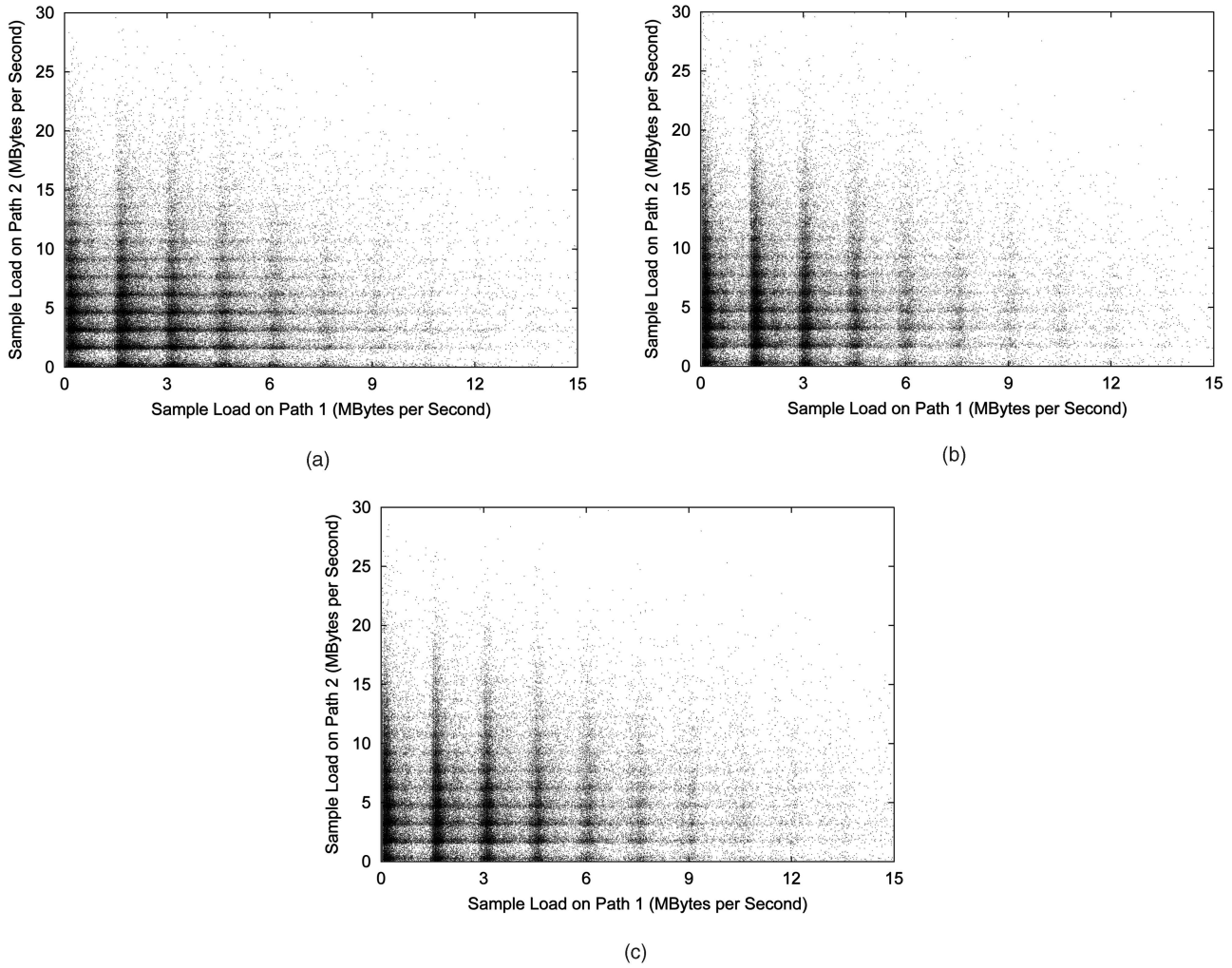

Fig. 10. Sample load plot for connection-oriented traffic with $p_1 = 0.35$, $p_2 = 0.65$, and $p_3 = 0$. (a) Weighted fair routing. (b) Generalized round robin routing. (c) Probabilistic routing.

4 Conclusions

In this paper, we have extended the load sharing framework to study how to effectively perform flow-based traffic splitting in multipath communication networks. The generalized load sharing (GLS) model is employed to conceptualize how traffic is split ideally on a set of active paths. A simple flow-based weighted fair routing (WFR) algorithm, called call-by-call WFR (CWFR), has been

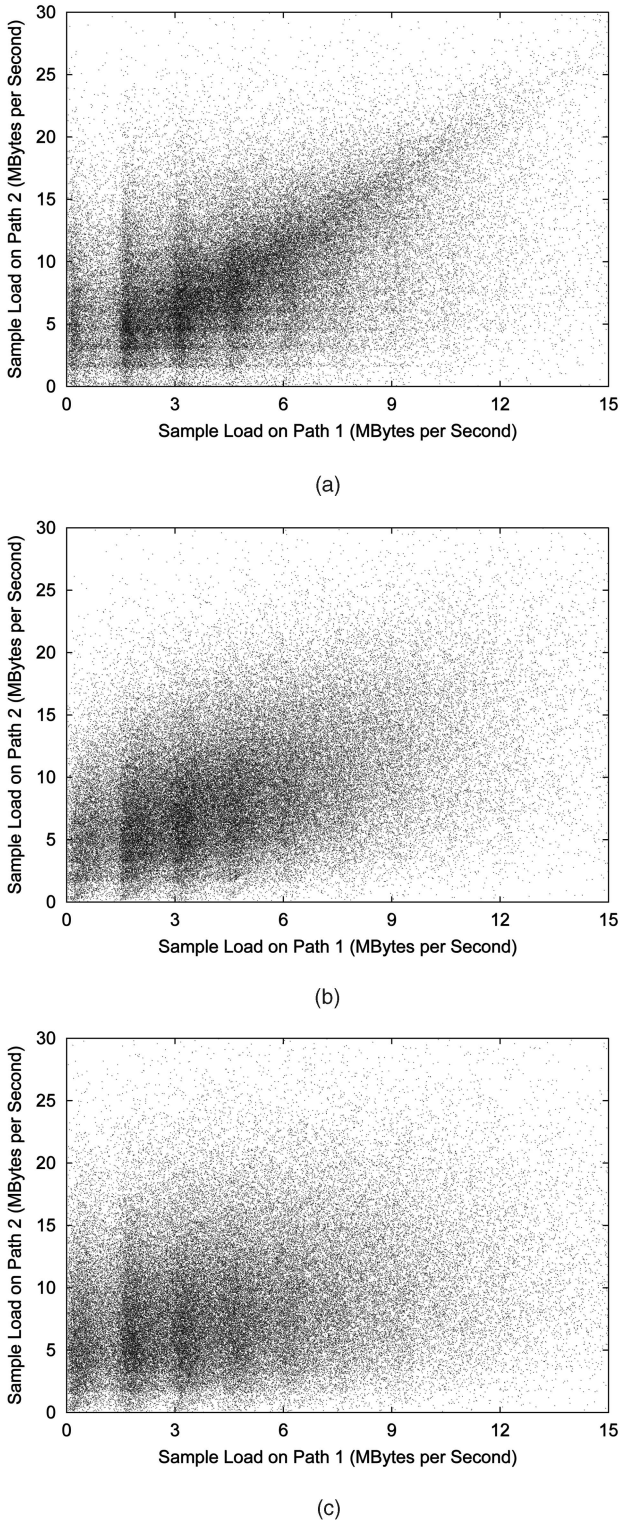

Fig. 11. Sample load plot for mixed traffic with $p_1 = 0.35$, $p_2 = 0.65$, and $p_3 = 0$. (a) Weighted fair routing. (b) Generalized round robin routing. (c) Probabilistic routing.

developed to imitate GLS so that all packets belonging to a single flow are sent on the same path. We have investigated how to couple the proposed basic packet-by-packet WFR (PWFR) and CWFR algorithms so as to permit a traffic splitter to handle both connection-oriented and connectionless traffic simultaneously.

A total of seven historical Internet backbone traces have been used in our simulation studies. For each of the traffic traces, we investigated two different scenarios: 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 both scenarios. These promising results form a basis for designing future adaptive quality of service (QoS) or constraint-based multipath routing protocols.

ACKNOWLEDGMENTS

This research is supported in part by the Areas of Excellence Scheme established under the University Grants Committee of the Hong Kong Special Administrative Region, China (Project Number AoE/E-01/99). The authors would also like to thank the US National Science Foundation Cooperative Agreement Numbers ANI-0129677 (2002) and ANI-9807479 (1998), the National Laboratory for Applied Network Research, Measurement and Network Analysis Group for making their real Internet backbone packet traces available for our research, and the anonymous reviewers for their valuable comments and suggestions which assisted us in improving the quality of the paper.

REFERENCES

[1] Z. Cao, Z. Wang, and E. Zegura, "Performance of Hashing-Based Schemes for Internet Load Balancing," *Proc. IEEE Infocom,* vol. 1, pp. 332-341, Mar. 2000.

[2] C. Casetti, J. Kurose, and D. Towsley, "An Adaptive Algorithm for Measurement-Based Admission Control in Integrated Services Packet Networks," *Computer Comm.,* vol. 23, nos. 14-15, pp. 1363-1376, Aug. 2000.

[3] S. Jamin, P.B. Danzig, S.J. Shenker, and L. Zhang, "A Measurement-Based Admission Control Algorithm for Integrated Service Packet Networks," *IEEE/ACM Trans. Networking,* vol. 5, no. 1, pp. 56-70, Feb. 1997.

[4] K.-C. Leung and V.O.K. Li, "A Resequencing Model for High Speed Networks," *Proc. IEEE Int'l Conf. Comm.,* vol. 2, pp. 1239-1243, June 1999.

[5] K.C. Leung and V.O.K. Li, "Generalized Load Sharing for Packet-Switching Networks," *Proc. IEEE Int'l Conf. Network Protocols,* pp. 305-314, Nov. 2000.

[6] K.C. Leung and V.O.K. Li, "Generalized Load Sharing for Packet-Switching Networks I: Theory and Packet-Based Algorithm," *IEEE Trans. Parallel and Distributed Systems,* vol. 17, no. 7, pp. 694-702, July 2006.

[7] C. Villamizar, "OSPF Optimized Multipath (OSPF-OMP)," Internet Draft, Internet Eng. Task Force, Work in Progress, Aug. 1999.

[8] S. Vutukury and J.J. Garcia-Luna-Aceves, "A Scalable Architecture for Providing Deterministic Guarantees," *Proc. IEEE Int'l Conf. Computer Comm. and Networks,* pp. 91-96, Oct. 1999.

[9] S. Vutukury and J.J. Garcia-Luna-Aceves, "A Traffic Engineering Approach Based on Minimum-Delay Routing," *Proc. IEEE Int'l Conf. Computer Comm. and Networks,* pp. 42-47, Oct. 2000.

Ka-Cheong Leung received the BEng degree in computer science from the Hong Kong University of Science and Technology in 1994 and the MSc degree in electrical engineering (computer networks) and the PhD degree in computer engineering from the University of Southern California, Los Angeles, in 1997 and 2000, respectively. He worked as a senior research engineer at Nokia Research Center, Irving, Texas, from 2001 to 2002. He was an assistant professor in the Department of Computer Science at Texas Tech University, Lubbock, Texas, between 2002 and 2005. Since June 2005, he has been with the University of Hong Kong, where he is a visiting assistant professor in the Department of Electrical and Electronic Engineering. His research interests include wireless packet scheduling, routing, congestion control, and quality of service guarantees in high-speed communication networks, content distribution, high-performance computing, and parallel applications. He is listed in the 60th (2006) edition of *Marquis Who's Who in America* and is a member of the IEEE.

Victor O.K. Li received the SB, SM, EE, and ScD degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1977, 1979, 1980, and 1981, respectively. He joined the University of Southern California (USC), Los Angeles, in February 1981 and became professor of electrical engineering and director of the USC Communication Sciences Institute. Since September 1997, he has been with the University of Hong Kong, where he is chair professor of information engineering. He also served as managing director of Versitech Ltd., the technology transfer and commercial arm of the university, from September 1997 to June 2004. He serves on various corporate boards. His research is in information technology, including all-optical networks, wireless networks, and Internet technologies and applications. Sought by government, industry, and academic organizations, he has lectured and consulted extensively around the world. He is very active in the research community, has chaired various international conferences and has served on the editorial boards of various international journals. He has given distinguished lectures at universities around the world and keynote speeches at many international conferences. He has received numerous awards, including, most recently, the UK Royal Academy of Engineering Senior Visiting Fellowship in Communications, the KC Wong Education Foundation Lectureship, the Croucher Foundation Senior Research Fellowship, and the Bronze Bauhinia Star, Government of the Hong Kong Special Administrative Region, China. He was elected an IEEE fellow in 1992.