CHAPTER 13

MULTI-PLANE MULTI-STAGE BUFFERED SWITCH

To keep pace with Internet traffic growth, researchers have been continually exploring new switch architectures with new electronic and optical device technologies to design a packet switch that is cost-effective and scalable to a very large capacity, for example, a few hundred tera bps or even a few peta bps [1].

Packet switches can be classified into single-stage switches and multi-stage switches. A single-stage switch fabric, such as a cross point switch, has only one path between each input–output port pair. Thus, it has no packet out-of-sequence problem and is relatively simple to design. However, due to the quadratic relationship between the switch size and the number of switching elements, it is not very scalable.

Multi-stage switch architectures, such as the Clos [2] or Benes networks, utilize multiple paths between each input–output pair with fewer switching elements allowing scalable and fault-tolerant designs. A multi-path switch fabric can be further classified into a memory-less switch fabric or a buffered switch fabric. A memory-less multi-path switch fabric has no packet out-of-sequence problem because the propagation delays through different paths are comparable. However, it requires the internal link to be resolved and output port contention by a centralized packet schedule with global information available, causing the control plane of the switch fabric to be unscalable in spite of a scalable data plane.

A buffered multi-path switch fabric does not require a centralized packet scheduler as contention is resolved between stages by using memories to temporarily store packets that lost contention for links or ports. Due to the nature of unbalanced incoming traffic, the queue lengths on the multiple paths between any input–output pair are very likely different. As a result, packets from the same flow may experience different delays while traversing different paths and cause packets to be out of sequence.

This chapter introduces ultra-scalable multi-plane multi-stage buffered switch architecture, called TrueWay, based on Clos-network. The Ciscos CRS-1 system [3] also uses a

High Performance Switches and Routers, by H. Jonathan Chao and Bin Liu

multi-plane multi-stage buffered switch based on Benes network. The more switch planes and stages, the more routing paths that exist between any input–output pair.

Thus, there are several challenging design issues related to designing the TrueWay switch:

- How to efficiently allocate and share the limited on-chip memories?
- How to intelligently schedule packets on multiple paths while maximizing the memory utilization and system performance?
- How to minimize link congestion and prevent buffer overflow (i.e., stage-to-stage flow control)?
- How to maintain packet order if they are delivered over multiple paths (i.e., port-to-port flow control)?

13.1 TRUEWAY SWITCH ARCHITECTURE

Figure 13.1 shows the TrueWay switch architecture [4]. The ingress traffic manager (TMI) distributes packets to different paths of different switch planes while the egress traffic manager (TME) collects packets traversing the switch fabric and buffers them to be transmitted to the network. The switch fabric consists of p switch planes, each consisting of a three-stage Clos network. The study shows that a speedup of 1.6 provides very good performance. The switch can also support multiple priorities (e.g., $q = 2$).

The modules in the first, second, and third stages are denoted as input modules (IMs), center modules (CMs), and output modules (OMs). Each module can be logically considered a crosspoint buffered switch. The first stage of the switch plane consists of k IMs, each with n inputs and m outputs. The second stage consists of m CMs, each with k inputs and k outputs. The third stage consists of k OMs, each with m inputs and n outputs.

Similar to conventional packet switches, each incoming packet is segmented into multiple fixed-length cells. The cells are then routed through the switch fabric independently and

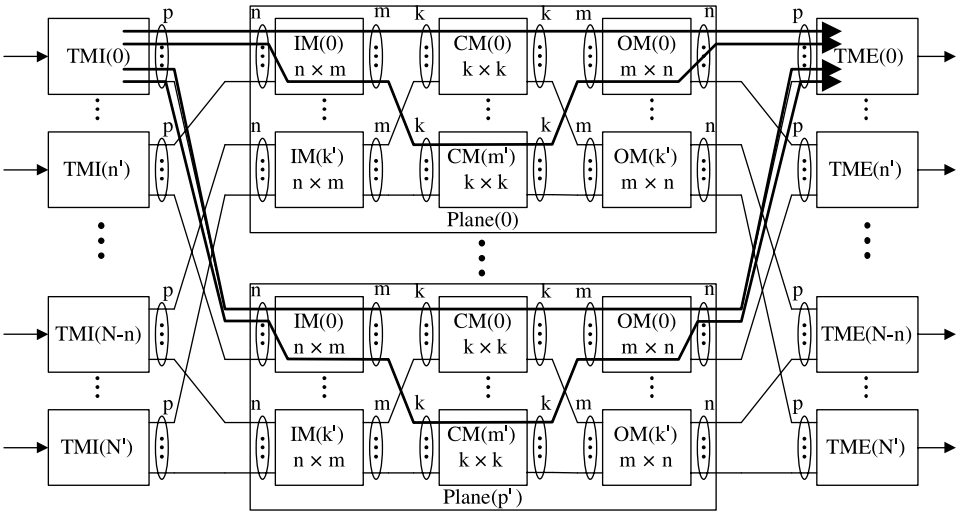


Figure 13.1 TrueWay switch architecture.

reassembled into packets using reassembly queues at the output ports. Cells are served using strict priority: Only when cells in the higher priority queues are completely served, can the cells in the lower priority queues be served.

13.1.1 Stages of the Switch

Each TMI has two types of queues, virtual output queue (VOQ) and virtual path queue (VPQ). Using two different queues at the TMI can avoid throughput degradation due to head-of-line (HOL) blocking. As shown in Figure 13.2, when a packet arrives at a TMI, it is segmented into cells and put into the corresponding VOQ. The packet in the VOQ, if a HOL packet, is moved to the VPQ depending on its Path ID (PID), and is used to determine the routing path consisting of the switch plane and CM numbers. Each output link of the TMI selects a VPQ and sends the HOL packet to the IM based on the scheduling scheme described later in this chapter.

Figure 13.3 illustrates the TME structure. When cells arrive at the TME, they are reassembled into a packet in the Reassembly Queue (RAQ). Once the RAQ has a complete packet, the pointers of the linked cells are moved from the RAQ to the class of service queue (CSQ). Since more than one RAQ may have a complete packet, packets can be served in a round-robin fashion to achieve fairness.

To avoid cell loss at the RAQ, TME sends flow control signals to the OM if the number of cells in the RAQ exceeds a threshold. Since the TME and OM are located at the same shelf, the round trip time (RTT) delay is small. Therefore, the RAQ size should only be greater than the largest packet size (e.g., 9 kB or 174 cells) plus the RTT delay (e.g., four cells) for 178 cells. Each cell is assumed to be 64 bytes with a payload of 52 bytes.

The term switch modules (SMs) refers to the IM, CM, or OM. Here, we assume $n = m = k$ to simplify the discussion. Each module has n^2 crosspoints and each crosspoint has q queues, each of which corresponds to a priority level. Therefore, each SM has $n^2 \times q$

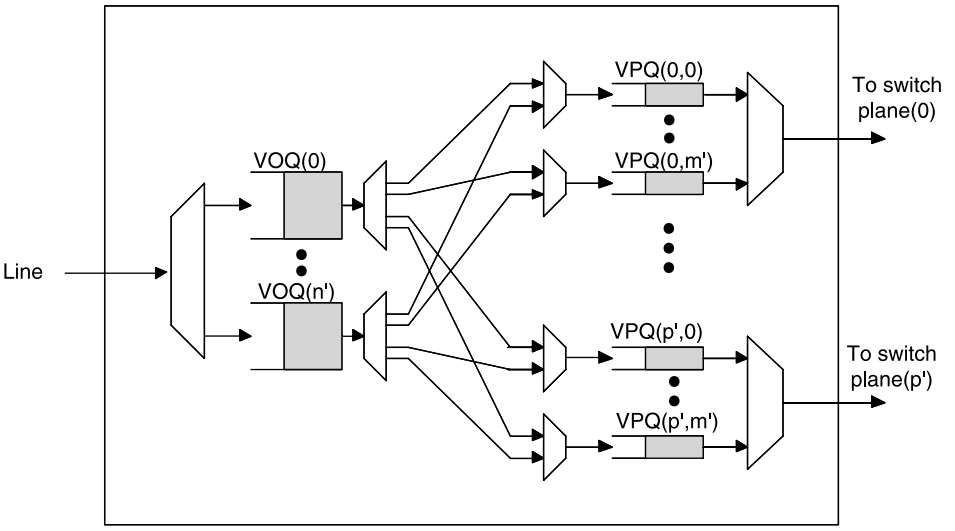


Figure 13.2 Ingress traffic manager (TMI) structure, where $n' = n - 1$, $m' = m - 1$, $p' = p - 1$.

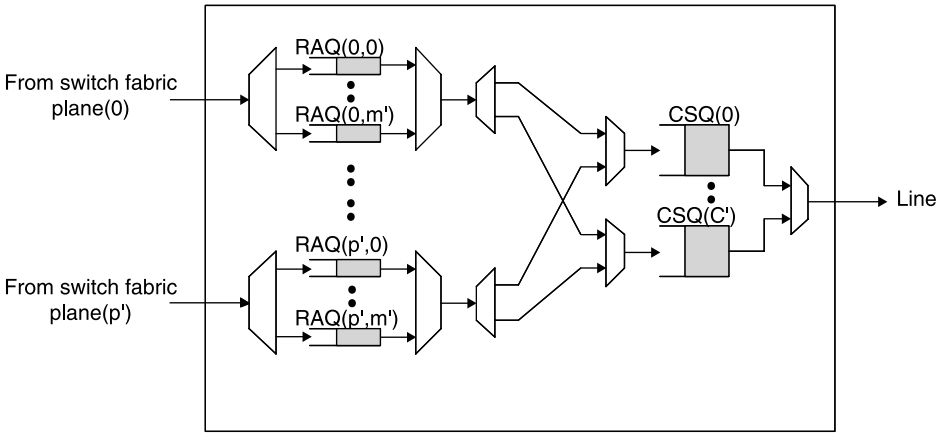


Figure 13.3 Egress traffic manager (TME) structure.

queues. Each queue can receive and send one cell in each time slot. An incoming cell is stored in one of the queues according to its destination address and priority level.

Memory is the cost-consuming component of the high-speed switch. Thus, the efficient memory allocation will not only affect system performance, but also influence the cost of design. From the memory organization point-of-view, the SM can be crosspoint buffered, output-shared, all-shared, or input-shared memory as shown in Figure 13.4.

In the crosspoint buffered switch, each queue has its dedicated memory. Since there is no memory sharing, this approach requires the largest memory. In the output-shared architecture, all the vertical queues of each output are grouped together in a single output memory. With stage-to-stage flow control (to be described later), each queue needs to

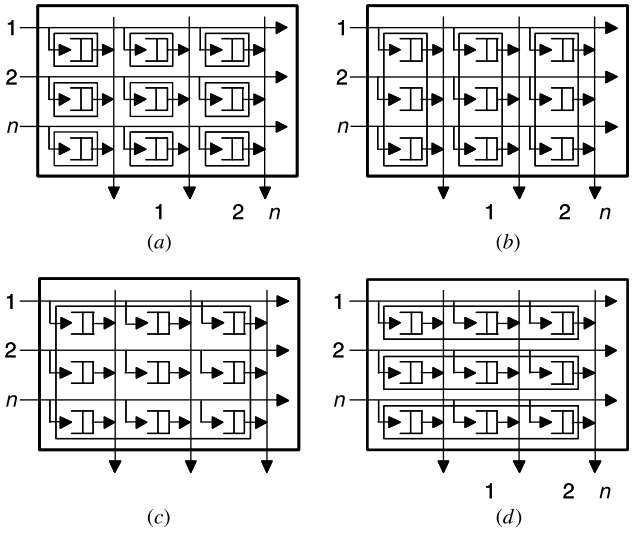


Figure 13.4 Different memory organizations for the switch modules: (*a*) Crosspoint buffer memory; (*b*) Output-shared memory; (*c*) All-shared memory; (*d*) Input shared memory.

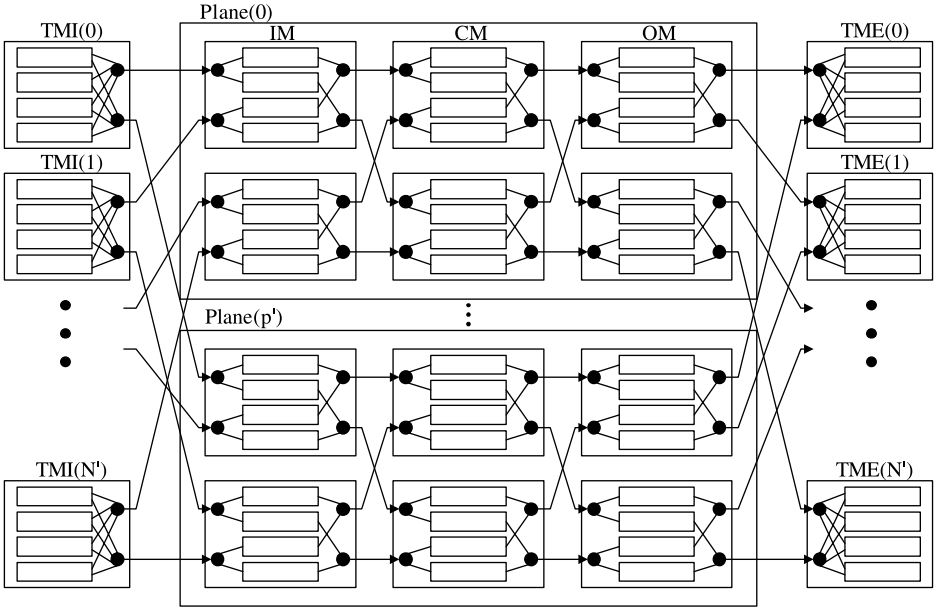


Figure 13.5 Queue structure in the switch.

accommodate at least the RTT between stages. With a few cells of RTT and an SM size of 64, each output memory needs to hold more than a few hundred cells. As a result, the total memory size of each SM is required to be more than 10,000 cells, or close to 1 Mbits, which is too large for a switch chip.

In the all-shared memory SM, the queues are shared by the entire SM, which results in the following two problems. One is the bottleneck of memory speed. The memory in the all-shared memory SM has to write n cells and read n cells within a time slot, which is not feasible for high speed link rate. Another is the fairness of sharing. Under hot-spot traffic, cells destined for the hot-spot port can monopolize the buffer and the other cells destined for non-hot-spot ports can be blocked.

One way to mitigate the problems of the all-shared memory SM is to partition the memory into buffers to allow each buffer to be shared by n queues, associated with the input link to make the flow control simple. Input-shared SM is a good choice because each buffer receives at most one cell per time slot. Although the buffer consists of n queues, it is likely that most queues are empty. Therefore, the buffer size does not need to be equal to the sum of n queues. The switch architecture showing SMs with input-shared queuing structure is illustrated in Figure 13.5.

13.2 PACKET SCHEDULING

In a typical multi-plane multi-stage switch, the number of reassembly queues is equal to the product of the number of inputs, the number of paths between any input–output pair, and the number of priorities. In this section, we discuss some packet interleaving schemes for the TrueWay switch to reduce the number of reassembly queues to the number of paths $\times$ the

number of priorities, resulting in a reassembly buffer size independent of the switch size. The proposed schemes also offer high throughput and fairness, while maintaining cell integrity.

For the sake of simplicity, we assume each output link of the TMI, IM, CM, and OM has the same packet scheduling policy. A cell is transferred from a queue at the upstream side to a queue at the downstream side in the switch. The queue at the upstream side is called a source queue (SQ) and the queue at the downstream side is called a destination queue (DQ). Cells waiting at an SQ compete for the output link, since one link can send at most one cell in a timeslot. Scheduling schemes are employed to select which cell to send.

The straightforward way is to schedule cells in a round robin (RR) manner. It is called complete cell interleaving (CCI), which will give the best load balancing among possible paths and minimum cell transmission delays through the switch fabric. However, CCI requires a large number of RAQs, since at a given time, a DQ may have cells coming from various inputs over different paths, waiting to be reassembled. An example of a CCI algorithm in action is shown in Figure 13.6*a*. Although, all cells are transferred with a minimum delay, the requirement for separate RAQs, one for each input, is clear at DQ(X).

If packet boundaries are taken into account when scheduling (i.e., wait until a packet is completely sent before sending cells from another packet), this algorithm is called complete packet interleaving (CPI).The number of RAQs required is one per plane per priority, which is much less than the CCI scheme. The price paid here is the degraded throughput, since some output queues may be idle while waiting for the last cells of an incomplete packet

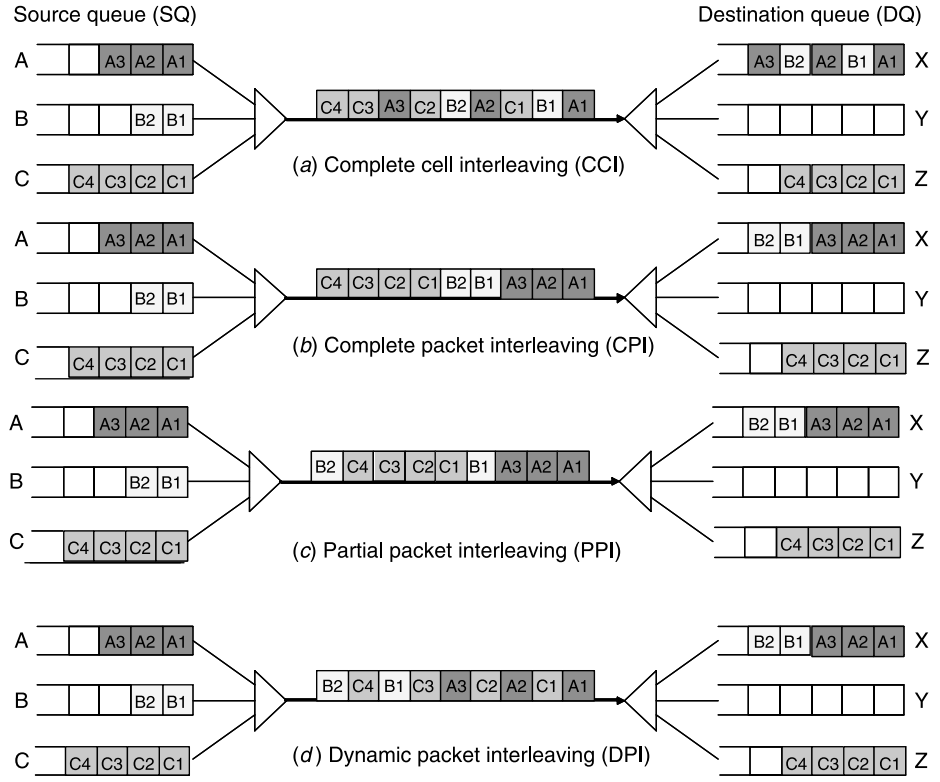


Figure 13.6 Packet scheduling schemes.

to arrive, although there are cells destined for the idle output. Figure 13.6*b* illustrates the CPI. Unlike the CCI example of Figure 13.6*a*, here cells from different inputs arrive at the DQ(X) one after another (i.e., not intermingled), so there is no need to hold separate queues per input.

In addition to the degraded throughput, the packet-interleaving scheme also suffers from a possible deadlock situation. Let us assume that all but the last cell of packet P_1 is sent from the TMI to the TME. When the last cell of P_1 takes its turn, its transmission can be blocked because the buffer at the IM can be full of other cells coming from different VOQs. At the same time, the buffers at the IM can be full of fresh packets destined for the TME, to which the partial packet is destined. No fresh packets at the IM can be sent because they are partial packets. In the worst case, there can be a situation where all partial packets at the TMI are blocked due to the full buffers at the IM and all fresh packets at the IM are blocked because of the partial packets. Then all cells in the switch fabric are blocked, which results in a deadlock situation. In practice, the line cards have more memory than the switch fabric cards. So, the deadlock case generally occurs due to the lack of memory in switch modules. To avoid the deadlock situation, at least one cell space in the SM buffers should be reserved for all partial packets. Then any partial packet can forward a cell to the next stage and there will never be a deadlock situation.

In the following two subsections, we present two packet-interleaving algorithms, namely, partial packet interleaving and dynamic packet interleaving. These schemes allow high throughput while keeping the RAQ sizes comparably small. The deadlock situation is avoided by reserving at least one free memory space in the buffer for partial packets. Then any partial packet can forward a cell to the next stage, without causing deadlock. The details on how to reserve memory space to avoid deadlock is discussed in Section 13.3.

13.2.1 Partial Packet Interleaving (PPI)

The CCI scheme is attractive from the load-balancing point of view and the CPI scheme is attractive from the reassembly point of view. To take advantage of both the CCI scheme and the CPI scheme, we consider the partial packet interleaving (PPI) scheme, which stands between the CCI scheme and the CPI scheme.

PPI is similar to CPI. Once an SQ(A) starts sending a packet, it keeps sending the packet without moving to another SQ. However, unlike CPI, if SQ(A) becomes empty at some point before completing the transfer of this packet, other SQs with pending packets to other DQs will become eligible to send their packets. In this scheme, cells from packets destined to different DQs are allowed to be interleaved, but not those destined for the same DQ. Figure 13.6*c* illustrates the PPI. In this example, let us assume cell C_1 arrives at the SQ(C), after cell B_1 arrives at the SQ(B) but before the arrival of cell B_2. If the CPI algorithm is used, none of the cells of packet P could be delivered until B_2 arrives (as in Fig. 13.6*b*), but PPI allows packet C to use the link while it is idle (i.e., waiting for cell B_2) thus improving throughput. As a result, idle time is reduced so the throughput is improved, while the required number of RAQs is still comparable to the CPI scheme.

13.2.2 Dynamic Packet Interleaving (DPI)

In the PPI scheme, once an SQ(A) is selected, another SQ(B) can only be selected if the packet in SQ(A) is completed or SQ(A) becomes empty, whichever happens first. The dynamic packet interleaving scheme (DPI) moves to the next eligible SQ as soon as it sends

TABLE 13.1 Number of RAQs Required at the TME for Different Scheduling Algorithms

Scheduling Scheme	Number of Reassembly Queues
CCI	$p \times q \times m \times n \times k$
PPI or DPI	$p \times q \times m$
CPI	$p \times q$

a cell from SQ(A). Figure 13.6*d* illustrates the operation of DPI. Assume that the SQs have a uniform arrival rate. Then in the CPI case, packets will be transferred one after another resulting in low throughput. In the PPI case, although the throughput will be similar to DPI, the load balancing for the following stage will be better for the DPI case. Thus, DPI is closer to the CCI scheme, in that it has a higher overall throughput. Fortunately, the number of RAQs is still the same as in the PPI scheme. Table 13.1 shows the number of RAQ requirements for each of the algorithms described above.

13.2.3 Head-of-Line (HOL) Blocking

As long as queues contain cells going to different destinations, HOL blocking (the blocking of a cell destined for an idle destination) is inevitable. For the TrueWay switch, the final destinations are the RAQs at the TMEs. In each OM, if each crosspoint has one queue per each RAQ, there will be no HOL blocking. For the CM, the immediate destinations are the queues at the OM. Since each crosspoint in the OM has as many queues as the total number of RAQs in the TMEs, the number of queues in each crosspoint of the CM increases very quickly. The effect is even more severe for the earlier stages (TMI and IM). In order to limit the number of queues in TMI and IM, HOL blocking should be allowed up to some extent to make the design feasible.

In the TMI, there are two kind of queues, VOQ and VPQ. The latter facilitates the cell distribution to different planes/CMs. The former keeps most cells of each flow (traffic of each input–output pair) and only sends one cell at a time to the VPQ, preventing the congested flow from sending too many cells and thus clogging the routing paths through the CMs.

13.3 STAGE-TO-STAGE FLOW CONTROL

It is a common practice to have a large memory at line cards and a small memory at switch fabric connection points. When a cell is transmitted over a link, the receiver should have free memory space to store the cell until it is transmitted to the next stage. Since the receiver has limited memory space, if the receiver memory becomes full, the sender must hold the cells until the receiver has free space. Therefore, a flow control mechanism has to be implemented to make sure there is no cell loss in the switch fabric. In this section, two well known flow control schemes: back-pressure and credit-based flow control (also known as N23), are first described, followed by a new flow control scheme dedicated to the TrueWay switch, called the DQ scheme.

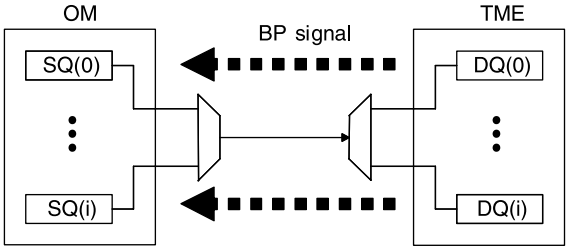


Figure 13.7 Back-pressure scheme between OM and TME.

13.3.1 Back-Pressure

Back-pressure (BP) is one of the simplest and commonly used flow control schemes. It uses a back-pressure signal to make the senders aware of the buffer occupancy. With reference to Figure 13.7, one can simply implement the back-pressure flow control scheme by letting the receiver periodically send a one-bit signal to all senders to indicate the buffer status. A "0" indicates the buffer has enough space so that the senders can send cells to it freely, and a "1" indicates the buffer has exceeded a threshold, thus all senders should stop sending cells to it.

The BP is a simple flow control scheme, but it lacks fairness. For instance, if the buffer is consumed by a congested port, all other traffic suffers because the BP signal only indicates whether the buffer exceeds the threshold or not. Although the senders have packets destined for the non-hot-spot ports, they cannot go through the buffer.

13.3.2 Credit-Based Flow Control

Since the BP flow control scheme is unfair to all flows, we next describe a credit-based link-by-link flow control scheme N23 [5], which processes on a per virtual connection (VC) basis to maintain fairness among all flows.

The credit plays a major role in the credit-based flow control scheme. Each sender can only forward data cells over the link, when it holds enough credits given by the receiver. In other words, if there is no credit for a sender, the flow is unable to forward cells to the link. On the other hand, the receiver periodically sends credits to the sender indicating availability of buffer space for receiving data cells of the VC. After having received additional credits, the sender is eligible to forward more data cells of the VC to the receiver according to the received credit information. Each time the sender forwards a data cell of a VC, it decrements its current credit balance for the VC by one.

The receiver is eligible to send a credit cell to the sender each time after it has forwarded N2 data cells since the previous credit cell for the same VC was sent. Upon receiving a credit cell with a credit value of C for a VC, the sender is permitted to forward up to C–E data cells of the VC before the next successfully transmitted credit cell for the VC is received, where E is the number of data cells the sender has forwarded over the VC for the past time period of RTT. The subtraction takes care of in-flight cells from the sender to the receiver, which the receiver had not seen when the credit cell was sent. In other words, the current credit balance is equal to C–E. As long as the credit balance is greater than zero, the flow is able to forward data. The N2 value can be a design or engineering choice. The larger N2 is, the less the bandwidth overhead is, and the more buffer space each VC uses.

13.3.3 The DQ Scheme

The N23 is a fair-flow control scheme that considers each flow individually [5]; however, in the multi-plane multi-stage switch architecture, such as the TrueWay switch, per VC flow control is not practical due to the excessive amount of credit counters. Therefore, by adopting and modifying the N23 scheme, a new per DQ-based flow control, called the DQ scheme, is proposed.

Similar to the credit-based flow control scheme described above, the DQ scheme also uses credits to maintain traffic control. In the DQ scheme, each sender maintains two counters. They are the queue outstanding cell count (QOC) and the RTT Outstanding Counter (ROC). The QOC represents the sum of the cells left in the DQ and the cells on the link destined for the DQ. If the distance between the two SMs is far, the number of cells on the link can be large. The QOC needs to be updated according to the information provided by the DQ. One simple way to update the QOC is to have the DQ periodically send a credit update (CRT). On the other hand, the ROC counts the number of cells transmitted during the RTT just before the CRT is received. In addition to these two counters, a fixed value called maximum reserved cells (MRC) is set to warn the sender about the buffer occupancy. The MRC is an engineering parameter that will affect the performance of the TrueWay switch. In the performance analysis section, we show the effect of various MRC values on the TrueWay switch performance.

From the sender point of view, the eligibility of sending cells depends on the value of the QOC. As long as the value of the QOC is less than MRC, the sender is able to forward cells to the DQ. Whenever a cell is transmitted, the QOC is incremented by one. The QOC is updated whenever a credit update is received.

The receiver on the other side periodically sends a CRT to the sender, which contains the number of cells in the DQ. Then the QOC can be updated as the sum of the CRT and ROC (i.e., QOC = CRT + ROC). The summation counts the cells in transition while the CRT is delivered. When the QOC is updated, the ROC is reset to 0.

In the DQ scheme, the QOC update depends on the CRT, thus the delivery of the CRT becomes a crucial step. Next, we examine two possible ways to deliver the CRT, depending on the data sent as the CRT update. One way is to determine how many cells were transmitted from the DQ during the last credit update period. This scheme is vulnerable to CRT loss and cell loss. Once a cell or CRT is lost, it is difficult to resynchronize the QOC and the actual outstanding number of cells for the DQ. The other way is to determine how many cells are left in the DQ. In this scheme, the system becomes much more tolerant of cell loss or credit loss. Since the CRTs are sent periodically, if one CRT is lost, the preceding CRT is able to recover the queuing status easily.

To avoid the deadlock situation, at least one cell space in the SM buffers should be reserved for all partial packets. Then any partial packet can forward a cell to the next stage and there will never be a deadlock situation. For this purpose, we introduce a queue reserved cell (QRC) counter. The QRC is set to the MRC as soon as its first cell is sent. In other words, QRC should be equal to or less than the MRC. For example, if MRC = 8, the QRC is set to eight as soon as the beginning of packet (BOP) cell is granted and the QRC is set to 0 as soon as the end of packet (EOP) cell is granted. The QRC is decremented by one whenever the SQ sends a cell to the DQ.

If the QOC of the DQ is larger than the MRC, the QRC is set to 0. Although the QRC becomes 0, the SQ can send more cells to the DQ if the buffer has a free space. If the QOC

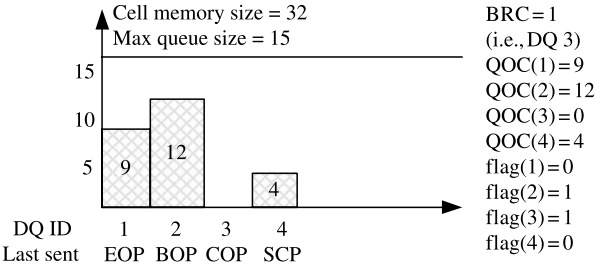
is less than the MRC and the DQ flag (DQF) is equal to 1, the sum of QOC and QRC should always be equal to MRC. The DQF bit indicates if the DQ is taken or not. If the DQ is not taken (i.e., DQF = 0), there is no partial packet destined for the DQ and any BOP cell or single cell packet (SCP) cell can be sent to the DQ. If the DQ is taken (i.e., DQF = 1), there is a SQ that has a partial packet destined for that DQ and only that SQ can send a cell to the DQ. If the DQF is set to 1, a BOP cell or SCP cell should not be sent. If a BOP or SCP cell is sent, more than one packet is interleaved and packet integrity is not maintained in the DQ. To avoid this, no more than one packet is allowed to be interleaved for the DQ.

The buffer reserved cell (BRC) counter is the sum of the QRCs of all DQs in the buffer. By adding the QOC and BRC, the memory space is reserved for the partial packet. When the partial packet arrives at the SQ, although the sum of the BOC and BRC is equal to the buffer size, the cell is eligible for transmission if the QRC is greater than 0.

Figure 13.8 shows an example of the DQ scheme. Let us assume that only four DQs are active at a certain time. A DQ is considered to be active if it has an outstanding cell (i.e., QOC is greater than 0) or if there is a partial packet destined for the DQ. The QOC can have a value between 0 and 15. The total cell memory size is 32.

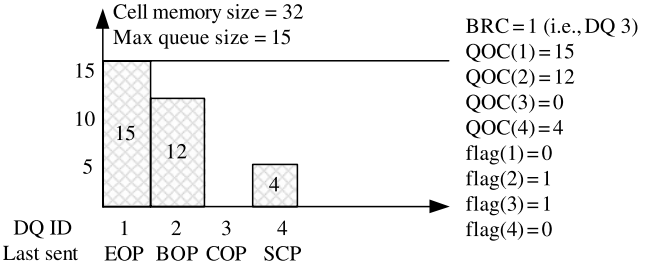
There are two partial packets (i.e., DQ(2) and DQ(3)). For DQ with a partial packet, only one SQ is eligible because the HOL cell of the SQ that sent a cell to the DQ should have a cell type of COP (Continue of Packet) or EOP and the DQF of the DQ should be set to 1. For the DQ without a partial packet, any BOP cell or SCP cell may be eligible. Since the sum of all QOCs and the BRC (i.e., 26) is less than the cell memory size (i.e., 32), there is room for a fresh packet. In the table part of Figure 13.8, "yes" means the cell is eligible and "no" means the cell is not eligible.

Figure 13.9 shows another example of the flow control mechanism. The DQ(3) has no outstanding cell but the last cell sent to the DQ was the BOP cell. Therefore, this DQ is considered to be active. Since the sum of QOCs and the BRC is equal to the cell memory size, there is no room for a fresh packet. The only eligible cell is one destined for DQ(3).



	1	2	3	4
BOP	Yes	No	No	Yes
COP	No	Yes	Yes	No
EOP	No	Yes	Yes	No
SCP	Yes	No	No	Yes

Figure 13.8 Example of DQ flow control.



	1	2	3	4
BOP	No	No	No	No
COP	No	No	Yes	No
EOP	No	No	Yes	No
SCP	No	No	No	No

Figure 13.9 Example of flow control.

13.4 PORT-TO-PORT FLOW CONTROL

When building a packet switch, it is a common practice to segment each arriving packet into multiple fixed-length cells (e.g., 64 bytes), route them through the switch fabric, and reassemble them back into packets with the reassembly queues at the output ports. Although packets belonging to the same flow must be delivered in order, packets passing through different paths may experience different queuing delays and packets can be delivered out-of-sequence through the switch fabric. For example, packet A is sent to the switch fabric from the TMI before packet B is sent. However, packet B may arrive at the TME before packet A arrives at the TME because packet A experienced more delay in the switch fabric than packet B.

In this section, we present four schemes to tackle the packet out-of-sequence issue in the multi-plane multi-stage buffered switch. The schemes can be categorized into two approaches, namely a hashing method and a buffer re-sequencing method. In the first approach (hashing), it forces the cells belonging to the same flow to take the same path through the switch fabric. As a result, all cells from a given flow will experience the same amount of queuing delay. Thus packets are delivered in order. Along this approach, we present two hashing schemes: static hashing and dynamic hashing. The second approach allows packet out-of-sequence within the switch fabric. However, TME uses a resequencing buffer to resequence the cells back into order before delivering them to the next link. We introduce two practical resequencing techniques. They are time-stamp-based resequencing and window-based resequencing.

13.4.1 Static Hashing

With reference to Figure 13.10, the static hashing scheme simply eliminates the packet out-of-sequence problem by hashing flows into a flow group and sending all packets of the same flow group to the same path. If cells belonging to the same flow traverse through the same path [6] (i.e., the same switch plane and the same CM) until they have all left the switch fabric, then all packets will be delivered in order.

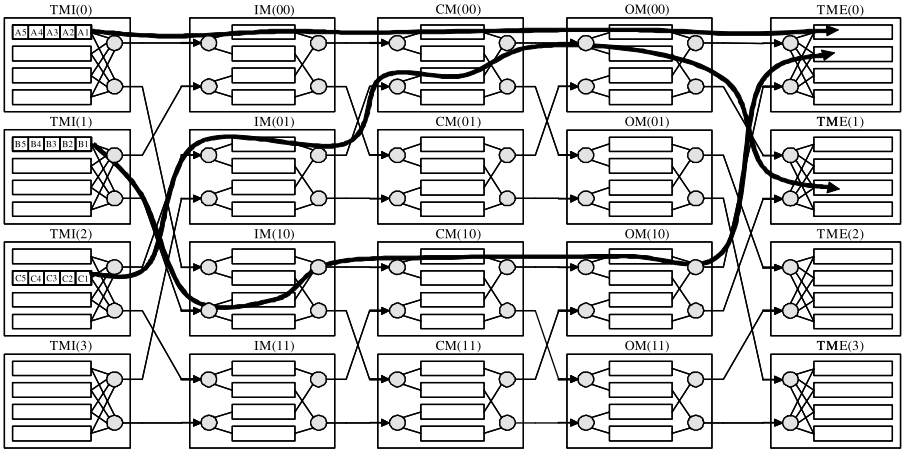


Figure 13.10 Static hashing.

As we know, in today's Internet, a core router may experience millions of packet flows; on the other hand, the TrueWay switch only has $p \times m$ paths between each input–output pair. Thus, it is impossible to assign a unique path to each individual packet flow traveling through the switch. As a result, the concept of flow group is introduced into the static hashing scheme. One simple way to hash flows into a flow group is to use CRC-16 (16-bit cyclic redundancy check). The static hashing scheme uses flow group ID (FGID) to distinguish each packet flow group. For instance, in the TrueWay switch with eight planes and 64 CMs, the total number of paths is 512, which is much smaller than the number of FGIDs (i.e., $216 = 65{,}536$ in the case of CRC-16). Therefore, the FGID must be mapped to the Path ID (PID).

One simple way to perform this mapping is to divide the FGID by the number of paths. The remainder is the PID. For example, if the FGID is 33,333 and the number of paths is 512, the PID becomes 53. Then all the packets with the FGID of 33,333 will be routed through the first plane (i.e., PLA = 0) and the 53rd CM (i.e., CMA = 53). This mapping is called static hashing. In summary, the static hashing allows TMIs to hash each incoming packet flow into flow groups and assign them into predetermined paths.

Load unbalancing is one problem of the static hashing scheme. This is mainly caused by different flow sizes (i.e., packet flows having different bandwidths). Even if the hashing achieves nearly even distribution of the packet flows to different routing paths, different flow sizes will cause the bandwidths on the routing paths to be uneven, and thus congestion on some of internal links.

13.4.2 Dynamic Hashing

To improve the load balancing problem, dynamic hashing is introduced where the input port maintains a count of outstanding packets in the switch fabric for each flow. If there is an outstanding packet in the switch fabric, all the following packets belonging to the flow must be sent to the same path with the previous packet. If not, they can be sent to any other path.

With reference to Figure 13.11, the TMI keeps track of the number of outstanding packets in the switch fabric using an outstanding packet counter (OPC). That is, the OPC indicates the outstanding number of packets in the switch fabric associated with the FGID

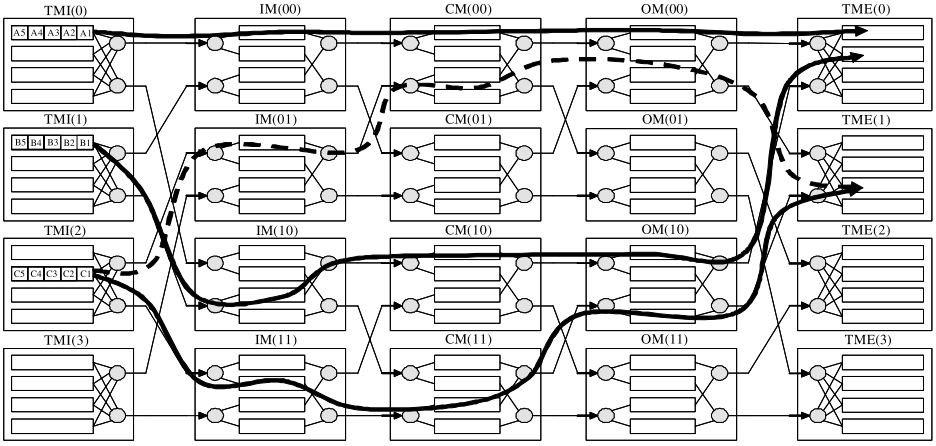


Figure 13.11 Dynamic hashing.

from the TMI. The OPC is incremented by one whenever a new packet with the FGID is sent to the switch fabric and is decremented by one whenever the TMI receives a packet acknowledgment (PACK) with the FGID from the TME. The TME sends a PACK when it receives the last cell of the packet (i.e., EOP or SCP cell).

The TMI maintains two tables: a distribution table and a status table. The distribution table contains the PID and the OPC for each FGID. The TMI determines the PID of each flow using its flow ID (FID). The PID is identified by a switch fabric plane number and a CM number in the plane.

The PID in the distribution table is valid while the OPC is greater than 0. If the OPC reaches 0, a new PID can be assigned to the FGID. If the OPC becomes 0, the FGID can be assigned to a different path from the previous path without having a packet out-of-sequence problem. This could be helpful for path congestion because if the OPC is equal to 0, the packets of the flow do not need to take the previous path. This scheme is called dynamic hashing. The simulation shows that dynamic hashing performs better than static hashing under traffic distributions with large flows.

In the case of multiple scheduling priorities, it will provide separate paths for each scheduling priority. Then since there is no overlap between different scheduling priorities, the operation of dynamic hashing for a single scheduling priority case can be applied to the multiple scheduling priorities.

The dynamic hashing works as follows. The TMI checks the FGID of the packet. If the OPC of the FGID is greater than 0, the packet is assigned to the path on the distribution table. If not, the TMI chooses one of the multiple paths according to the congestion status of each path, which is provided by the status table. The status table maintains two flags. One flag is used to indicate if the path is not accessable due to a link failure or a chip failure. If this flag is set, no packets must be sent to the path. The other flag is used to indicate if the path is congested or not. To set this flag, the input port needs feedback information from the switch fabric. The TMI will choose one path among the paths whose congestion flag is set to 0.

The status table is used to find an uncongested path. Path selection is performed in two phases. In the first phase, the status table chooses the plane in a round-robin fashion, which has at least one path whose congestion flag is set to 0. In the second phase, the status table

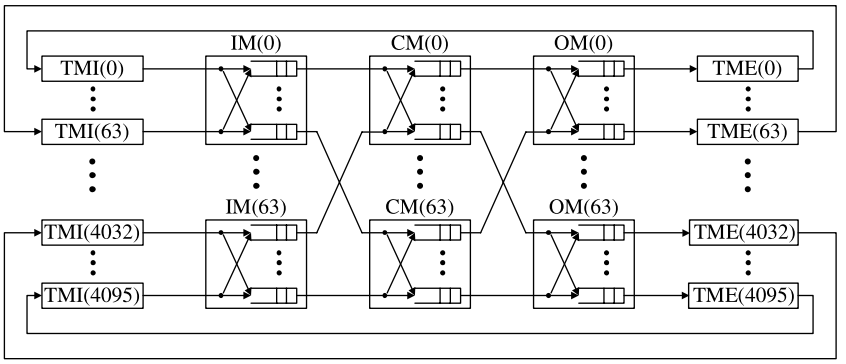


Figure 13.12 Packet acknowledgment in dynamic hashing.

chooses the CM, whose congestion flag is equal to 0. If the congestion flag is equal to 1, the path is considered congested. If it is equal to 0, the path is considered uncongested. If all congestion flags are equal to 0, the arbiter chooses one path in a round-robin manner among the CMs.

It is possible for packets of the same flow to take different paths as long as all the previous packets have left the switch fabric. This scheme achieves better load balancing than static hashing because the TMI assigns packets of the same flow to less congested paths. The static hashing scheme must assign packets to the pre-determined path even though the path is congested.

Next, we show how PACK can be efficiently delivered along with a data packet. The PACK can be delivered by the cell header. With reference to Figure 13.12, the PACK is sent to the neighboring TMI and the TMI sends the PACK to the switch fabric using the cell header. The PACK is sent through the same plane that the data cells go through. In the multi-plane multi-stage switch architecture, it is possible that multiple PACKs are generated at the same timeslot. Therefore, buffers for PACKs are inevitable in the multi-plane multi-stage switch. Although PACKs are delivered via the same switch fabric as the data packets, each IM, CM, and OM uses separate output FIFO queues to store PACKs in case of congestion. In other words, PACKs are queued separately from data packets at each switch module. Note that TMEs and TMIs do not require FIFOs to store the PACKs because they are able to process PACKs as soon as they arrive.

We use an example to show how PACKs are delivered in the multi-plane multi-stage switch. For instance, TMI(0) sends a packet to TME(4032). TMI(0) sends the last cell of the packet to TME(4032) and increments the OPC by one. TME(4032) receives the last cell of the packet, creates a PACK, and passes it to TMI(4032). Then TMI(4032) sends the PACK to IM(63). IM(63) chooses one of the CMs (e.g., CM(0)) for the PACK and stores it at the output buffer. Again, the output buffers at the IM, CM, and OM are FIFO. Therefore, if the buffer is not empty, it sends one PACK in each cell slot. CM(0) receives the PACK and stores it at the FIFO destined for OM(0) because the PACK is destined for TMI(0). Similarly, OM(0) receives the PACK and stores it at the FIFO destined for TME(0). TME(0) receives the PACK and passes it to TMI(0). Finally, TMI(0) receives the PACK and decrements the OPC by one.

Since the FIFOs at the IM, CM, and OM have a finite size, the PACK can be discarded due to the buffer overflow. To recover a lost PACK, the TMI sends a flush packet if the OPC is non-zero for a long time. If a flush packet is sent, the packets with the corresponding FID

are held at the TMI until the flush packet is acknowledged. If the flush PACK is received, the TMI resets the OPC and chooses a new path for the packet with the FGID. To distinguish the flushed PACK from other PACKs, one more bit in the cell header must be used.

In the worst case, each TME receives up to p PACKs and sends p PACKs, where p is the number of planes in the multi-plane multi-stage switch. The PACK will not be backlogged unless more than p PACKs are destined for the same TMI. Since each TMI can receive up to p PACKs per time slot, there can be contention in the switch fabric if more than p PACKs are destined for the same TMI. But since the PACKs are served in a cell-interleaved mode, it is unlikely that the PACKs are congested in the switch fabric.

13.4.3 Time-Stamp-Based Resequence

The first technique to resequencing packets is to use a time-based sequence number (SN). With reference to Figure 13.13, each packet carries a SN based on the arrival time at the input port, and is resequenced at the TME before its departure. The time-stamp-based resequence is a very simple resequencing technique to implement; however, it has limitations on a large scale switch. First, in a large scale switch, the SN is large and results in a large overhead ratio, which can be too big to be practical. A high overhead ratio implies an increase in implementation cost or performance degradation due to reduced internal speedup. Second, since Internet traffic is very complicated and each packet may travel freely along any path in the time-stamp-based scheme, it is difficult to estimate the out-of-sequence degree. Even in the case where the degree of out-of-sequence is bounded, implementing the resequencing circuits increases the cost. With these two disadvantages, implementation of time-stamp based resequencing in a high performance switch is commonly discouraged.

13.4.4 Window-Based Resequence

The next resequencing technique we propose is called window-based resequencing. Similar to time-stamp-based resequencing, window-based resequencing uses a SN for each flow. The SN ranges from 0 to W-1, where W is the window size. Therefore, the number of cells in the switch fabric for a flow cannot exceed W. If there are N input ports and N output

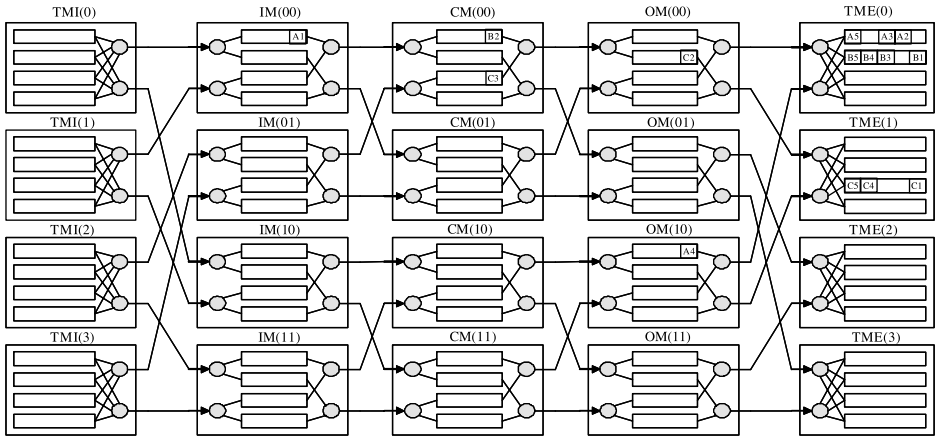


Figure 13.13 Time-stamp-based resequencing.

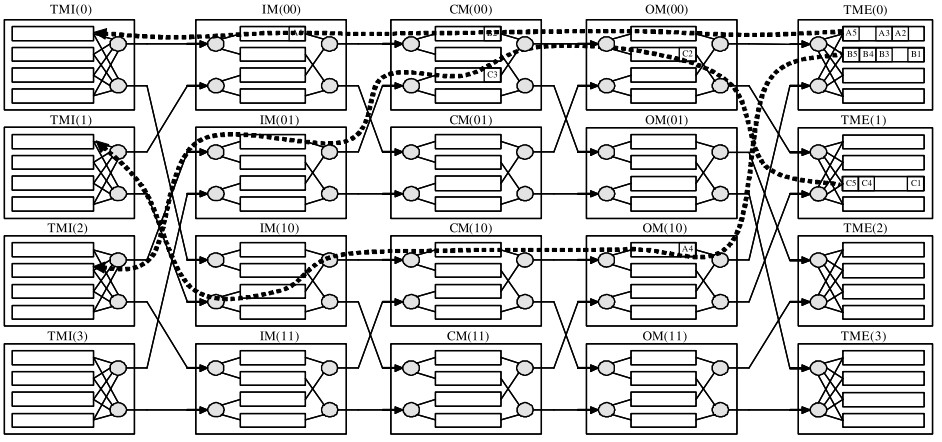


Figure 13.14 Window-based resequencing.

ports, the number of flows is N^2. Each input port has N VOQs; each output port has N virtual input queues (VIQs).

With reference to Figure 13.14, when an input port sends a cell to the switch fabric, it attaches an SN to the cell header. The SN is given by the VOQ manager. The VOQ manager maintains two pointers for each VOQ [the sequence head pointer (SEQ.head) and the sequence tail pointer (SEQ.tail)]. The SEQ.head is the SN to be attached to the cell sent from the input port. Right after a cell is sent, the SEQ.head is incremented by one.

When an output port receives a cell, it stores the cell in the VIQ according to its SN. The output port moves a cell from the VIQ to the reassembly queue (RAQ) only when the VIQ has a cell in order. Since the VIQ can receive cells out-of-order, it maintains a sequence pointer (SEQ.ptr). The SEQ.ptr indicates the SN for the VIQ to wait for. When a cell arrives at the TME with the SEQ.ptr, the cell is moved from the VIQ to the RAQ and the SEQ.ptr is incremented by one. If the SEQ.ptr is a multiple of P (i.e., the number of switch planes), the output port sends an acknowledgment (ACK) packet with the SN of SEQ.ptr.

When an input port receives an ACK packet, it updates its SEQ.tail pointer. If the SN in the ACK packet is in the eligible range, the SEQ.tail is updated to the SN in the ACK packet. If the SN in the ACK packet is not in the eligible range, the SEQ.tail is not updated. The delivery of ACK packets can be implemented as those in the dynamic hashing algorithm described in Section 13.4.2.

The window-based resequencing scheme uses a sliding window technique. With reference to Figure 13.15, the TMI only allows W unacknowledged packets for each flow in the switch fabric. Once TME acknowledges the number of successful packet transmissions, TMI releases the same amount of packets into the switch fabric. Thus, although packets may get out-of sequence within the switch fabric, the TME only needs W length of resequencing buffer to sort the packets back in order.

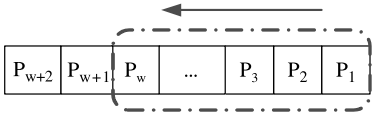


Figure 13.15 Sliding window technique in window-based resequencing.

Figure 13.16 summarizes the window-based resequencing algorithm. The three pointers (SEQ.head, SEQ.tail, and SEQ.ptr) are updated as already described. This ensures that the three pointers are always within the eligible range.

The question is how large the window size W must be in order to get 100 percent throughput. W must be equal to or greater than the round-trip time (RTT) in the unit of time slot, multiplied by the number of cells transmitted by the input port at each time slot, where the time slot is the time to transmit one cell. For example, if the RTT is 24 time slots and the number of cells transmitted by the input port in a time slot is eight, W must be at least 192 cells long.

The RTT is composed of the propagation time between the input port and the output port, and the queuing time at the switch fabric. The propagation time is deterministic and fixed because it is proportional to the distance between the input port and the output port. If the switch fabric has a multi-stage architecture, the propagation time between the input port and the output port is the sum of the propagation times between the input port and IM, IM and CM, CM and OM, and OM and the output port.

For example, if the link speed is 2.5 Gbps and the cell size is 64 bytes (i.e., 512 bits), the time slot is 204.8 nsec. If the distance between IM and CM is 100 m, the propagation delay is 500 nsec (i.e., $100\,\text{m} \div 2 \times 10^8\,\text{m/s}$), which is equivalent to three time slots. Let us assume that the RTT between IM and CM is eight time slots and that between the input port and IM is four time slots. Then the propagation time between the input port and the output port is 24 time slots.

If the switch fabric is not congested, the queuing time at the switch fabric is negligible and the RTT is close to the propagation time. If the queuing time is less than eight time slots and the propagation time is 24 time slots, the RTT is less than 32 time slots and it is enough to set the window size W to 256 cells. Note that the queuing time is the sum of the queuing delay of the forward direction from the input port to the output port for the delivery of the data cells, and that of the reverse direction from the output port to the input port for the delivery of the ACK packets.

If all input ports dispatch cells as evenly as possible to all possible paths, the queuing delay at the switch fabric will be minimized. One way to dispatch cells as evenly as possible

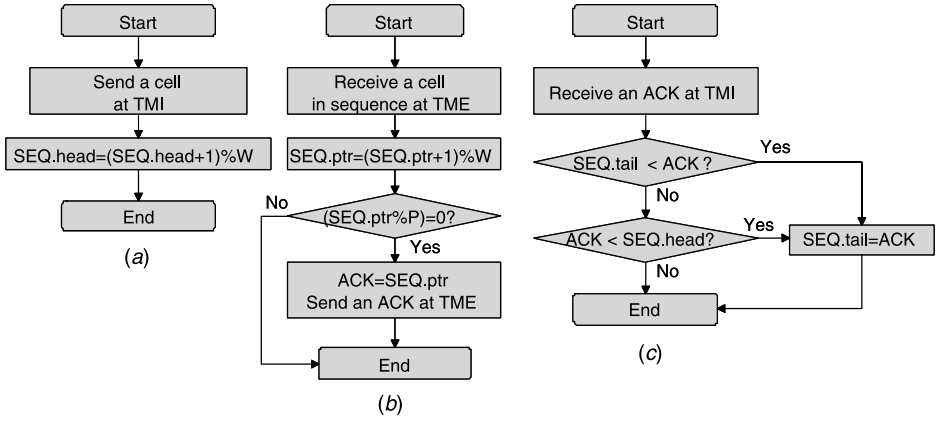


Figure 13.16 Window-based resequencing flow diagram (*a*) Cell transmission at TMI; (*b*) Cell reception and ACK transmission at TME; (*c*) ACK reception at TMI.

to all possible paths is RR dispatching. That is, each input port maintains a path round-robin (RR) pointer per output port. If there is a valid cell at the head of the VOQ, the HOL cell of the VOQ is sent to the path in the path RR pointer and the path RR pointer is incremented by one.

13.5 PERFORMANCE ANALYSIS

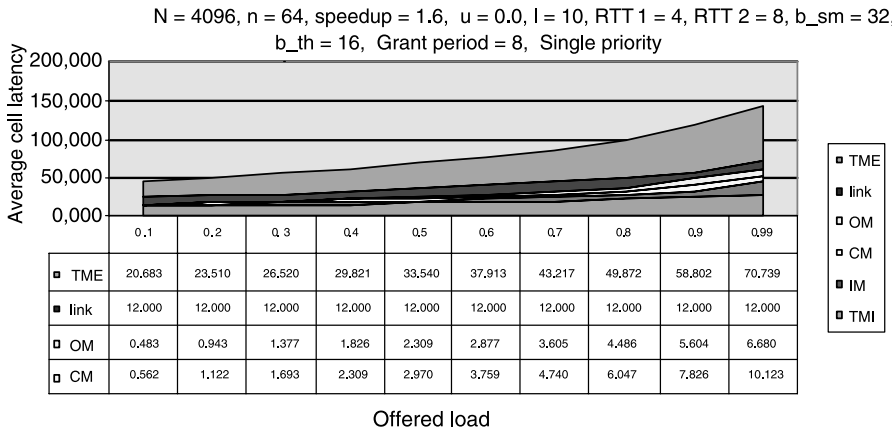
This section provides performance results of the TrueWay switch architecture. The system parameters used in computer simulations evaluating the performance are as follows. There are eight planes and the IM, CM, and OM have 16 input links and 16 output links. There are 256 TMs. Each TM receives up to four cells per time slot and sends up to eight cells per time slot satisfying an internal speedup of 1.6.

The RTT between the TM and the IM/OM is assumed to be four time slots. The RTT between the IM/OM and CM is assumed to be eight time slots. The cell acknowledgment period is four time slots. The number of flows per TMI is 100,000. The VOQ size is 1024 cells and the cell memory size at the TMI is 262,144 cells. The RAQ size is 256 cells and the cell memory size at the TME is 4096 cells. The DQ size is 15 cells and the cell memory size at the SM is 32 cells.

It is assumed that all TMIs have the same traffic model. The simulation assumes a single priority level and a unicast destination. Each simulation generated about 20 million cells.

13.5.1 Random Uniform Traffic

Random uniform traffic is characterized as the traffic with the same length packets (e.g., one cell) with uniformly distributed destinations and similar flow bandwidth. The throughput performance of the proposed switch under random uniform traffic is always the same as the offered load, which is the expected result. The delay performance of the proposed switch under random uniform traffic is slightly larger than that of the output-buffered switch, as shown in Figure 13.17. This is because of the arbitration time and the distance between modules.



	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.99
TME	20.683	23.510	26.520	29.821	33.540	37.913	43.217	49.872	58.802	70.739
link	12.000	12.000	12.000	12.000	12.000	12.000	12.000	12.000	12.000	12.000
OM	0.483	0.943	1.377	1.826	2.309	2.877	3.605	4.486	5.604	6.680
CM	0.562	1.122	1.693	2.309	2.970	3.759	4.740	6.047	7.826	10.123

Figure 13.17 Delay performance under random uniform traffic.

13.5.2 Hot-Spot Traffic

The destinations of packets coming to the switch are independent of each other. Therefore, it is possible that a large portion of the traffic is destined for the same TME at the same time. The destination of a packet is determined by the non-uniform parameter u. If $u = 0.0$, the destination is uniform over all the TMEs. If $u = 1.0$, the destination is fixed to one hot-spot TME. In between these extremes, $u \times 100$ percent of the traffic has a fixed destination to one hot-spot TME and the other $(1 - u) \times 100$ percent of the traffic is uniformly distributed over all the TMEs. Figure 13.18 shows the maximum throughput versus the non-uniform parameter u for various scheduling schemes.

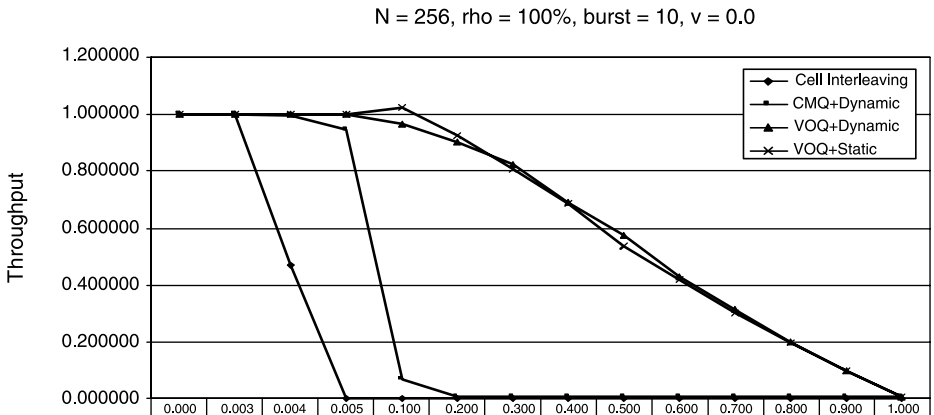
13.5.3 Bursty Traffic

For this simulation, the average packet size is assumed to be 10 cells, with a maximum packet size of 192 cells. If the average packet size is smaller, the performance would be better. The average packet size in Internet traffic is about 280 bytes (i.e., five cells) and the maximum packet size is 9000 bytes (i.e., 161 cells).

Figure 13.19 shows the throughput performance of the PPI scheme and DPI scheme under bursty traffic. DPI1 is the DPI scheme with an MRC of 1 cell. DPI8 has an MRC of eight cells. It is observed that DPI8 performs best among the four schemes. Thus, our choice for the scheduling scheme is DPI8. It is also possible to let the MRC value be set externally.

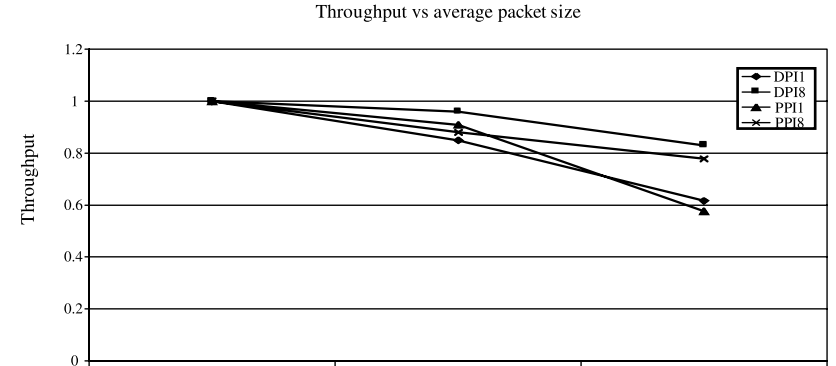
13.5.4 Hashing Schemes

This section describes the performance of the TrueWay switch with two different hashing schemes. To analyze the performance of static hashing and dynamic hashing, a variety of traffic loading techniques are used to evaluate the impact on both hashing algorithms.



	0.000	0.003	0.004	0.005	0.100	0.200	0.300	0.400	0.500	0.600	0.700	0.800	0.900	1.000
Cell Interleaving	0.99996	0.99996	0.47078	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
CMQ+Dynamic	1.00017	1.00017	0.99880	0.94553	0.06767	0.00779	0.00779	0.00779	0.00779	0.00779	0.00779	0.00779	0.00779	0.00779
VOQ+Dynamic	1.00013	1.00013	1.00013	1.00013	0.96793	0.90294	0.82363	0.69075	0.57626	0.43028	0.31600	0.20058	0.09790	0.00779
VOQ+Static	1.00039	1.00039	1.00039	1.00039	1.02291	0.92576	0.80667	0.68665	0.53625	0.42035	0.30045	0.19506	0.09673	0.00780

Figure 13.18 Throughput performance under hot-spot traffic.

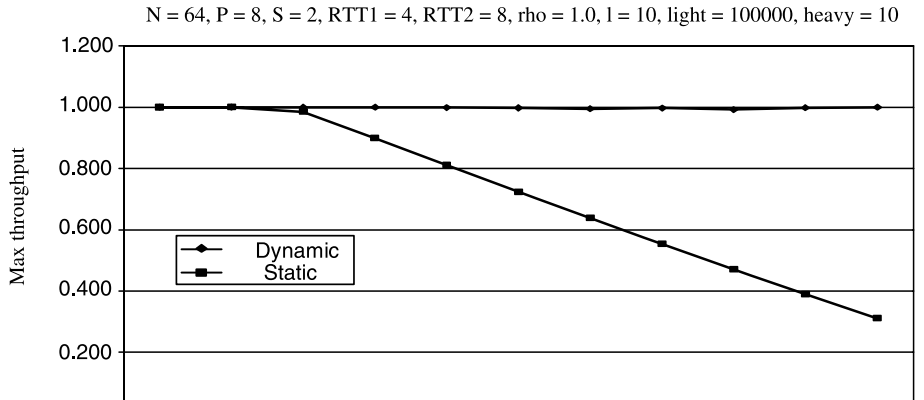


	1	10	100
DPI1	0.999997	0.84996	0.616659
DPI8	0.999997	0.95979	0.830788
PPI1	0.999997	0.90896	.576497
PPI8	0.999997	0.880875	0.778683

Average packet size

Figure 13.19 Performance under bursty traffic.

It is true that the flow bandwidth of Internet traffic has large variations. Some flows are a few kbps while some flows are a few Gbps. Assume that the port speed is 10 Gbps, the number of light flows is 100,000, and the number of heavy flows is 10. Let v be the percentage of the heavy traffic. If $v = 0.0$, then all flows have the same bandwidth and the flow bandwidth is 100 kbps. If $v = 0.5$, the heavy flows have a bandwidth of 500 Mbps while the light flows have a bandwidth of 50 kbps. If $v = 1.0$, all flows will have the same bandwidth, 1 Gbps. Figure 13.20 shows the impact of flow bandwidth variation on the system throughput for static hashing and dynamic hashing. We can see that the dynamic



	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Dynamic	1.000	1.000	1.000	1.000	0.999	0.998	0.995	0.997	0.993	0.999	1.000
Static	1.000	1.000	0.985	0.899	0.811	0.724	0.638	0.553	0.471	0.390	0.311

Percentage of heavy traffic

Figure 13.20 Throughput performance of static hashing and dynamic hashing.

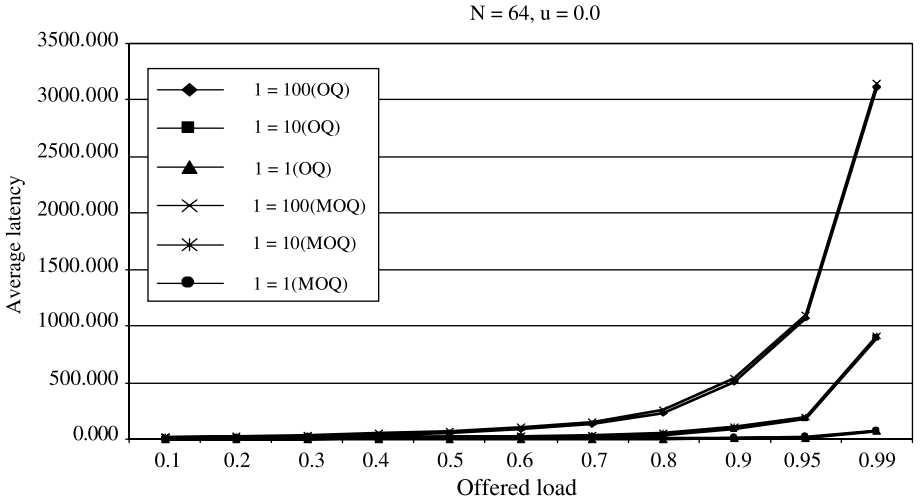


Figure 13.21 Delay performance under bursty traffic.

hashing scheme is immune to any bandwidth variation, while the static hashing could not maintain 100 percent throughput under various traffic loading conditions.

13.5.5 Window-Based Resequencing Scheme

We set the window size W to 256, which is analyzed in the earlier section that evaluates the TrueWay switch performance with the window-based resequencing scheme.

Figure 13.21 shows the delay performance of the proposed scheme under bursty traffic, compared with the optimal output-queued switch (OQ). The packet size is assumed to be geometrically distributed with the average packet size of l. Three sets of traffic patterns are shown ($l = 1$, $l = 10$, and $l = 100$). As in Figure 13.21, the delay in the switch fabric is very small. The worst-case average queuing delay observed in the switch fabric, VOQ, and VIQ was 21 time slots (i.e., approximately 4.3 μsec if one time slot is 204.8 nsec) when $l = 100$ and the offered load is 99 percent. This verifies that the queuing delay can be neglected in the window-based resequencing scheme. Thus the analysis about the window size in the earlier section is valid.

13.6 PROTOTYPE

A small-scale TrueWay switch on a chassis is prototyped, as shown in Figure 13.22, to evaluate various packet scheduling schemes, link-to-link and port-to-port flow control schemes. The prototype consists of a high-speed multi-trace backplane and up to 16 plug-in cards. The backplane and plug-in cards were manufactured using a state-of-the-art 14-layer printed circuit board (PCB) process. The backplane provides a total bandwidth of 640 Gbps for the inter-connection of the plug-in cards, where each card has 16 2.5-Gbps duplex ports connecting to other cards. Each of the plug-in cards can have up to two units, where each unit consists of one Xilinx Virtex-II 3000 FPGA, one Velio Octal 3.125 Gbps SerDes, and the necessary logic circuits.

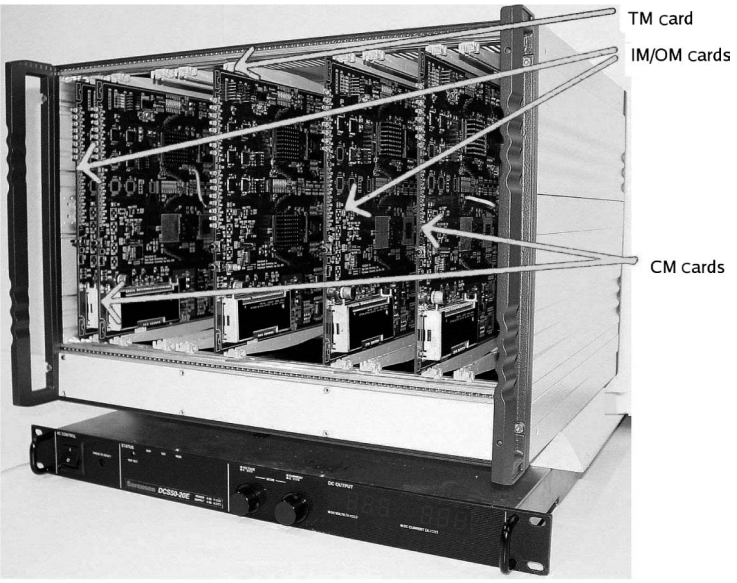


Figure 13.22 TrueWay testbed.

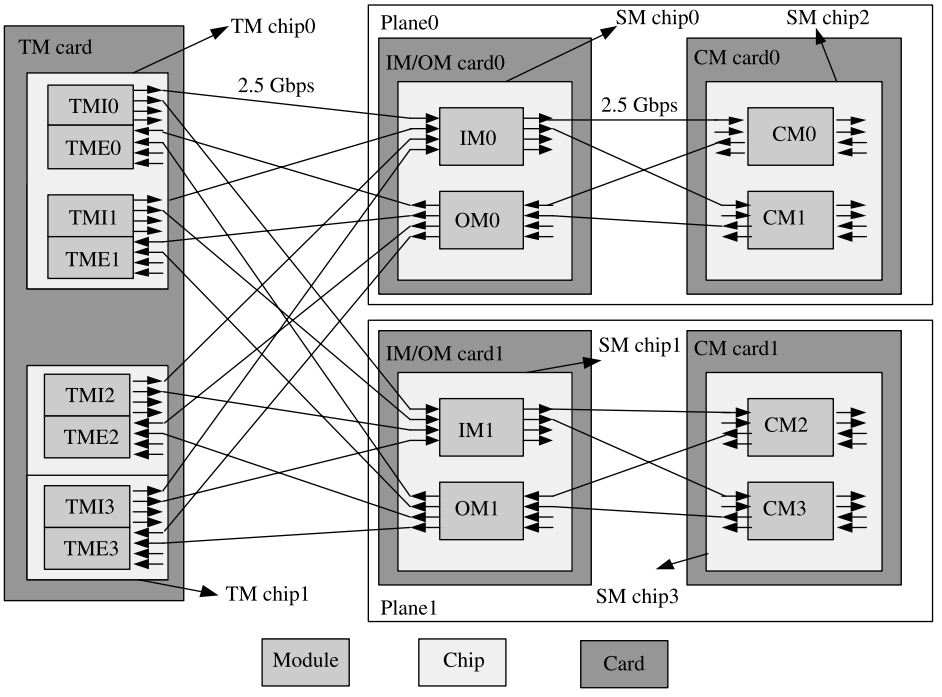


Figure 13.23 Logical view of the prototype.

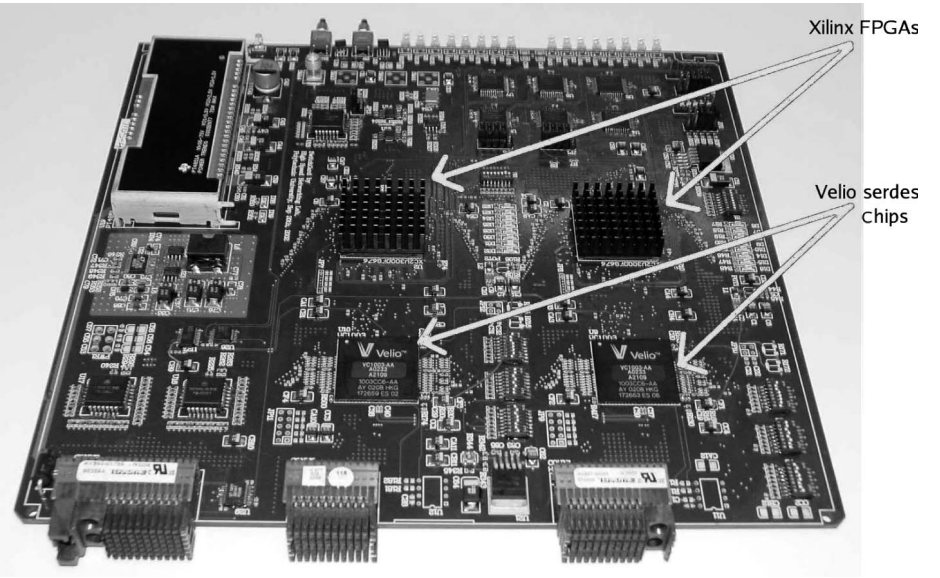


Figure 13.24 Testbed card with two FPGAs (under heat sink) and two Velio Serdes chips.

The chassis provides a generic framework for testing different packet scheduling schemes under the Clos-network switch structure by programming FPGA chips. The hardware platform can be configured, for instance, to a 16×16 switch with four switch planes and a total capacity of 40 Gbps.

The current prototype only consists of a 4×4 switch with two switch planes for demonstration purposes, as shown in Figure 13.23. This switch consists of five cards and six FPGAs in total. The traffic manager (TM) card has two FPGAs, where each FPGA can accommodate two TMI/TME unit pairs (i.e., two TM chips), making a total of four TMs connecting to the four ports of the switch. The switch fabric includes four cards, each with one FPGA. One card/chip can accommodate two SMs. Each IM/OM card has one IM/OM pair (i.e., one SM chip) and each CM card has two CMs (i.e., one SM chip). One IM/OM card/chip and one CM card/chip form a single switch plane. Figure 13.24 shows a photo of this implementation.

Some testing features are also implemented into the TM chips. Each TM chip has a built-in traffic generator that can generate different types of packet streams to be applied to the switch. At the same time, TM chips also check if any packet is lost in the switch fabric. The Xilinx ISE Foundation toolkit with VHDL (very high-speed integrated circuit hardware description language) was used for the entire FPGA implementation. A 2-unit card is shown in Figure 13.24.

REFERENCES

[1] H. J. Chao, "Next generation routers," *IEEE Proceedings*, vol. 90, no. 9, pp. 1518–1558 (Sept. 2002).

[2] C. Clos, "A study of non-blocking switching networks," *Bell System Technical Journal*, vol. 32, no. 3, pp. 406–424 (Mar. 1953).

[3] R. Sudan and W. Mukai, *Introduction to the Cisco CRS-1 Carrier Routing System*, Cisco Systems, Inc., San Jose, California (Jan. 1994).

[4] H. J. Chao, J. S. Park, S. Artan, S. Jiang, and G. Zhang, "TrueWay: A highly scalable multi-plane multi-stage buffered packet switch," in *Proc. IEEE Workshop on High Performance Switching and Routing*, Hong Kong, pp. 246–253 (May 2005).

[5] H. T. Kung, T. Blackwell, and A. Chapman, "Credit-based flow control for ATM networks: Credit update protocol, adaptive credit allocation, and statistical multiplexing," in *Proc. ACM SIGCOMM*, London, United Kingdom, pp. 101–115 (Aug. 1994).

[6] Y. C. Jung, C. K. Un, S. M. Ryu, and S. C. Lee, "Analysis of out-of-sequence problem and preventative schemes in parallel switch architecture for high-speed ATM network," *Proceedings of the IEEE*, vol. 141, no. 1, pp. 29–38 (Feb. 1994).