CHAPTER 3
Switching and Forwarding

H. Peyravi
Department of Computer Science
Kent State University
Kent, Ohio 44242
peyravi@mcs.kent.edu
http://mars.mcs.kent.edu/~peyravi
Fall 2001
### Switching and Forwarding

**Scalable Networks**

- **Switch**: Forwards packets from input port to output port; port selected based on destination address in packet header.

<table>
<thead>
<tr>
<th>Input Ports</th>
<th>Output Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>T3</td>
<td>T3</td>
</tr>
<tr>
<td>T3</td>
<td>T3</td>
</tr>
<tr>
<td>STS-1</td>
<td>STS-1</td>
</tr>
</tbody>
</table>

- Can build networks that cover large geographic area
- Can build networks that support large numbers of hosts
- Can add new hosts without affecting performance of existing hosts

### Routing Techniques Elements

<table>
<thead>
<tr>
<th>Performance criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of hops</td>
</tr>
<tr>
<td>Distance</td>
</tr>
<tr>
<td>Speed</td>
</tr>
<tr>
<td>Delay</td>
</tr>
<tr>
<td>Cost</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decision time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet</td>
</tr>
<tr>
<td>Session</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decision place</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributed</td>
</tr>
<tr>
<td>Centralized</td>
</tr>
<tr>
<td>Source</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Information sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
</tr>
<tr>
<td>Local</td>
</tr>
<tr>
<td>Adjacent nodes</td>
</tr>
<tr>
<td>Nodes along route</td>
</tr>
<tr>
<td>All nodes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Routing strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed</td>
</tr>
<tr>
<td>Adaptive</td>
</tr>
<tr>
<td>Random</td>
</tr>
<tr>
<td>Flooding</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Adaptive routing update time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous</td>
</tr>
<tr>
<td>Periodic</td>
</tr>
<tr>
<td>When topology changes</td>
</tr>
<tr>
<td>Major load changes</td>
</tr>
</tbody>
</table>
**Source Routing**

- Address contains a sequence of ports on path from source to destination.

**Virtual Circuit Switching**

- Explicit connection setup (and tear-down) phase
- Subsequent packets follow same circuit
- Analogy: phone call
- Sometimes called *connection-oriented* model
- Each switch maintains a VC table.
Datagrams

- No connection setup phase
- Each packet forwarded independently
- Analogy: postal system
- Sometimes called *connectionless* model
- Each switch maintains a forwarding (routing) table

Virtual Circuit vs. Datagram

**Virtual Circuit Model:**
- Typically wait full RTT for connection setup before sending first data packet.
- While the connection request contains the full address for destination, each data packet contains only a small identifier, making the per-packet header overhead small.
- If a switch or a link in a connection fails, the connection is broken and a new one needs to be established.
- Connection setup provides an opportunity to reserve resources.

**Datagram Model:**
- There is no round trip time delay waiting for connection setup; a host can send data as soon as it is ready.
- Source host has no way of knowing if the network is capable of delivering a packet or if the destination host is even up.
- Since packets are treated independently, it is possible to route around link and node failures.
- Since every packet must carry the full address of the destination, the overhead per packet is higher than for the connection-oriented model.
### Switching and Forwarding

**Performance**

- Switches can be built from a general-purpose workstations; will consider special-purpose hardware later.

  ![Switch Diagram](image)

  - **Aggregate bandwidth**
    - $1/2$ of the I/O bus bandwidth
    - Capacity is shared among all hosts connected to switch
    - Example: 800Mbps bus can support 8 T3 ports

  - **Packets-per-second**
    - Must be able to switch small packets
    - 100,000 packets-per-second is an achievable number
    - Example: 64-byte packets implies 51.2Mbps

---

### Routing

- **Forwarding versus Routing**
  - Forwarding: selects an output port based on destination address and routing table
  - Routing: process by which routing table is built

- **Network as a Graph**

  ![Graph Diagram](image)

- **Problem:** Find the lowest cost path between any two nodes

- **Factors:**
  - Static: topology
  - Dynamic: load
Interior Gateway Protocols (IGP)

- used for Intra-domain routing e.g. between routers within Kent campus

Two major approaches

- **Distance Vector**: Each router sends a vector of distances to its neighbors. The vector contains distances to all nodes in the network
  - Example: RIP (Routing Information Protocol)
- **Link State**: Each router sends a vector of distances to all nodes. The vector contains only distances to neighbors
  - newer method used in Internet
  - Example: OSPF (Open Shortest Path First)

We will discuss RIP and OSPF later in Chapter 4 together with inter-domain routing using BGP

---

**Distance Vector**

- Each node maintains a set of triples:
  - \((\text{Destination}, \text{Cost}, \text{NextHop})\)
- Each node sends updates to (and receives updates from) its directly connected neighbors
  - periodically (on the order of several seconds)
  - whenever its table changes (called triggered update)
- Each update is a list of pairs:
  - \((\text{Destination}, \text{Cost})\)
- Update local table if receive a “better” route
  - smaller cost
  - came from next-hop
- Refresh existing routes; delete if they time out
- How do you tell a node is down?
  - send test packet e.g. ping
  - don’t see periodic update
### Example 1
- F detects that link to G has failed
- F sets distance to G to infinity and sends update to A
- A sets distance to G to infinity since it uses F to reach G
- A receives periodic update from C with 2-hop path to G
- A sets distance to G to 3 and sends update to F
- F decides it can reach G in 4 hops via A

### Example 2
- Link from A to E fails
- A advertises distance of infinity to E
- B and C advertise a distance of 2 to E
- B decides it can reach E in 3 hops; advertises this to A
- A decides it can reach E in 4 hops; advertises this to C
- C decides that it can reach E in 5 hops......

### Heuristics to break routing loops
- set infinity to 16
- split horizon - don’t send back to origin i.e. don’t send (E,2)
- split horizon with poison reverse - send (E,∞)
- only works for 2 node loops
**Link State**

- **Strategy:** Send to all nodes (not just neighbors) information about directly connected links (not entire routing table).

- **Link State Packet (LSP)**
  - id of the node that created the LSP
  - cost of link to each directly connected neighbor
  - sequence number (SEQNO)
  - time-to-live (TTL) for this packet

- **Reliable Flooding**
  - store most recent LSP from each node
  - forward LSP to all nodes but one that sent it
  - generate new LSP periodically (hours) or on topology change; increment SEQNO
  - start SEQNO at 0 when reboot
  - decrement TTL of each stored LSP before flooding and also by “ageing”; reflood and discard when TTL=0

---

**Route Calculation (in theory)**

- Dijkstra’s shortest path algorithm
- $N$ denotes set of nodes in the graph
- $l(i,j)$ denotes non-negative cost (weight) for edge $(i,j)$
- $s \in N$ denotes this node
- $M$ denotes the set of nodes incorporated so far
- $C(n)$ denotes cost of the path from $s$ to node $n$

\[
M = \{s\}
\]

for each $n$ in $N - \{s\}$

\[
C(n) = l(s, n)
\]

while ($N \neq M$)

\[
M = M \cup \{w\} \text{ such that } C(w)
\]

is the minimum for all $w \in (N - M)$

for each $n$ in $(N - M)$

\[
C(n) = \text{MIN}(C(n), C(w) + l(w, n))
\]
**Route Calculation (in practice)**

- Forward search algorithm
- Each switch maintains two lists: Tentative and Confirmed
- Each list contains a set of triples:
  \[(\text{Destination, Cost, NextHop})\]

1. Initialize Confirmed with entry for me; cost = 0.
2. For the node just added to Confirmed (call it Next) select its LSP.
3. For each Neighbor of Next, calculate the Cost to reach this Neighbor as the sum of the cost from me to Next and from Next to Neighbor.
4. If Tentative is empty, stop. Otherwise, pick entry from Tentative with the lowest cost, move it to Confirmed, and return to step 2.

---

3.1. If Neighbor is currently in neither Confirmed or Tentative, add (Neighbor, Cost, NextHop) to Tentative, where NextHop is the direction to reach Next.

3.2. If Neighbor is currently in Tentative and Cost is less than current cost for Neighbor, then replace current entry with (Neighbor, Cost, NextHop), where NextHop is the direction to reach Next.
### Metrics

- **Original ARPANET metric**
  - measured number of packets enqueued on each link
  - took neither latency or bandwidth into consideration

- **New ARPANET metric**
  - stamp each incoming packet with its arrival time ($AT$)
  - record departure time ($DT$)
  - when link-level ACK arrives, compute
    \[ \text{Delay} = (DT - AT) + \text{Transmit + Latency} \]
  - if timeout, reset $DT$ to departure time for retransmission
  - link cost = average delay over some time period

- **Problems with “New” metric**
  - under low load, static factors dominated cost; worked OK
  - under high load, congested links had very high costs; packets oscillated between congested and idle links
  - range of costs too large; preferred path of 126 lightly loaded 56Kbps links to a 1-hop 9.6Kbps path

- **Revised ARPANET metric**
  - replaced delay measurement with link utilization
  - average to suppress sudden changes
    - compressed dynamic range (see Fig 4.21)

### Switching and Forwarding Routing

<table>
<thead>
<tr>
<th>Step</th>
<th>Confirmed</th>
<th>Tentative</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>(D,0,-)</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>(D,0,-)</td>
<td>(B,11,B)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(C,2,C)</td>
</tr>
<tr>
<td>3.</td>
<td>(D,0,-)</td>
<td>(B,11,B)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(C,2,C)</td>
</tr>
<tr>
<td>4.</td>
<td>(D,0,-)</td>
<td>(B,5,C)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(A,12,C)</td>
</tr>
<tr>
<td>5.</td>
<td>(D,0,-)</td>
<td>(A,12,C)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(C,2,C)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(B,5,C)</td>
</tr>
<tr>
<td>6.</td>
<td>(D,0,-)</td>
<td>(A,10,C)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(C,2,C)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(B,5,C)</td>
</tr>
<tr>
<td>7.</td>
<td>(D,0,-)</td>
<td>(C,2,C)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(B,5,C)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(A,10,C)</td>
</tr>
</tbody>
</table>
highly loaded link never has a cost more than 3 times its idle cost
most expensive link only 7 times the cost of the least expensive
high-speed satellite link more attractive than low-speed terrestrial link
cost is a function of link utilization only at moderate to high loads.
changes not instantaneous - only notify changes when exceed threshold

Bridges and LAN switches

- **Simple Bridge**: a node which accepts frames from an Ethernet (port) and forwards them on all other Ethernet (ports)
- Improvement would be to forward frames only to port with the destination host
- How does bridge get this information?
  - Human download
  - **Learning Bridge**: Bridge inspects source address in all frames - knows which port host is on
  - Create table - timeout entries in case host moved
Spanning Tree Algorithm

- Problem: If use multiple learning bridges can get loops
- How get loop? Human error or for redundancy
- Solution: Distributed Spanning Tree Algorithm
- Can represent extended LAN as Graph.
- Spanning Tree is subgraph that covers all the vertices but with no cycles
- Theory:
  - pick bridge with lowest id as root - it forwards all frames
  - Each bridge computes shortest path to root - and uses port involved to forward towards root
  - Each LAN picks bridge closest to root

Practical Algorithm

- Exchange configuration messages containing
  - the id for the sending bridge
  - the id of what it believes is the root
  - distance in hops to root
- Bridge records the current best message on each port, adding one to distance to root
- Best if
  - root id is smaller or
  - root id equal but distance shorter or
  - both equal but sending bridge has lower id
- Bridge forwards rather than generates messages when realizes not root
- Bridge stops forwarding messages on port when it’s not designated bridge for LAN
- Can extend spanning tree to prune multicasts - not widely done.
Limitations of Bridges

- Spanning Tree scales linearly
- Can only connect networks with same frame headers
- Congestion and dropped frames possible at bridges
- Latency and variability may increase
- Want to reduce broadcast traffic
  - VLAN - virtual LAN: partition extended LAN into multiple VLAN
  - Broadcast packets are only sent on ports that are in same VLAN
  - Adds VLAN header after Ethernet header to do this

Asynchronous Transfer Mode (ATM)

Overview

- Connection-oriented packet-switched network
- Used in both WAN and LAN settings
- Signalling (connection setup) Protocol: Q.2931
- Specified by ATM Forum
- Packets are called cells: 5-byte header + 48-byte payload
- Commonly transmitted over SONET (but not necessarily)

Cells

- Variable versus Fixed-Length
  - no optimal fixed-length
    - if small: high header-to-data overhead
    - if large: low utilization for small messages
  - fixed-length are easier to switch in hardware
    - simpler
    - enables parallelism
- Small size improves queue behavior
  - finer-grained pre-emption point for scheduling link
    - maximum packet = 4KB
    - link speed = 100Mbps
    - transmission time = \( \frac{4096 \times 8}{100} = 327.68\mu s \)
    - high priority packet may sit in the queue 327.68\(\mu s\)
    - in contrast, \( \frac{53 \times 8}{100} = 4.24\mu s \) for ATM
  - near cut-through behavior
    - two 4KB packets arrive at same time
    - link idle for 327.68\(\mu s\) while both arrive
    - at end of 327.68\(\mu s\), still have 8KB to transmit
    - in contrast, can transmit first cell after 4.24\(\mu s\)
    - at end of 327.68\(\mu s\), just over 4KB left in queue

- Carrying Voice in Cells
  - voice digitally encoded at 64Kbps (8-bit samples at 8KHz)
  - need full cell’s worth of samples before sending cell
  - example: 1000-byte cells implies 125ms per cell (too long)
  - smaller latency implies no need for echo cancellors

- Settled on compromise of 48 bytes: \( \frac{32+64}{2} \)

---

Cell Format

- User-Network Interface (UNI)
  - host-to-switch format
  - GFC: Generic Flow Control (still being defined)
  - VCI: Virtual Circuit Identifier
  - VPI: Virtual Path Identifier
  - Type: management, congestion control, AAL5 (later)
  - CLP: Cell Loss Priority
  - HEC: Header Error Check (CRC-8)

- Network-Network Interface (NNI)
  - switch-to-switch format
  - GFC becomes part of VPI field
## ATM Adaptation Layer (AAL)

- **AAL 1 and 2** designed for applications that need guaranteed rate (e.g., voice, video)
- **AAL 3/4** designed for packet data
- **AAL 5** is an alternative standard for packet data

### AAL 3/4

- **Convergence Sublayer Protocol Data Unit (CS-PDU)**
  - CPI: common part indicator (version field)
  - Btag/Etag: beginning and ending tag
  - BAsize: hint on amount of buffer space to allocate
  - Length: size of whole PDU

### Cell Format

<table>
<thead>
<tr>
<th>Payload</th>
<th>ATM header</th>
<th>Type</th>
<th>SEQ</th>
<th>MID</th>
<th>Payload</th>
<th>Length</th>
<th>CRC-10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>40</td>
<td>2</td>
<td>4</td>
<td>10</td>
<td>32</td>
<td>10</td>
</tr>
</tbody>
</table>

- **Type**
  - BOM: beginning of message
  - COM: continuation of message
  - EOM: end of message
- **SEQ**: sequence number
- **MID**: message id
- **Length**: number of bytes of PDU in this cell
**AAL5**

- **CS-PDU Format**
  - Pad Len CRC-32
  - Length: size of PDU (data only)
  - CRC-32 (detects missing or misordered cells)

- **Cell Format**
  - End-of-PDU bit in Type field of ATM header

- **VPI/VCI**
  - Host: treat as 24-bit circuit identifier
    - If cheap: one-per application; use for demultiplexing
    - If expensive: multiplex several applications onto one VCI
  - Network: aggregate multiple circuits into one path

---

**ATM in the LAN**

- Originally WAN technology
- Adopted for LANs because
  - It was switched (as opposed to Ethernet which was shared),
  - Fast (155Mbps and above),
  - Lack of distance limitation
- Problem with implementing broadcast if don’t know all node addresses and setup VCs to them
- Solution:
  - Redesign protocols e.g. ATMARP
  - LAN emulation (LANE) - effectively shared media emulation
**LAN emulation**

- **Addresses**: ATM address (used to establish VC), LANE MAC address, VCI
- **LANE** uses various servers and LAN emulation clients (LECs) – hosts, routers, etc to make LANE layer appear like standard MAC layers to higher layers

**Diagram:**

```
nodes: LES, BUS, H1, H2

arrows:
- ATM network (dashed line)
- Point-to-point VC (solid line)
- Point-to-multipoint VC (dashed line)
```

**LANE uses:**

- **LANE configuration server (LECS)**: collects ATM addresses of clients, supplies MAC parameters (LES, type, MTU etc)
- **LANE server (LES)**: clients register ATM/MAC addresses, gets addr of BUS
- **broadcast and unknown server (BUS)**: maintains point-to-multipoint VC to all registered clients, delivers multicast packets and first unicast between clients, supplies ATM addr corresponding to MAC addr
Switching Hardware

Overview

- Terminology: $n \times m$ switch has $n$ inputs and $m$ outputs
- Design Goals
  - throughput (depends on traffic model)
  - scalability (a function of $n$)
- Ports and Fabrics

  - input port (contend for fabric)
  - internal (contend for output port)
  - output port (contend for link)

  - circuit management (e.g., map VCs, route datagrams)
  - buffering (input and/or output)

  - as simple as possible
  - sometimes do buffering (internal)

Buffering

- Wherever contention is possible
  - input port (contend for fabric)
  - internal (contend for output port)
  - output port (contend for link)

- Head-of-Line Blocking
  - input buffering
Crossbar switches are nonblocking and simple but with $N^2$ complexity.

They have been used in small networks or as building blocks.

Although it is nonblocking, in packet mode it becomes a blocking network.

A queueing function is added to the crossbar to overcome this problem in three ways.

- Input queueing
- Output queueing
- Crosspoint queueing

A central controller sets up the cross-points and schedules the packet delivery. It is simple, but the central controller is the bottleneck.

It is very expensive for large switch.

Output queueing provides ideal performance.
Knockout Switch

- It is designed for packet switching with fixed length (Knockout I) or variable length (Knockout II) packets.
- It uses one broadcast input bus for each input port to all output ports.
- Each output port has a bus interface that prevents contention on the bus and allows simultaneous packets to the same output port.
- Packet filters detect the address of each packet and implement the self routing function.
- When two packets arrive to a $2 \times 2$ switch, one is selected randomly.

Example of Crossbar

- Boxes with $D$ introduce 1 bit delay to keep the competition synchronous.
- Concentrator: select $L$ of $N$ packets
- Complexity: $N^2$
- **Output Buffer**
- The concentrator selects $L$ packets and store them into a shared buffer based on their time of arrival.
- Selecting $L$ packets out of $N$ contenders is analogous to a knockout tournament.
- For an $N \times L$ knockout concentrator, there are $L$ rounds of competitions.
Self-Routing Fabrics

- **Banyan Network**
  - constructed from simple $2 \times 2$ switching elements
  - self-routing header attached to each packet
  - elements arranged to route based on this header
  - no collisions if input packets sorted into ascending order
  - complexity: $n\log_2 n$

- **Batcher Network**
  - switching elements sort two numbers
    - some elements sort into ascending (clear)
    - some elements sort into descending (shaded)
  - elements arranged to implement merge sort
  - complexity: $n\log_2 n$

- **Common Design: Batcher-Banyan Switching Fabric**
- **Sunshine Switch**
A Brief Summary of INs

- Most INs have been designed for a particular applications, such as voice data, signaling, etc.
- Different applications need different bandwidth requirements.
- Circuit switching concept has evolved to handle stream-type traffic (voice, video), with fixed throughput and constant delay.
- Packet switching has evolved as an efficient way to transport communication traffic with the following property.
  - Buffering
  - Statistical multiplexing
  - Variable throughput
  - Variable delay
  - It supports both virtual circuit and datagram techniques.
  - Very attractive for applications with low throughput and low delay, (inquiry/response), and high throughput and high delay (file transfer).
  - It is not suitable for real-time type traffic such as voice, video, and computer-to-computer data transfer.
Many switching fabrics have been designed to satisfy high-performance requirements that include:
- High degree of parallelism
- Distributed control
- Routing function on hardware

Fast packet switching can be classified based on their internal fabric structures that include:
- Buffered-based banyan fabrics
- Sort-banyan-based fabrics
- Fabrics with disjoint-path topology and output queueing
- Crossbar-based fabrics
- Time division fabrics with common packet memory
- Fabrics with shared medium.

**Banyan and Buffered Banyan Fabrics**

- Banyan is a rich class that includes regular and rectangular banyan networks, and delta networks.
- A $N = b^k$ delta network with self-routing property (delta-b) is constructed with $k$ stages of identical $b \times b$ switching elements.
- Many of the well-known INs such as omega, flip, cube, shuffle-exchange, and baseline belong to the class of delta networks.
- These networks are attractive for packet switching since several packets can be switched simultaneously and in parallel.
- Although these networks have different interconnection patterns, they have the same performance for packet switching.
- They all have the following properties:
  - All consists of $\log_b N$ stages of $N/b \times b$ switches.
  - Self-routing (digit-controlled) in which a unique $k$ digit base $b$ destination address is used.
  - They can be constructed in a modular fashion from smaller networks (block structured).
  - They can operate in synchronous or asynchronous mode.
  - Their regularity makes them attractive for VLSI implementation.
- These networks become inherently blocking regardless of being blocking, nonblocking, and rearrangeable in circuit-switched implementations because packets could collide with each other.
There are two types of blocking

- Internal link blocking: contention for a particular link inside the network.
- Output port blocking: two or more packets are contending for the same output port.

There are several ways to reduce the blocking (increase the throughput) of banyan switches.

- Increasing internal link speed.
- Placing buffers in every switching node.
- Using a handshaking or backpressure mechanism to delay the transfer of blocked packets.
- Using multiple network to provide multiple paths or using multiple links for each switch.
- Using a distribution network at the front of banyan for load balancing.

The Integrated Services Packet Network (ISPN) is based on large high performance packet switch structure.

The switch interfaces up to 1000 high-speed digital transmission facilities via packet processors (PP).

A PP provides input buffering, adds the routing header, and performs the link level protocol functions.

A control processor (CP) performs all connections control functions.

The switch fabric consists of a 10-stage self-routing buffered banyan with 1024 ports of 5120 buffered $2 \times 2$ switching elements.
The switch fabric uses backpressured flow control mechanisms between stages which prevent buffer overflow or packet loss.

- It uses the *virtual cut-through* buffering technique.
- When a packet arrives at a switching element and the output port is free, it bypasses the buffer and directly sends it to the output port.
- AT&T introduced a *wideband packet technology* network based on a $16 \times 16$ buffered banyan with 8 Mbits/s.
- Buffered banyan networks have been studied analytically and by simulations models based on different buffer-size, the position of buffers, traffic distributions, and switching size.

---

**Sort-Banyan-Based Fabrics**

- Banyan network is internally blocking; two packets destined for two different destinations may collide in one of the stages.
- If packets are first sorted based on their destination address, and then routed through the network, then the internal blocking can be avoided.
- Batcher-banyan is an example of sort-banyan network.
- Blocking still can happen if two or more packets have the same destination address.
Disjoint-Path and Output Queuing

- The switch fabrics is based on nonblocking fully interconnected topology.
- To resolve the output contention, output queuing is used.
- In general, a higher throughput can be achieved when output buffering is used since there is no HOL blocking.
- Infinite output buffering capability gives the best delay/throughput performance.

Fabric with Shared Medium

- A bus or ring network is used as switching medium.
- They provide flexibility in terms of access protocols and distribution of traffic.
- Their bandwidth and throughput are limited compared to multipath switch networks.
- Multiple rings or multiple buses can be used to increase the capacity.
- The frame duration of 125 μs maintains complete time transparency for circuit-switched channel.