Interconnection Networks
Outline

- Topologies
  - Properties
- Circuit and Packet Switched Networks
- Routing
- Virtual Channels
Where Is Interconnect Used?

- To connect components

- Many examples
  - Processors and processors
  - Processors and memories (banks)
  - Processors and caches (banks)
  - Caches and caches
  - I/O devices
Why Is It Important?

- Affects the scalability of the system
  - How large of a system can you build?
  - How easily can you add more processors?

- Affects performance and energy efficiency
  - How fast can processors, caches, and memory communicate?
  - How long are the latencies to memory?
  - How much energy is spent on communication?
Interconnection Network Basics

- **Topology**
  - Specifies the way switches are wired
  - Affects routing, reliability, throughput, latency, building ease

- **Routing (algorithm)**
  - How does a message get from source to destination
  - Static or adaptive

- **Buffering and Flow Control**
  - What do we store within the network?
    - Entire packets, parts of packets, etc?
  - How do we throttle during oversubscription?
  - Tightly coupled with routing strategy
Topology

- Bus (simplest)
- Point-to-point connections (ideal and most costly)
- Crossbar (less costly)
- Ring
- Tree
- Multistage: Omega, Butterfly
- Hypercube
- Mesh
- Torus
- ...


Metrics to Evaluate ICNs

- Cost (Area, Power)
- Latency (per SD pair, Average over all SD pairs)
- Contention
- Energy (per bit transferred), Overall communication energy
- Bandwidth (peak, average, bisection)
- System performance (Program Execution Time)
Bus

All nodes connected to a single link
+ Simple + Cost effective for a small number of nodes
+ Easy to implement coherence (snooping and serialization)
- Not scalable to large number of nodes (limited bandwidth, electrical loading, reduced frequency)
- High contention, fast saturation
Point-to-Point

Every node connected to every other with direct/isolated links

+ Lowest contention
+ Potentially lowest latency
+ Ideal, if cost is no issue

-- Highest cost
O(N) connections/ports per node
O(N^2) links
-- Not scalable
-- How to lay out on chip?
Crossbar

- Every node connected to every other with a shared link for each destination
- Enables concurrent transfers to non-conflicting destinations
- Could be cost-effective for small number of nodes

+ Low latency and high throughput
- Expensive
- Not scalable, $O(N^2)$ cost
- Difficult to arbitrate as $N$ increases

Used in core-to-cache-bank networks in
- IBM POWER5 onwards
- Sun Niagara I/II
Crossbar Design
Sun UltraSPARC T2 Core-to-Cache Crossbar

- High bandwidth interface between 8 cores and 8 L2 banks & NCU
- 4-stage pipeline: req, arbitration, selection, transmission
- 2-deep queue for each src/dest pair to hold data transfer request
Multistage Logarithmic Networks

- Idea: Indirect networks with multiple layers of switches between terminals/nodes
- Cost: $O(N \log N)$, Latency: $O(\log N)$
- Many variations (Omega, Butterfly, Benes, Banyan, ...)
- Omega Network:
Circuit vs. Packet Switching

**Circuit switching** sets up full path before transmission
- Establish route then send data
- No one else can use those links while “circuit” is set
  + faster arbitration
-- setting up and bringing down “path” takes time

**Packet switching** routes per packet in each router
- Route each packet individually (possibly via different paths)
- If link is free, any packet can use it
-- potentially slower --- must dynamically switch
  + no setup, bring down time
  + more flexible, does not underutilize links
A multistage network has more restrictions on feasible concurrent Tx-Rx pairs vs a crossbar
But more scalable than crossbar in cost, e.g., $O(N \log N)$ for Butterfly
Packet Switched MIN

Packets “hop” from router to router, pending availability of the next-required switch and buffer
Ring

Each node connected to exactly two other nodes. Nodes form a continuous pathway such that packets can reach any node.

+ Cheap: $O(N)$ cost
- High latency: $O(N)$
- Not easy to scale
  - Bisection bandwidth remains constant

Used in Intel Haswell, Intel Larrabee, IBM Cell, ...
Hierarchical Rings

- More scalable
- Lower latency
- More complex

(a) 4-, 8-, and 16-bridge hierarchical ring topologies.

(b) Three-level hierarchy (8x8).
Mesh

- Each node connected to 4 neighbors (N, E, S, W)
- $O(N)$ cost
- Average latency: $O(\sqrt{N})$
- Easy to layout on-chip: regular and equal-length links
- Path diversity: many ways to get from one node to another

- Used in Tilera 100-core
- And many on-chip network prototypes
Torus

- Mesh is not symmetric on edges: performance very sensitive to placement of task on edge vs. middle
- Torus avoids this problem
  + Higher path diversity (and bisection bandwidth) than mesh
  - Higher cost
  - Harder to lay out on-chip
    - Unequal link lengths
Torus, continued

- Weave nodes to make inter-node latencies ~constant
Hypercube

- “N-dimensional cube” or “N-cube”

- Latency: $O(\log N)$
- Radix: $O(\log N)$
- #links: $O(N\log N)$

+ Low latency
- Hard to lay out in 2D/3D
Interconnection Network Basics

- **Topology**
  - Specifies the way switches are wired
  - Affects routing, reliability, throughput, latency, building ease

- **Routing (algorithm)**
  - How does a message get from source to destination
  - Static or adaptive

- **Buffering and Flow Control**
  - What do we store within the network?
    - Entire packets, parts of packets, etc?
  - How do we throttle during oversubscription?
  - Tightly coupled with routing strategy
Handling Contention

Two packets trying to use the same link at the same time

- What do you do?
  - Buffer one
  - Drop one
  - Misroute one (deflection)

- Tradeoffs?
**Bufferless Deflection Routing**

- **Key idea:** Packets are never buffered in the network. When two packets contend for the same link, one is deflected.

New traffic can be **injected** whenever there is a free output link.

---

Buffered Flow Control

A buffer between two channels decouples the resource allocation for each channel.

Packet-buffer flow control: channels and buffers are allocated per packet.

Store-and-forward

Cut-through

Wormhole routing: same as cut-through, but buffers in each router are allocated on a per-flit basis, not per-packet.
Flits do not carry headers. Once a packet starts going over a channel, another packet cannot cut in. If the packet is stalled, other packets can’t use the channel.

With VCs, the flit can be received into one of N buffers. This allows N packets to be in transit over a given physical channel. The packet must carry an ID to indicate its virtual channel.
Routing Algorithm

**Deterministic:** always chooses the same path for a communicating source-destination pair. Eg. Dimension Ordered Routing (XY)

- **Oblivious:** chooses different paths, without considering network state

- **Adaptive:** can choose different paths, adapting to the state of the network. Deals with unexpected events (faults, congestion). More router complexity. Potentially better performance. Deadlocks have to be handled explicitly.

- **How to adapt:**
  - Local/global feedback
  - Minimal or non-minimal paths
Deterministic Routing

- All packets between the same (source, dest) pair take the same path

- **Dimension-order routing**
  - First traverse dimension X, then traverse dimension Y
  - E.g., XY routing (used in Cray T3D, and many on-chip networks)

+ Simple
+ Deadlock freedom (no cycles in resource allocation)
- Could lead to high contention
- Does not exploit path diversity
Deadlock

Deadlock happens when there is a cycle of resource dependencies – a process holds on to a resource (A) and attempts to acquire another resource (B) – A is not relinquished until B is acquired.
Deadlock Example

Each message is attempting to make a left turn – it must acquire an output port, while still holding on to a series of input and output ports.
Handling Deadlocks

- Avoid cycles in routing
  - Dimension order routing (XY)
    - Cannot build a circular dependency
    - Restrict the “turns” each packet can take

- Avoid deadlock by adding more buffering (escape paths)

- Detect and break deadlock
  - Preemption of buffers
Deadlock Free Proofs

- Number edges and show that all routes will traverse edges in increasing (or decreasing) order – therefore, it will be impossible to have cyclic dependencies.
- Example: k-ary 2-d array with dimension routing: first route along x-dimension, then along y.
Adaptive Routing

- **Minimal adaptive**
  - Router uses network state (e.g., downstream buffer occupancy) to pick which “productive” output port to send a packet to
  - Productive output port: port that gets the packet closer to its destination
  + Aware of local congestion
- Minimality restricts achievable link utilization (load balance)

- **Non-minimal (fully) adaptive**
  - “Misroute” packets to non-productive output ports based on network state
  + Can achieve better network utilization and load balance
- Need to guarantee livelock freedom
On-Chip Networks

- Connect **cores, caches, memory controllers, etc**
- Buses and crossbars are not scalable
- **Packet switched**
- **2D mesh**: Most commonly used topology
- Primarily serve **cache misses** and **memory requests**
Slide Contents

• Onur Mutlu, 18-447, CMU
• R Balasubramonian, CS6810, Utah