Exploiting Parallelism to Boost Data-Path Rate in High-Speed IP/MPLS Networking

Indra Widjaja and Anwar I. Elwalid
Bell Laboratories, Lucent Technologies
Murray Hill, NJ 07974
{iwidjaja, anwar}@research.bell-labs.com

Abstract— **Link bundling is a way to increase routing scalability whenever a pair of Label Switching Routers in MPLS are connected by multiple parallel links. However, link bundling can be inefficient as a Label Switched Path (LSP) has to be associated with a particular link. In this paper, we show that the efficiency of link bundling can be significantly improved if traffic can be effectively distributed across the parallel links. We propose an IP switch architecture that is capable of distributing flows both inside the switch and among the parallel links based on operations that are relatively simple to implement. The switch requires no speedup, guarantees in-sequence packet delivery for a given flow, avoids complex co-ordination algorithms, and can achieve LSP throughput higher than the line rate. By means of simulation using IP traces, we investigate the performance of the proposed switch, and show that the switch achieves good load-balancing performance. We describe extensions to the basic architecture which allows for very large bundle size, handles incremental upgrade strategy, improves reliability, and accommodates non-IP traffic.**

Keywords— **Switch architecture, link bundling, hashing, load balancing, performance.**

I. Introduction

The Internet Protocol (IP) has become the dominant network-layer protocol for dealing with data traffic, and is increasing its importance for dealing with multi-service traffic. In the backbone network, IP packets are traditionally forwarded by routers hop-by-hop based on lookup results of IP destination addresses. Recently, the industry has been developing a new technology called MultiProtocol Label Switching (MPLS) which adds a connection capability in the data path of a router so that forwarding can rely on short "labels" [1]. A router that can perform forwarding based on labels is called a Label Switching Router (LSR). A labeled packet traverses along a Label Switched Path (LSP) from an ingress LSR to an egress LSR. The ingress LSR is responsible for grouping packets that are to be forwarded in the same manner into Forwarding Equivalence Classes (FECs). Packets of the same FEC are appended with the same label before being forwarded through the LSP. Each intermediate LSR uses the incoming label value of a packet to index into the Next Hop Label Forwarding Entry (NHLFE), which gives information of the outgoing label value, outgoing link, and possibly other book-keeping information. At the egress LSR, the top-most label is stripped from each packet, and if the resulting packet contains no more labels, forwarding will continue using the hop-by-hop approach.

Tier-1 Internet Service Providers (ISPs) have been constantly upgrading their IP/MPLS backbone networks towards higher speed links to meet the growth of IP traffic volume. Most major ISP backbone networks currently employ 10-Gbps link speed in the segments that carry a substantial amount of traffic. In many cases, the inter-city links do not carry the same capacities. Typically, these capacities range from OC-3 (150 Mbps) to OC-192 (10 Gbps) circuits (for example, see [2], [3] and [4]). In the future, it is likely that many of these links will have to carry traffic from tens to hundreds of Gbps.

The continual upgrades of links and routers/LSRs in response to increased bandwidth demand have not been without issues. Due to the artifact of SONET framing, each new generation of high-speed network interface typically increases its carrying capacity by a factor of four (e.g., from OC-12 to OC-48 to OC-192, etc.). This implies that just after an upgrade, a new link is likely to be inefficiently utilized as its utilization will be roughly at 25 percent of that in old link. Another potential disadvantage that can be very costly is that many routers require "forklift upgrades" whenever their network interfaces are to be upgraded to the next-generation speed. Moreover, this also makes the lifetime of a router to be relatively short. For example, if traffic growth is at 60 percent annually and a new link is four times faster than the old one, then router upgrades occur approximately every three years.

Incremental link upgrades can be facilitated by deploying multiple parallel links between two neighboring routers. This allows the aggregate link capacity to be increased more gradually, and the lifetime of a router to be prolonged as long as the available network interfaces have not been exhausted. In the future as link bandwidth demand grows to hundreds or thousands of Gbps, multiple links may be the only economical way to increase capacities as the electronic circuitry (especially the random access memory) cannot practically support such a speed. High-speed switch architectures that have to work with today's DRAM speed are contemporary topics of research investigation (for example, see [5][6]).

Link bundling is as a way to increase routing scalability whenever a pair of LSRs are connected by multiple parallel links having the same attributes [7]. This is done by advertising the parallel links as a single link into the Interior Gateway Protocol (IGP) such as OSPF or IS-IS, resulting in a reduced amount of information that has to be flooded and a smaller link-state database. The link that is advertised into IGP is referred to as a *bundled link*, while each of the parallel links is referred to as a *component link*. The advantage of implementing link bundling (LB) in terms of routing scalability is depicted in Figure 1. Without link bundling (Fig. 1 (a)), IGP needs to keep track of each individual parallel link in the network. Link bundling (Fig. 1 (b)) essentially makes IGP view the parallel links as a single link.

Link bundling, however, requires an LSP to be associated

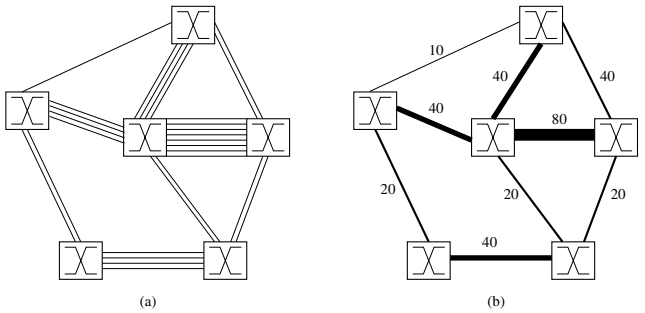

Fig. 1. Network topologies: (a) physical, (b) logical.

with a particular component link. The association is done during LSP setup by the sender of an RSVP-TE Path message (or a CR-LDP REQUEST message), and generally lasts for the lifetime of an LSP. A new LSP request with bandwidth reservation B can be established only if at least one of the component links has unreserved bandwidth (equivalent to available bandwidth, but may be different due to a booking factor) greater than B. If two component links have a combined unreserved bandwidth greater than B, but each component link has unreserved bandwidth less than B, then the reservation will fail. We will show that packing inefficiency in link bundling may result in relatively high LSP blocking rate.

In this paper, we present a more efficient bundling scheme, called link bundling with distributed traffic assignment (LB/DA), which not only increases routing scalability, but also improves overall system performance. LB/DA allows LSP traffic to be distributed among the component links within a bundled link (or bundle). To prevent misordering of packets within a given flow, LB/DA distributes LSP traffic to component links at the flow level rather than at the packet level. We will show that flow assignment can be quite uniform in the backbone network where there exist quite a large number of flows on a link. With LB/DA, a new LSP request with bandwidth reservation B would be satisfied as long as the bundle has total unreserved bandwidth greater than B. In other words, the maximum bandwidth that can be reserved for an LSP can potentially be as much as the capacity of the bundle.

The main challenge in supporting LB/DA is to construct an IP/MPLS switch that:

1. is capable of emulating a bundle as a true fat pipe so that LSPs with bandwidth reservation higher than the bandwidth of each individual component link may be established,
2. runs at the speed no greater than that of a component link (i.e., no speedup), and
3. maintains packet ordering for each flow.

To this end, we propose a switch architecture which satisfies the above requirements. We take advantage of the symmetry in the switch architecture to obviate implementation complexity associated with real-time coordination algorithms that manage contention in the switch. We show that our proposed switch becomes more robust in terms of its load balancing ability as the switch becomes more stressed. Other desirable features include growability, flexibility and reliability.

The rest of this paper is organized as follows. In Section II, we compare the performance of LB versus LB/DA at the connection level. In particular, we compare the "lost revenue" in the two systems. In Section III, we present the operation and design of our proposed switch which is capable of implementing LB/DA. We then evaluate the performance of our switch in Section IV. In Section V, we address support for large bundle sizes, switch growability and upgrade strategy, switch protection, and support for non-IP traffic. Finally, we conclude the paper in Section VI.

II. LB VERSUS LB/DA

In MPLS, the ingress LSR typically establishes an LSP by originating an RSVP-TE Path message which then traverses through each intermediate LSR along the path as specified by the Explicit Route Object (ERO) until the Path message arrives at the egress LSR [8]. Labels are allocated by a Resv message traversing in the opposite direction issued by the egress LSR upon receipt of the Path message.

When a pair of LSRs are connected by a bundle with LB, the upstream LSR needs to select a particular component link to be associated with the requested LSP before forwarding a Path message. If the LSP request is satisfied, bandwidth reservation is made for the selected component link, and packets of that LSP can only be sent on that selected component link.

Unlike LB which dedicates an LSP to a component link, LB/DA associates an LSP with a bundle. When a pair of LSRs are connected by a bundle with LB/DA, the upstream LSR can freely forward a Path message on any of the component links. If the LSP request is satisfied, bandwidth reservation is made for the bundle, and packets of that LSP can be distributed on all component links of the bundle. The key distinction between the two schemes in terms of LSP assignment is illustrated in Figure 2.

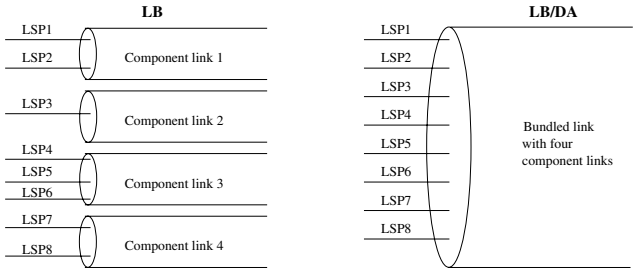

Fig. 2. LB versus LB/DA.

With LB, an LSP request on a bundle is blocked if the unreserved bandwidth for each component link is less than the requested bandwidth. On the other hand, an LSP request with LB/DA is blocked only if the total unreserved bandwidth for the bundle is less than the requested bandwidth. It is intuitively clear that LB is less efficient than LB/DA since it is possible that an LSP request with LB may not fit into any component link even though the total unreserved bandwidth among the component links are greater than the LSP bandwidth requirement. To derive a simple model for comparing the efficiencies of both LB and LB/DA at the connection level, we first assume that all LSP requests have the same bandwidth requirement and that there are always LSP requests waiting to be established. Let the LSP bandwidth requirement be denoted by x, where the bandwidth requirement is normalized to the capacity of a component link.

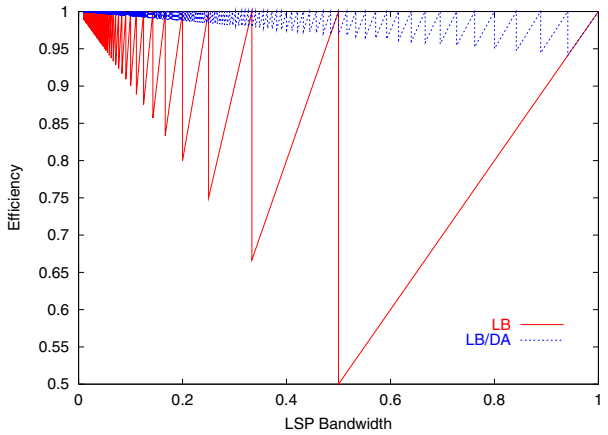

Fig. 3. Efficiency of link bundling.

We define efficiency as the maximum possible utilization of the bundle. It is easy to see that the efficiency of LB, $E_{LB}(x)$, for this simple case is given

$$E_{LB}(x) = \begin{cases} x\lfloor 1/x \rfloor & 0 \le x \le 1, \\ 0 & x > 1, \end{cases}$$

independent of the bundle size, where $\lfloor y \rfloor$ is the greatest integer less than or equal to y. For example, if $x = 0.5$, then two LSPs can be fitted for each component link and the efficiency is 1. On the other hand, if $x = 0.5 + \epsilon$ ($\epsilon \sim 0$), then only one LSP can be fitted for each component link, which lowers the efficiency to 0.5. With LB/DA, the efficiency is given by

$$E_{LB/DA}(x) = \begin{cases} (x/W)\lfloor W/x \rfloor & 0 \le x \le W, \\ 0 & x > W, \end{cases}$$

where W is the bundle size.

The efficiency of LB and LB/DA for this simple model with respect to LSP bandwidth requirement is plotted in Figure 3. Here, it is assumed that the bundle size is 16. Note that LB/DA generally achieves a much higher efficiency than LB.

It is worthwhile to point out that the efficiency of LB/DA improves as the bundle size increases. Furthermore, LB/DA also allows for LSP requests to exceed the capacity of a component link.

We now consider a more practical case where LSP requests may have different bandwidth requirements. In particular, we assume that a new request would have an LSP bandwidth requirement that is uniformly distributed from 0 to B_{max}. We also assume that LSP requests arrive at an LSR according to a Poisson process and that LSP holding time is exponentially distributed. When two or more component links can satisfy an LSP bandwidth requirement in LB, it is up to the LSR to select which outgoing component link to be associated with the LSP. In this paper, we evaluate two selection strategies: least-loaded and most-loaded. In the least-loaded (LL) strategy, the component link with the highest unreserved bandwidth is selected. In the most-loaded (ML) strategy, the component link with the lowest unreserved bandwidth that is higher than the LSP bandwidth requirement is selected.

To compare LB and LB/DA, we choose a metric based on revenue. Define the realized revenue, R_r, as the ratio of total bandwidth provided to the total bandwidth requested at an LSR. For simplicity, we assume that revenue is proportional to the bandwidth. Define the lost revenue, R_l, as $1 - R_r$. Figure 4 (a) compares the lost revenues per unit of time as a function of offered load for the case where the bundle size is equal to 16, and B_{max} is equal to 1 (i.e., the capacity of the link component). As can be seen from the figure, LB with LL incurs the highest revenue lost, while LB/DA incurs the least. For example, when the offered load is 0.8, LB with LL incurs a 26-percent loss of revenue, LB with ML incurs a 14-percent loss, and LB/DA incurs a 6-percent loss. Notice that LB with ML is better than LB with LL. The reason is that LB with ML tries to pack LSPs to as small number of component links as possible, which has the effect of maximizing the amount of unreserved bandwidth in each component link.

It is also of interest to examine the lost revenue when the LSP bandwidth requests may be higher than the component's capacity. Figure 4 (b) shows that even when the bandwidth requests exceed the component's capacity by a relatively small percentage (e.g., $B_{max} = 1.1$), LB incurs a significantly higher loss of revenue. Notice that the penalty in LB is still appreciable even when the offered load is very small. This figure demonstrates that if LSP bandwidth requests may exceed 1 as may very well happen in practice, then the performance of LB/DA will be far more superior to that of LB.

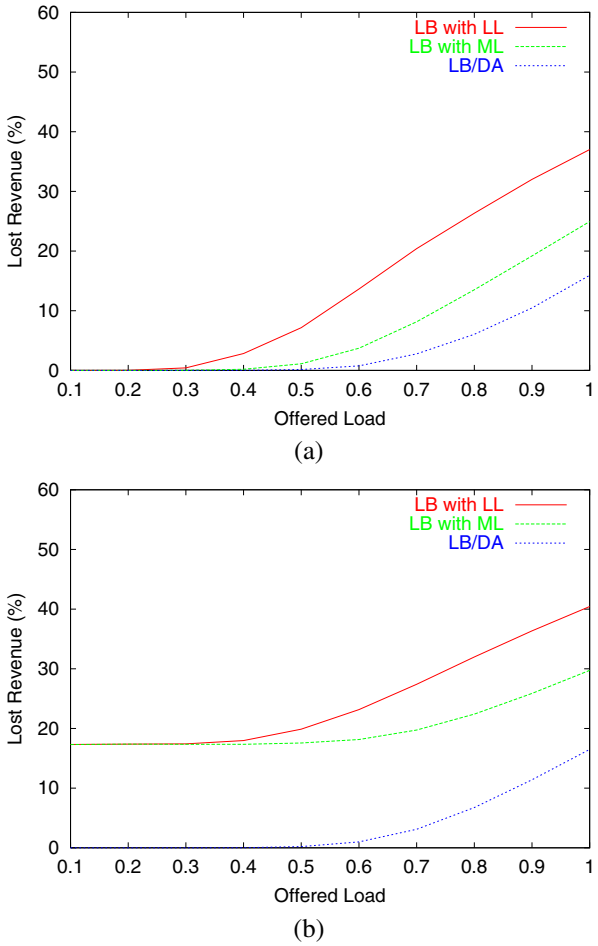

Fig. 4. Lost revenue versus offered load: (a) $B_{max} = 1$, (b) $B_{max} = 1.1$.

III. Switch Architecture

In this section, we present the basic switch architecture that is capable of uniformly distributing flows inside the switch and among the component links of a bundle. We will discuss some extensions to the basic architecture in Section V.

A. Switch Operation

Before we present the detail of the switch design in Section III-B, here we describe the switch operation. Figure 5 shows the block diagram of the switch architecture which is composed of three stages of packet switching elements (PSEs). In this paper, each PSE is assumed to be an output-buffered packet switch. Each PSE in the first and third stage can be configured to have several numbers of bundles of varying number of component links. Since the bundle is bi-directional, the inputs and outputs in the figure are symmetric. For simplicity, we assume that the bundle size is never greater than the first-stage PSE size (this assumption will be removed later in Section V-A). At each PSE in the first stage, its output i ($i = 1, \cdots, N$) is connected to PSE i in the second stage. Similarly, at each PSE in the second stage, its output j ($j = 1, \cdots, K$) is connected to PSE j in the third stage. The first-stage and third-stage PSEs are of size $N \times N$, while the second stage is of size $K \times K$. It is important to note that that there is no restriction on the values of N and K, as would be required in the non-blocking three-stage circuit switch [9]. The choice of these values depends mainly on implementation cost and practical switch size.

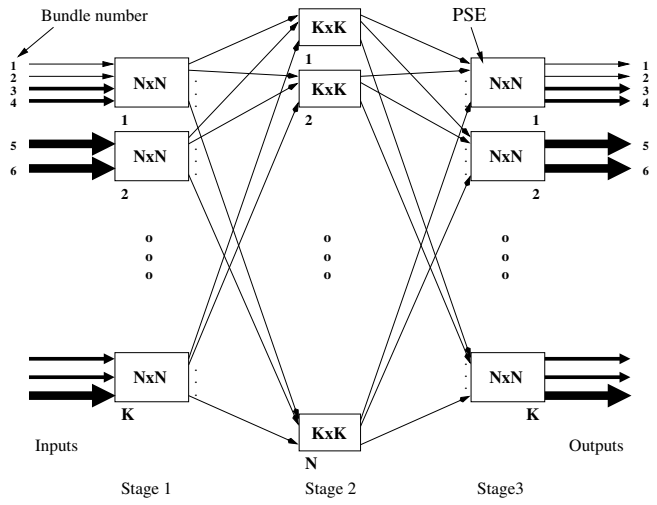

Fig. 5. Basic switch architecture.

Figure 6 shows how a traffic stream consisting of many IP flows is routed inside the switch. The first-stage PSE distributes the traffic stream from each of its inputs uniformly among its outputs. The distribution function, whose detail will be described in in Section III-B, ensures that packets of a given flow follow the same order. The second-stage PSE routes each of its arriving packets to its output according to which third-stage PSE the packet is destined to. Finally, the third-stage PSE routes each of its arriving packets to the appropriate bundle. The third-stage PSE also performs a similar distribution function so that packets intended for the same bundle are uniformly distributed among the component links.

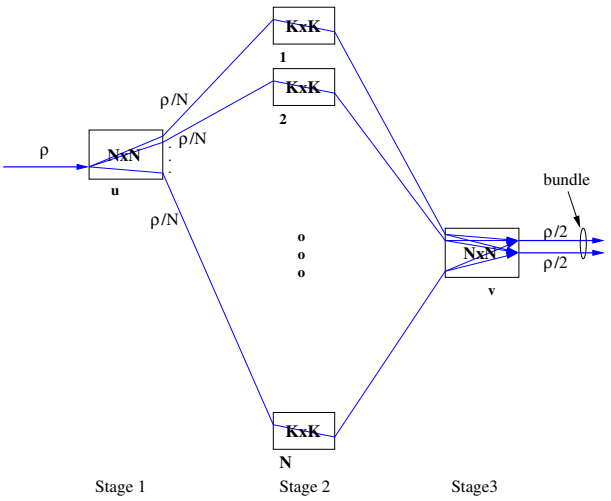

Fig. 6. Traffic splitting inside the switch.

To best understand how the load balancing works in the switch, focus on a particular third-stage PSE v. Let the offered load (normalized to a component link) of the aggregate traffic stream from the first-stage PSE u destined to the third-stage PSE v be denoted by ρ_{uv} ($u = 1, 2, \cdots, K$). The aggregate offered load to PSE v from all possible inputs is $\rho_v = \sum_{u=1}^{K} \rho_{uv}$. Note that the traffic stream destined to PSE v has to pass through output buffer v of each second-stage PSE. Let ρ_{uv}^w be the portion of the offered traffic that originates from PSE u and destined to PSE v via the second-stage PSE w. Then, the aggregate offered load to output buffer v of PSE w from all possible inputs is $\sum_{u=1}^{K} \rho_{uv}^w$. Due to the uniform splitting of traffic stream in the first stage, $\rho_{uv}^w = \rho_{uv}/N$, for any w. Thus, the aggregate offered load to output buffer v of any PSE in the second stage becomes $\frac{1}{N} \sum_{u=1}^{K} \rho_{uv} = \frac{1}{N} \rho_v$. In other words, the aggregate traffic destined to PSE v is split equally among the second-stage PSEs. Since each PSE is assumed to be an output-buffered switch, output v of each second-stage PSE achieves a capacity of unity concurrently. This implies that the third-stage PSE v can achieve an aggregate capacity of N.

B. Switch Design

Each PSE forwards an incoming packet to its output based on the internal packet header information. Figure 7 shows the basic packet format internal to the switch. The header contains three additional fields that are generated at the ingress interface of each component link. Stage-k address identifies the output of a PSE in stage k.

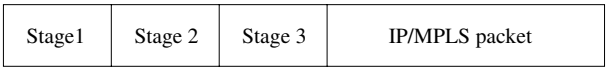

Fig. 7. Internal packet format.

We now show how the three fields are generated. Figure 8 shows the structure of the header processor together with the forwarding table (Label Information Base for MPLS or Forwarding Information Base for IP). An entry in the forwarding table is created when an LSP is established. An incoming MPLS packet is

indexed into the table via the incoming label to yield the stage-2 address for the second field of the internal packet header. The bundle # identifies the appropriate bundle within the PSE. However, it does not determine which component link the packet has to go through (We will describe how this determination is done below). The bundle # is encoded as (offset, mask), where the offset identifies the position of the first component in the bundle, and the mask determines the size of the bundle. As an example, suppose that a third-stage PSE with 16 links (numbered 0 to 15) are configured with four bundles. Links 0 and 1 are associated with bundle 1, links 2 and 3 with bundle 2, links 4 to 7 with bundle 3, and links 8 to 15 with bundle 4. A packet destined to bundle 3 would have the bundle # encoded as (4, 0x03), while a packet destined to bundle 4 would have the bundle # encoded as (8, 0x07).

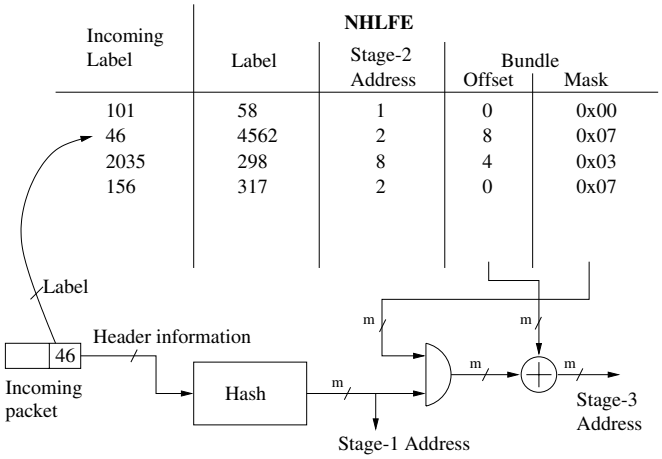

Fig. 8. Header processing.

When a packet label is being indexed into the forwarding table, the packet is simultaneously hashed based on some information in the packet header to yield the stage-1 address. We will show that the information can be based on a combination of IP source field, IP destination field, protocol field, source port field, and destination port field. For the case of MPLS packets, this information may be hashed once at an ingress LSR and the output is stored in a particular label (e.g., bottom label) of a label stack. In [10][11], hashing based on the CRC polynomial was compared to other hashing functions, and was found to offer superior results. We will examine the performance of CRC hashing function in detail in Section IV-A. We use m bits (where $m = log_2 N$[1]) of the hashed output bits to indicate one of the N possible output buffers in the first-stage PSE. The mask obtained from the forwarding table is used to select the d least significant bits of the hashed output, where 2^d corresponds to the size of the bundle. The selection is simply done by using a simple AND operation. This operation ensures each component link within a bundle receives approximately equal number of flows. The output of the AND operation is then added to the offset to yield the stage-3 address. Note that the simplicity of the header processor ensures that wire-speed performance can be achieved. Also note that the above operation ensures that packet ordering within a flow is preserved.

[1]Here, we assume that N is power of two. This assumption will be removed in Section V-A.

IV. Switch Performance

In this section, we investigate the packet-level performance of the switch. In particular, it is important to understand how the load-balancing operation affects the switch performance.

We use a discrete-event simulator to study various performance behavior. To emulate the actual operational behavior of the switch, we use IP traffic traces obtained from NLANR as the source model [12]. We choose the traffic traces from MAE-West interconnection at NASA-Ames which appears to closely reflect the traffic behavior in the core network. A trace contains about 90 seconds worth of traffic measurement on two interfaces of a router with a typical combined average packet rate of approximately 250 Mbps. The measurement information gives the timestamp of the packet, IP information, and transport-layer information. For the purpose of our investigation, we only need the 5-tuple flow information and the packet length information. Instead of using the timestamp of the packet as the arrival time, the simulator generates the arrival time of each packet so that different traffic loads can be easily experimented. We assume that packet inter-arrival time is exponentially distributed. This assumption has recently been shown to be suitable for the backbone network due to the high degree of aggregation of flows [13].

Our interest is in high-speed LSRs located in a backbone network where each component link typically operates at the speed of 10 Gbps. Using a single trace at such speed would significantly increase the rate at which the end-user application would send its packets. To mitigate the effect of speedup on the end-user flow behavior, we use 16 different traces and interleave the packets from different traces as the source model. For different sources, we do not use different sets of traces due to storage limitation. Instead, we use the same set but modify the source IP address of each packet by adding a fixed offset modulo the highest possible IP address value. This is done to approximately capture the varieties of packet origins associated with different sources. We also randomize the location of the first packet in the trace to reduce possible correlations that may exist among different sources.

In the following, we assume that each PSE is a 16×16 output-buffered packet switch (i.e., $N = K = 16$). Each first-stage (or third-stage) PSE has five bundles: two small bundles (Λ_1 and Λ_2) containing two component links each, and three large bundles (Λ_3, Λ_4 and Λ_5) containing four component links each. We consider the following simple traffic pattern where there is a mixture of point-to-multipoint and point-to-point traffic. We assume a scenario where each large input bundle has LSPs to all output bundles. In particular, incoming traffic to the large input bundle is split to each output bundle in proportion to the output bundle's capacity. We assume that the small input bundle Λ_1 (Λ_2) at the first-stage PSE u has only one LSP to the large output bundle Λ_4 (Λ_5) at the corresponding third-stage PSE u. In other words, incoming traffic to a small input bundle is entirely delivered to a unique large output bundle.

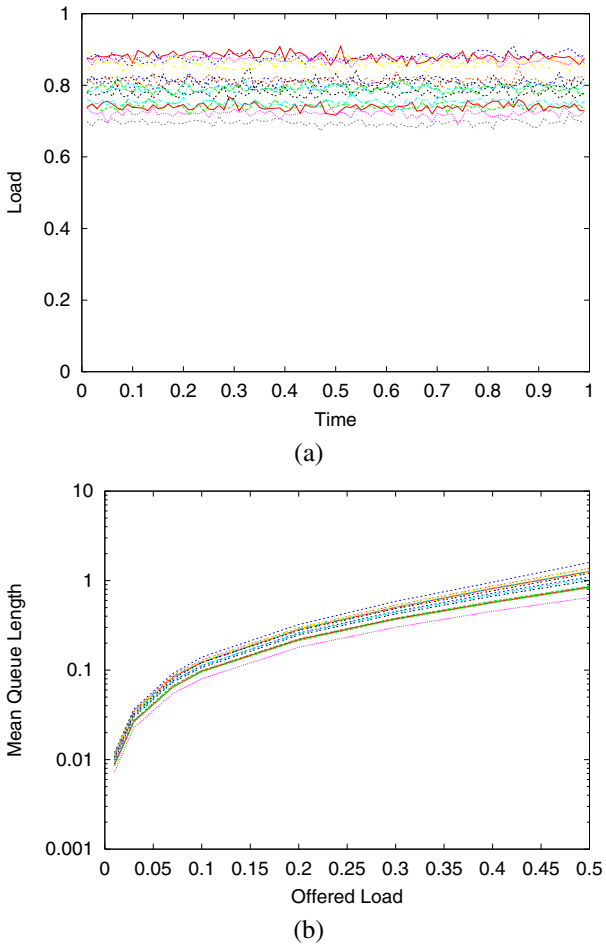

Fig. 9. Hashing based on destination IP address: (a) load, (b) mean queue length.

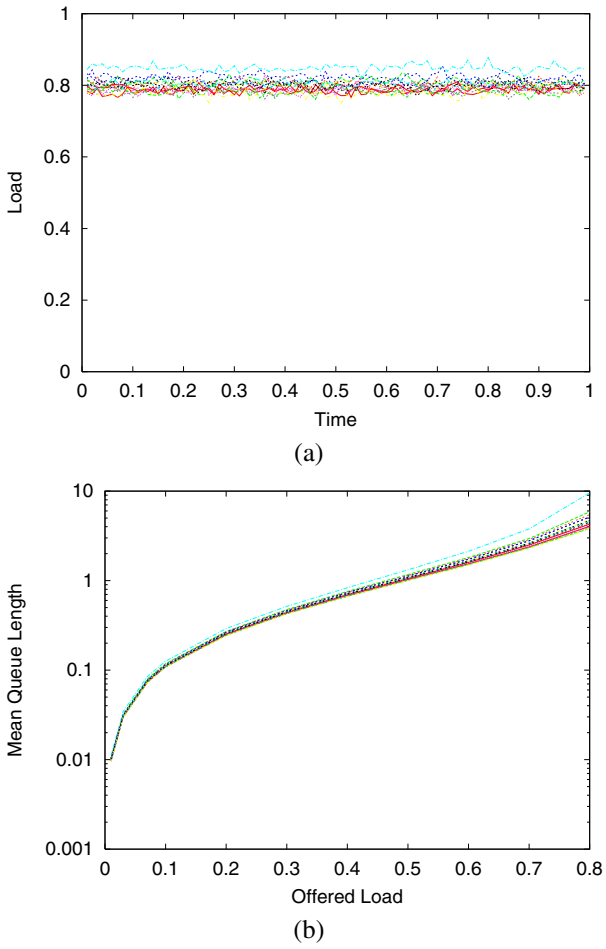

Fig. 10. Hashing based on source and destination IP addresses: (a) load, (b) mean queue length.

A. *Load Balancing*

We now investigate the load-balancing performance of the switch. There are three measurement points of interest as far as load balancing is concerned. In the first stage, the load on each output buffer in PSE j should be ideally equal. Next, the load on a particular output buffer k on each PSE in the second stage should be ideally equal. Finally, the load on each output buffer associated with the same bundle in the third stage should be ideally equal. Let $OB(i, j, k)$ denote output buffer k of PSE j in stage i. We assume that the normalized offered load to a large input bundle ρ_L, and to a small input bundle ρ_S are equal in this section.

We first focus on the load balancing performance at $OB(1, u, 1)$ - $OB(1, u, 16)$ for a particular PSE u in the first stage. We consider the following inputs to the hash function:

1. hashing based on destination IP address,
2. hashing based on source and destination IP addresses, and
3. hashing based on the 5-tuple IP flow information (i.e., source IP address, destination IP address, protocol, source port, and destination port).

Figure 9 shows that hashing based on only destination IP address may lead to unbalanced loads. In particular, the offered loads may vary from 0.7 to 0.9 (see Figure 9(a)), resulting in unbalanced queue performance (see Figure 9(b)). A marked improvement in hashing performance can be gained by adding the source IP address to the hashing input, as indicated in Figure 10. Notice that the variability in the loads decreases appreciably, and the queue performance becomes more alike. Notice, however, that a particular buffer consistently receives more traffic than the others (as indicated by the blue line). This anomaly appears to be caused by coincident hot spots among several nodes. When 5-tuple hashing is used as shown in Figure 11, the mean queue lengths become remarkably close to each other. This figure indicates that hot spots do not occur at the micro-flow level.

We complete this section by demonstrating the load-balancing performance of the switch at other stages using the 5-tuple hashing. Figure 12 shows the load-balancing performance at $OB(2, 1, k)$ - $OB(2, 16, k)$, for a particular output buffer k. As expected, we see good load balancing for a particular output buffer across all PSEs because of the symmetry in both the switch topology and the traffic split. Finally, Figure 13 shows the load-balancing performance at $OB(3, v, 13)$ - $OB(3, v, 16)$, for a PSE v. The figure also demonstrates good load-balancing performance across all component links in a bundle.

B. *Short-Term Imbalance*

To characterize the short term variations among the loads, we observe the traffic loads in each time slot $t \in 1, 2, \cdots$ at the

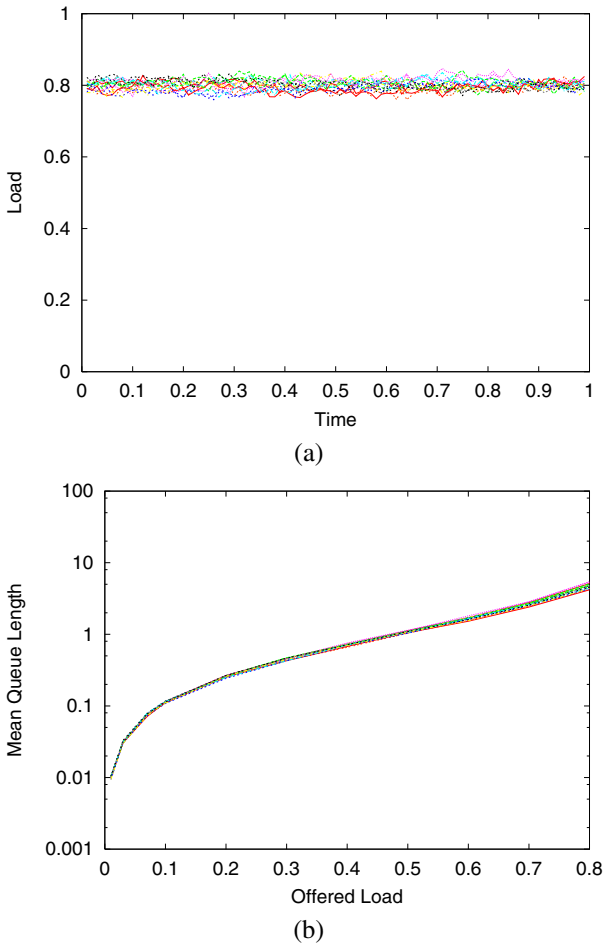

Fig. 11. Hashing based on 5-tuple IP flow: (a) load, (b) mean queue length.

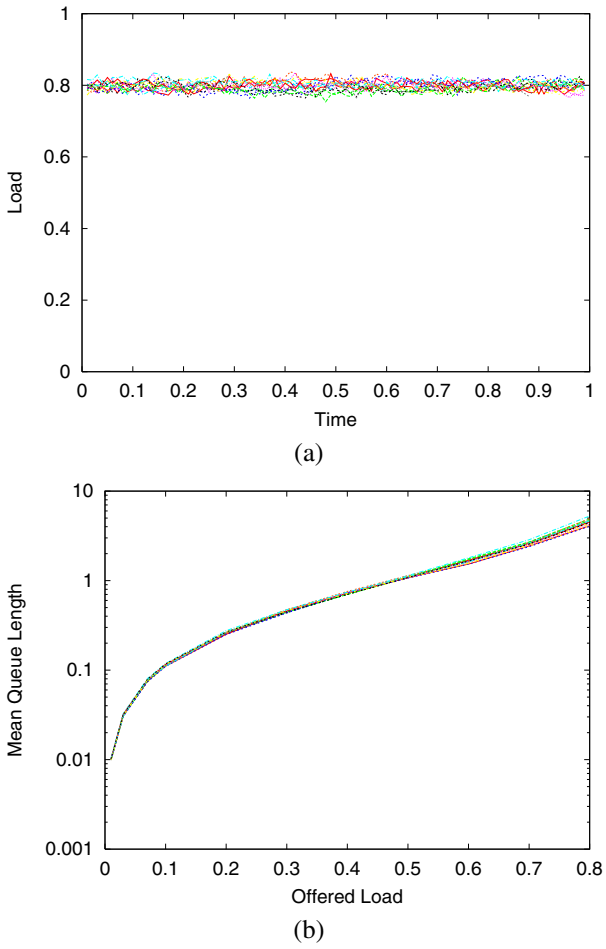

Fig. 12. Stage 2: (a) load, (b) mean queue length.

measurement points of interest. Let $\bar{\gamma}(t)$ be the average load in time slot t, and let $\gamma_{max}(t)$ be the maximum load in time slot t. We define the short-term imbalance in time slot t as $I(t) = \gamma_{max}(t)/\bar{\gamma}(t)$. A perfectly balanced system would have a measure of $I(t) = 1$, which can only occur in a fluid model if the time slot is relatively small.

Figure 14(a) plots the short-term imbalance as a function of time for the case where $\rho_L = \rho_S = 0.8$, and the time slot duration is equal to 10 msec. Notice that the maximum load is consistently within 5 percent of the average load.

Let $E[I(t)]$ be the average short-term imbalance over a long period of time. Figure 14(b) shows the effect of the offered load on $E[I(t)]$ for the case when the time slot duration is 10ms and 100ms. As expected, the longer time slot duration provides better load balancing due to the smoothing effect of momentary traffic fluctuation and traffic burstiness. In both cases, the load balancing performs better when the offered load increases, showing that the switch is more robust as it becomes more stressed.

C. Delay-Throughput Performance

We now study the delay-throughput performance of our switch compared to a switch implementing plain LB. In particular, it is of interest to investigate the delay performance of the point-to-point traffic from input bundle Λ_2 to output bundle Λ_5. Throughout the simulation, we fix ρ_L to 0.8. If $\rho_S = 0$, then the throughput at the large output bundle Λ_5 is equal to 0.6 since $1/4$ of the traffic from a large output bundle is destined to two small output bundles. This implies that the average residual bandwidth at the output bundle is $(1-0.6) \times 4 \times 10 = 16$ Gbps.

As we vary ρ_S, we monitor the delay of the point-to-point traffic at the output bundle Λ_5. Figure 15 plots the switch delay against throughput for the point-to-point traffic. Notice that the entire residual capacity can be used for the point-to-point traffic with LB/DA. On the other hand, LB requires each component of an input bundle to map to each component of an output bundle. As a result, each component can only access up to 4 Gbps of the residual bandwidth. Thus, the total throughput that can be achieved by the input bundle of size two is 8 Gbps.

V. Switch Extensions

In this section, we explore possible extensions to the basic switch architecture.

A. Large Bundle Size

The basic architecture described early assumes that a bundle has to be contained within a single PSE in the first/third stage. As a consequence, a large bundle size (e.g., with 128 component links) may lead to a costly architecture since the PSE in

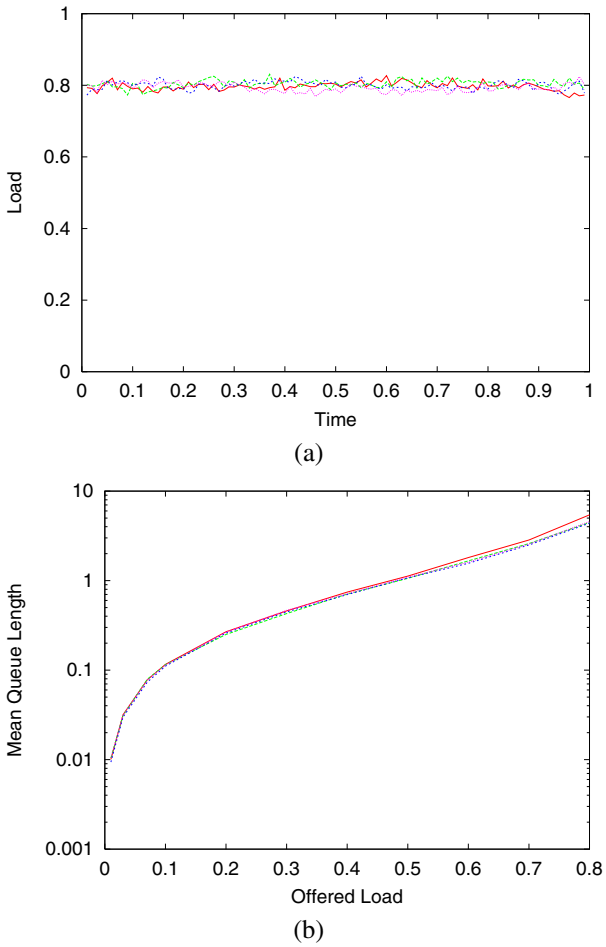

Fig. 13. Stage 3: (a) load, (b) mean queue length.

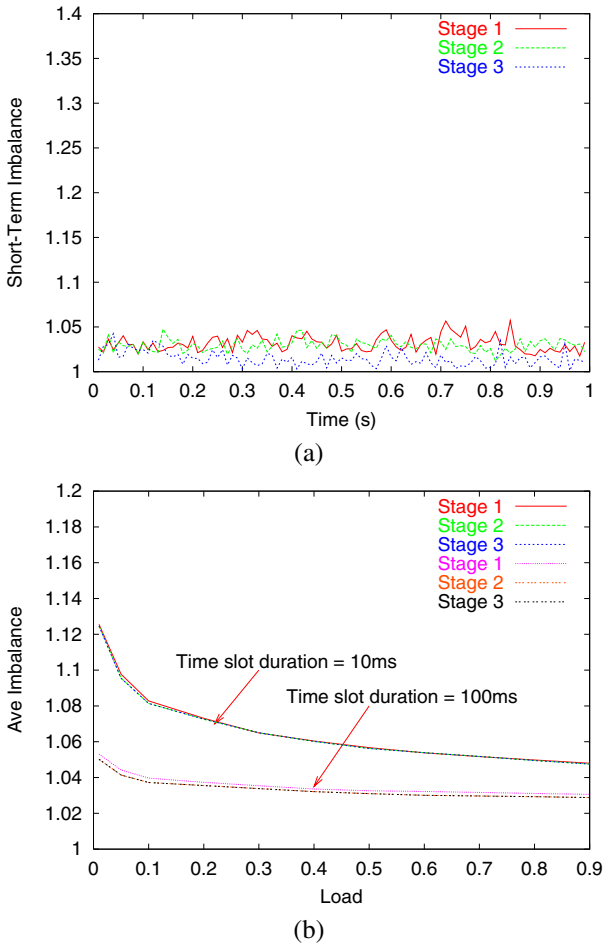

Fig. 14. Imbalance: (a) short-term imbalance, (b) average short-term imbalance.

the first/third stage must be of size no less than the bundle size (128×128 in this example).

Figure 16 shows how a cost-effective switch can still be built by having a large bundle that spans multiple smaller PSEs. Traffic streams that are destined to multiple sub-bundles of the same bundle are split at the second-stage PSEs. This extension can be easily implemented by modifying the forwarding table so that the stage-2 address is encoded as (offset, mask), similar to the encoding of the bundle identification described in Section III-B. The extension is very simple if the PSE in the second stage splits its traffic equally. In such a case, the sub-bundles have to be of the same size.

B. Upgrade Strategy

It is generally desirable for service providers to be able to upgrade their LSRs incrementally to deal with traffic growth, so that total equipment expenditure can be dispersed over the lifetime of the LSR. In this section, we investigate our switch upgrade strategy based on the initial and ultimate router requirements from the time a router is acquired until the time it is fully equipped.

Let us define the following notations:

- t_l: sustainable period, which is the period during which the switch can be grown to meet traffic growth.
- c_i: initial aggregate capacity of an LSR in number of component links.
- r: annual traffic growth rate.

Then the ultimate aggregate capacity of an LSR, c_f is given by

$$c_f = c_i(1+r)^{t_l}.$$

Suppose that $c_i = 15$, $t_l = 6$, and $r = 60\%$, then $c_f = 251$. Thus, the switch should be expandable so that it can support up to 256 ports. Possible combinations of PSE sizes include $(N = 16, K = 16)$, $(N = 32, K = 8)$ and $(N = 8, K = 32)$.

The switch architecture can also be made flexible so that additional hardware may be added on an as-needed basis, as shown in Figure 17. In a typical design, each pair of PSEs in the first and third stages reside in one shelf (19" or 23" wide). Thus, in an early deployment stage, only a few of such shelves may be needed. The second-stage PSEs are needed at all time. However, it is possible to design a PSE of size $K \times K$ with a distributed architecture so that only $k \leq K$ line cards are needed if only k inputs/outputs are used (see Figure 17).

C. Switch Protection

In certain applications, it may be critical for the second-stage PSEs to be reliable since traffic from any component link traverses through each of these PSEs. Therefore, these PSEs may

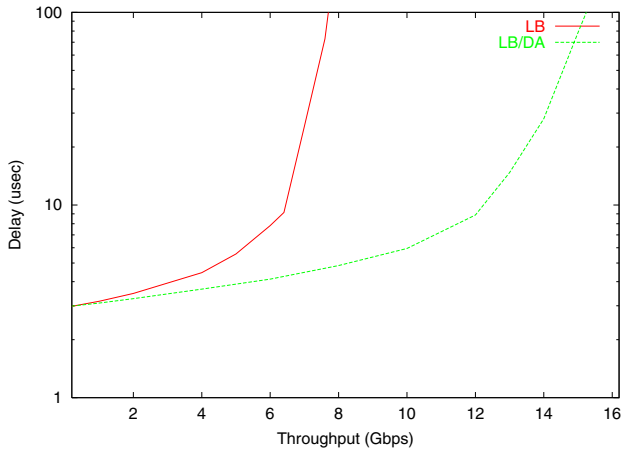

Fig. 15. Switch delay for point-to-point traffic.

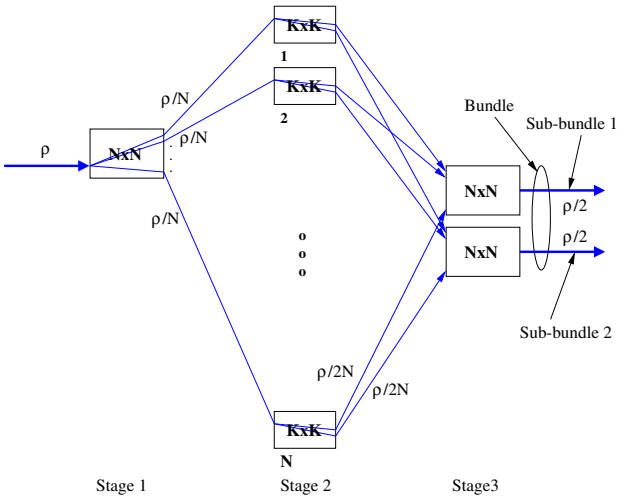

Fig. 16. Large bundle size.

need to be protected. For economic reason, the other PSEs may be left unprotected. Instead, a failure in one of these PSEs would trigger a higher-layer mechanism (e.g., MPLS path protection) to perform the recovery. In this section, we focus on redundancy in the second stage, and assume a single-failure scenario (i.e., only one PSE will fail at a given moment of time).

Under normal operating condition, the first-stage PSE splits its traffic to all its outputs equally, as shown in Figure 18(a). Note that only $N-1$ inputs can be used to avoid overload under failure condition. When one of the PSEs in the second stage fails, that failed PSE is avoided. As a result, each PSE in the first stage changes its splitting ratio among the other working second-stage PSEs, as shown in Figure 18(b). Unlike $1:n$ protection approach where a spare PSE is normally placed in a standby mode and used only when a failure occurs, our approach makes use of each working PSE in the second stage. The main advantage is that the second stage appears to have a speedup of $N/(N-1)$ under normal condition, making the switch more tolerant to traffic imbalances.

Note that a transition from normal to failure conditions requires that the traffic split in the first stage be modified from N to $N-1$. To realize this, we propose a "weighted-hash array"

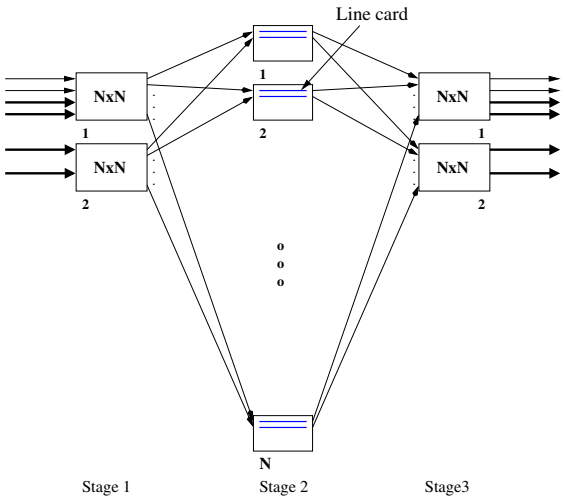

Fig. 17. "Pay-as-you-grow" approach.

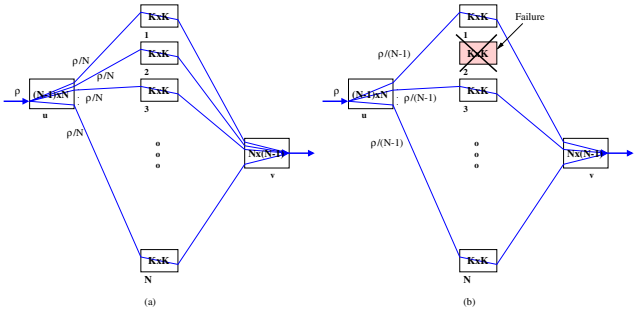

Fig. 18. Protected second stage: (a) normal condition, (b) failure condition.

which records the values of possible outputs in the first stage. The output of the CRC-32 hashing function is used to index the array to select a particular output associated with a particular packet.

Figure 19 shows an example of the array of length 16. It is assumed that $N=4$ so that each possible output is recorded four times (independent of the locations) under uniform splitting. When the switch is operating normally, a hashed output's 0x0C would select output 3 (see Figure 19 (a)). If the second-stage PSE that is connected to output 3 of the first-stage PSE fails, the elements of the array containing output 3 would be replaced with other values (see Figure 19 (b)). Observe that packets with a hashed output's 0x0C would now select output 0 instead.

In the example, output 0 receives $6/16$ of the traffic while outputs 1 and 2 each receives $5/16$ of the traffic. Such imbalance can be minimized by increasing the length of the array (e.g. to 256). Finally, note that this approach can be used to implement weighted hashing whenever non-uniform traffic splitting is desired.

D. Support for Non-IP Traffic

Previous discussions assume that the traffic entering the switch is completely IP-based. In some environments, an LSP may have to carry non-IP traffic across an MPLS network. Possible services where this situation may arise include circuit emulation and layer-2 VPN. We point out, however, that non-IP

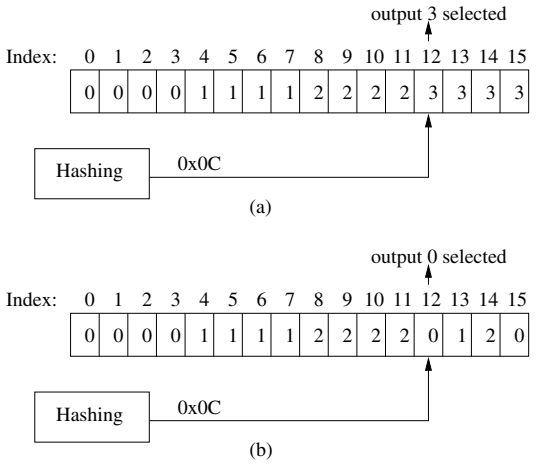

Fig. 19. Protected second stage: (a) normal condition: output 3 selected, (b) failure condition: output 0 selected.

traffic typically constitutes a small portion of the total traffic in an MPLS network. A full treatment of this topic is outside the scope of this paper, and here we only give a brief outline of possible approaches to minimize switch imbalances. A forthcoming paper will discuss detailed algorithms, implementation issues, and performance impacts.

For non-IP traffic, the IP flow information is not available in the MPLS packet, and thus hashing cannot be used. To this end, we propose to use the basic link bundling for establishing an LSP with non-IP traffic. Inside the switch, the associated non-IP traffic stream for a given LSP is routed through the same path. Figure 20 shows how the non-IP traffic may cause potential congestion in the first and second stages because of traffic imbalances. Clearly, this requires IP traffic to be re-distributed to ensure that the switch is as balanced as possible.

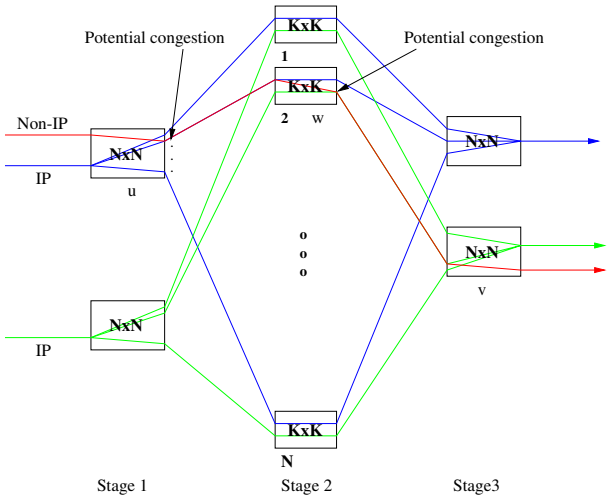

Fig. 20. Support for non-IP traffic.

The basic idea is that support for non-IP traffic requires some PSEs to split its incoming traffic unevenly, which implies the need for weighted hashing, as described early. Suppose that a non-IP LSP request with bandwidth B arrives at PSE u in the first stage, and is to be routed through PSE's output i. Let ϕ_i represent the hashing weight on output i, so that output i has the load of $\frac{\phi_i}{\sum_{j \neq i} \phi_j}$. Then one possible approach is to equalize the loads in PSE u by decreasing the weight on output i by $\Delta\phi_i \equiv max(\frac{N-1}{N}B, \phi_i)$, and by increasing the weights on other outputs $j \neq i$ by $\Delta\phi_i/(N-1)$.

Although the above approach equalizes the loads at PSE u, it fails to consider the congestion status at PSE w in the second stage. An alternative approach is to adjust the weights at PSE u so that the bottleneck at both PSE u and PSE w is minimized. Another more elaborate approach is to also consider other PSEs in the first stage so that the congestion status of all PSEs in the second stage are taken into consideration.

VI. Conclusion

The proposed link bundling with distributed assignment (LB/DA) is more efficient than the basic link bundling (LB). In particular, LB/DA provides higher revenue due to lower blocking of LSP bandwidth requests compared to LB. We have proposed an IP switch architecture that is capable of effectively distributing traffic inside the switch and among the component links within a bundle to support pipes of bandwidth higher than the line (component link) rate. The design relies on hashing to preserve packet sequence without costly state information or complex co-ordination algorithm. By means of simulation using real IP traffic traces, we showed that hashing based on 2-tuple IP flow information achieves good load-balancing performance, and the 5-tuple information gives the best load-balancing performance. Finally, we described various extensions of the basic switch architectures. Future work will focus on the details of these extensions.

References

[1] E. Rosen, A. Viswanathan and R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031, Jan. 2001

[2] Qwest IP Network, http://www.qwest.com/about/qwest/network/nationip.html.

[3] Genuity Network Infrastructure, http://www.genuity.com/infrastructure/us_back100.htm.

[4] AT&T IP Backbone Network, http://www.ipservices.att.com/ backbone/.

[5] S. Iyer, A. Awadallah and N. McKeown, "Analysis of a Packet Switch with Memories Running Slower than the Line Rate," in *Proceedings of INFOCOM'2000*, Israel, Mar. 2000.

[6] S. Iyer and N. McKeown, "Making Parallel Packet Switches Practical," in *Proceedings of INFOCOM'2001*, Alaska, Apr. 2001.

[7] K. Kompella, Y. rekhter and L. Berger, "Link Bundling in MPLS Traffic Engineering," Internet Draft <draft-ietf-mpls-bundle-04.txt>, work-in-progress, Jul. 2002.

[8] D. O. Awduche et al., "RSVP-TE: Extensions to RSVP for LSP Tunnels," RFC 3209, Dec. 2001.

[9] C. Clos, "A Study of Nonblocking Switching Networks," *BSTJ*, vol. 32, no. 2, pp. 406–424, Mar. 1953.

[10] R. Jain, "A Comparison of Hashing Schemes for Address Lookup in Computer Networks," *IEEE Trans. Commun.*, pp. 1570-1573, Oct. 1992.

[11] Z. Cao, Z. Wang and E. Zegura, "Performance of Hashing-Based Schemes for Internet Load Balancing," in *Proceedings of INFOCOM'2000*, Israel, Mar. 2000.

[12] NLANR Network Traffic Packet Header Traces, http://moat.nlanr.net/Traces/.

[13] J. Cao, W. S. Cleveland, D. Lin and D. X. Sun, "Internet Traffic Tends to Poisson and Independence as the Load Increases," Technical Report, Bell Labs, 2001.