NOC: Networks on Chip
SoC Interconnection Structures

COE838/EE8221: Systems-on-Chip Design
http://www.ecb.torontomu.ca/~courses/coe838/

Dr. Gul N. Khan
http://www.ecb.torontomu.ca/~gnkhan
Electrical, Computer & Biomedical Engineering
Toronto Metropolitan University

Overview

- Interconnection Networks on a Chip
- Bus and NoC/Network-on-Chip Systems
- NoC topologies
- Routing and Switching Techniques
- Flow Control

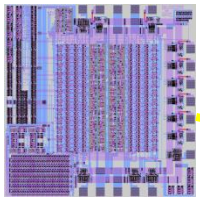
Chapter 5: Computer System Design – System on Chip by M.J. Flynn and W. Luk
Chapter 12: On-Chip Communication Architectures – SoC Interconnect by S. Pasricha & N. Dutt

SoC Integration and Interconnect Architecture

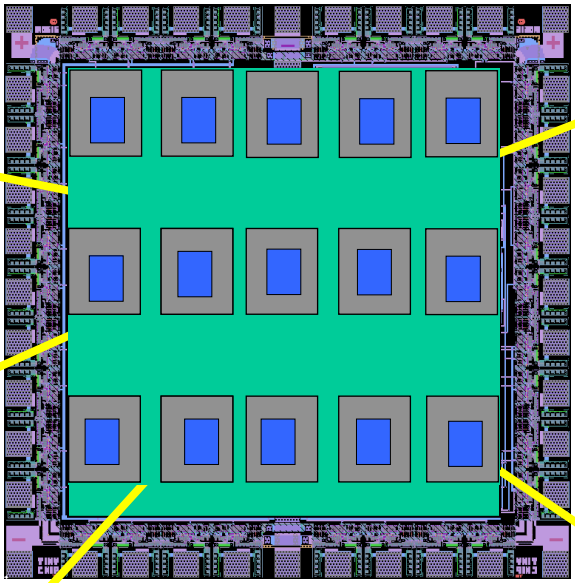
- **Integration: Critical part of SoC design**
  - Integration of SoC cores.
  - Connect the processing and other IP cores.
  - Maximize the reuse of design for lower cost.
- **SoC Interconnect Architectures**
  - Bus-based Interconnection.
  - NoC (Network on Chip) : NoC hides the physical interconnects from the designer.

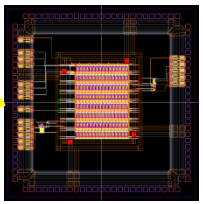
System-on-Chip and NoC

System-on-Chip and Network-on-Chip

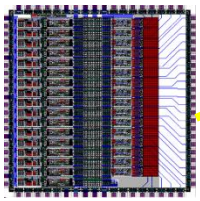


MPEG CORE



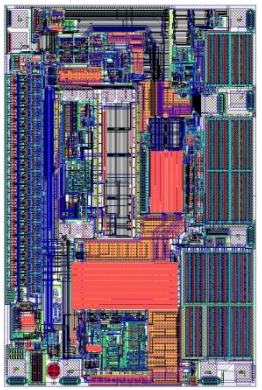


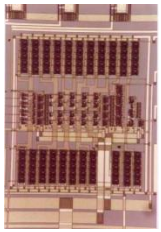
CPU Processing Cores



VGA/HDMI CORE

DSP

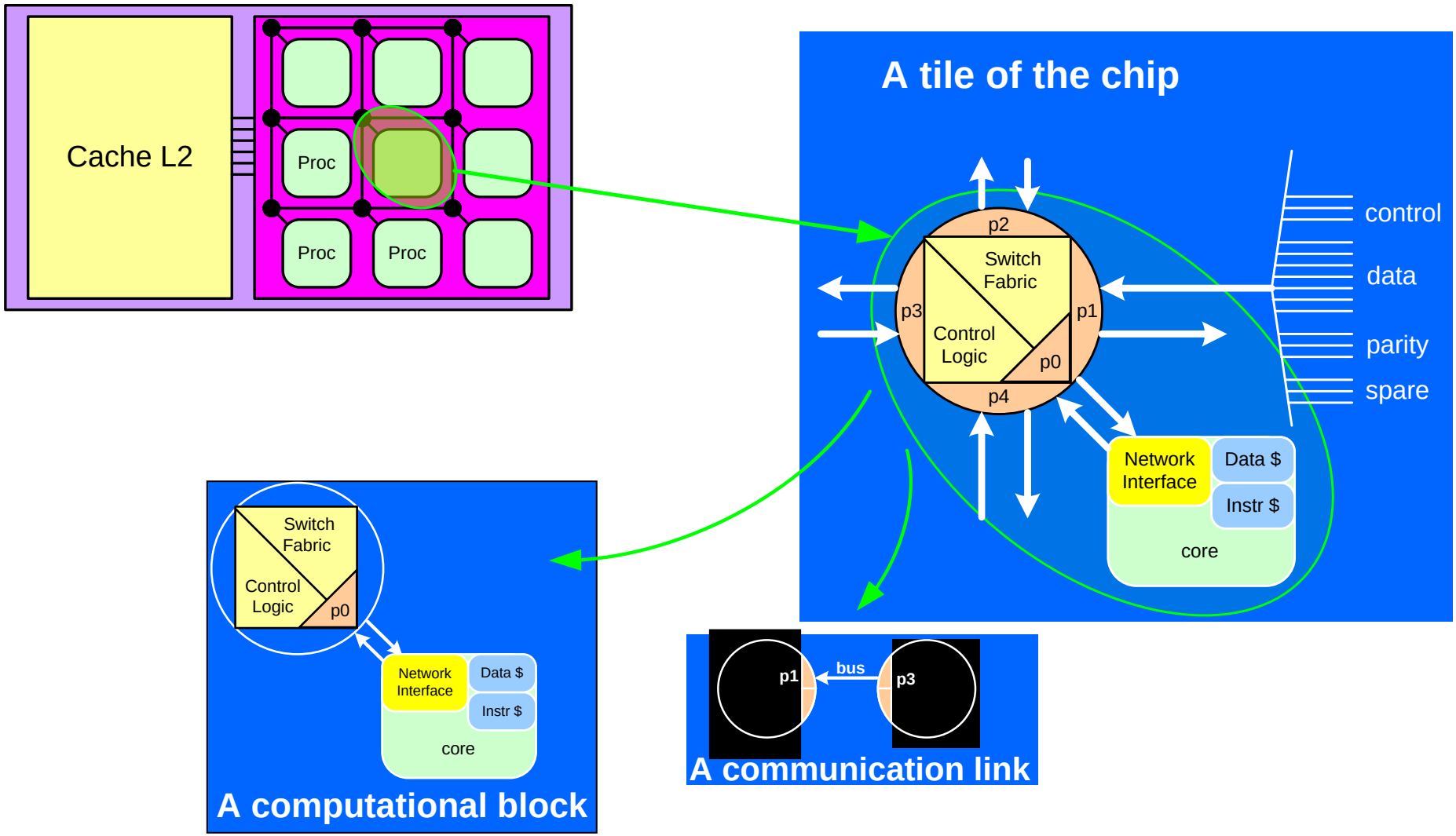




Analog ADC/DAC

Components

Typical SoC (CPU) Structure

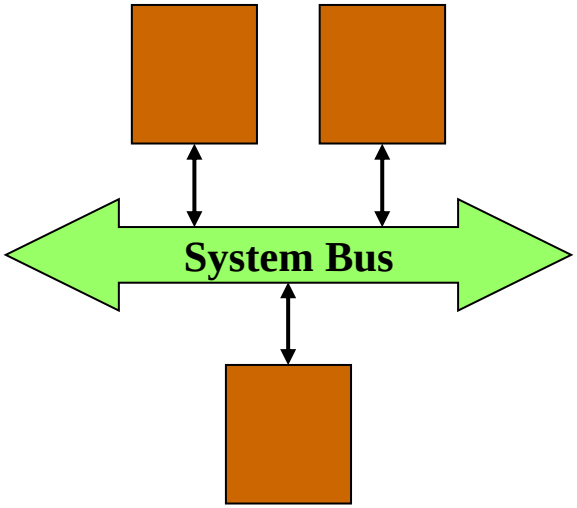


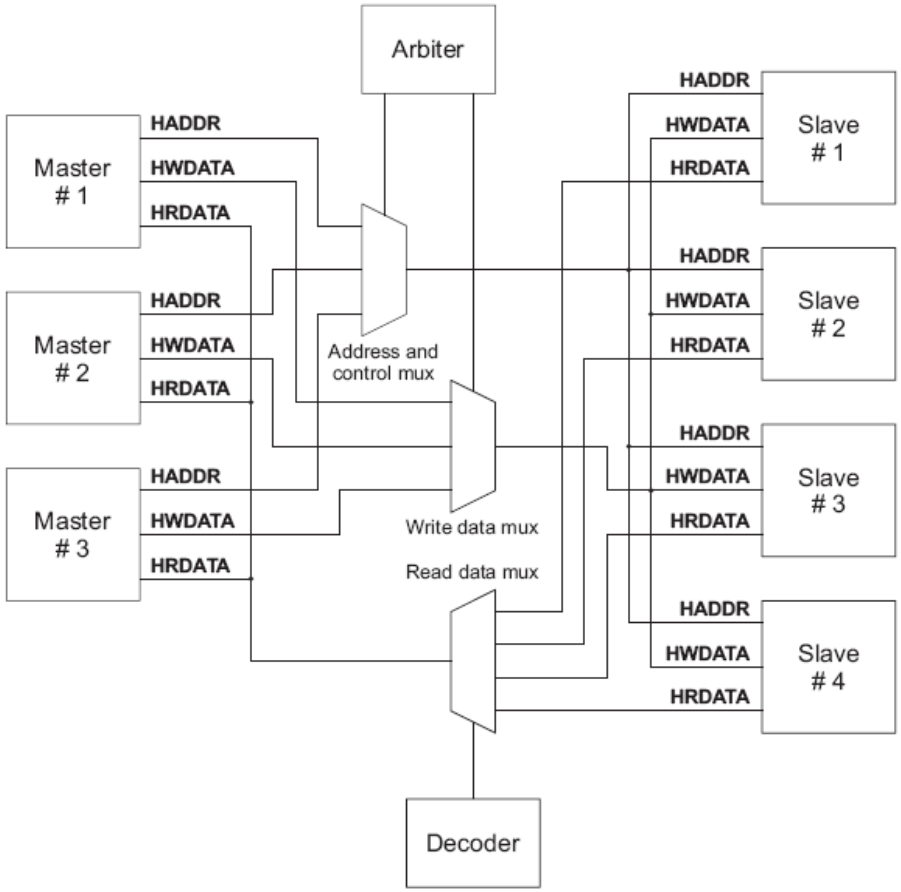
MPSoC: Multiple Processor (Cores) SoC

Communication between CPU/cores can be performed by message passing or shared memory. Number of processing cores on the same chip-die is increasing.

- **Memory sharing will require: SHARED BUS**
  * Large Multiplexers
  * Cache coherence techniques
  * Not Scalable
- **Message Passing: NOC**
  * Scalable
  * Require data transfer transactions
  * Has overhead of extra communication

Why NOC (Network-on-Chip)





Shared bus is not a long-term solution

- It has poor scalability

On-Chip micro-networks suit the demand of scalability and performance

On-Chip and Off-Chip Networks

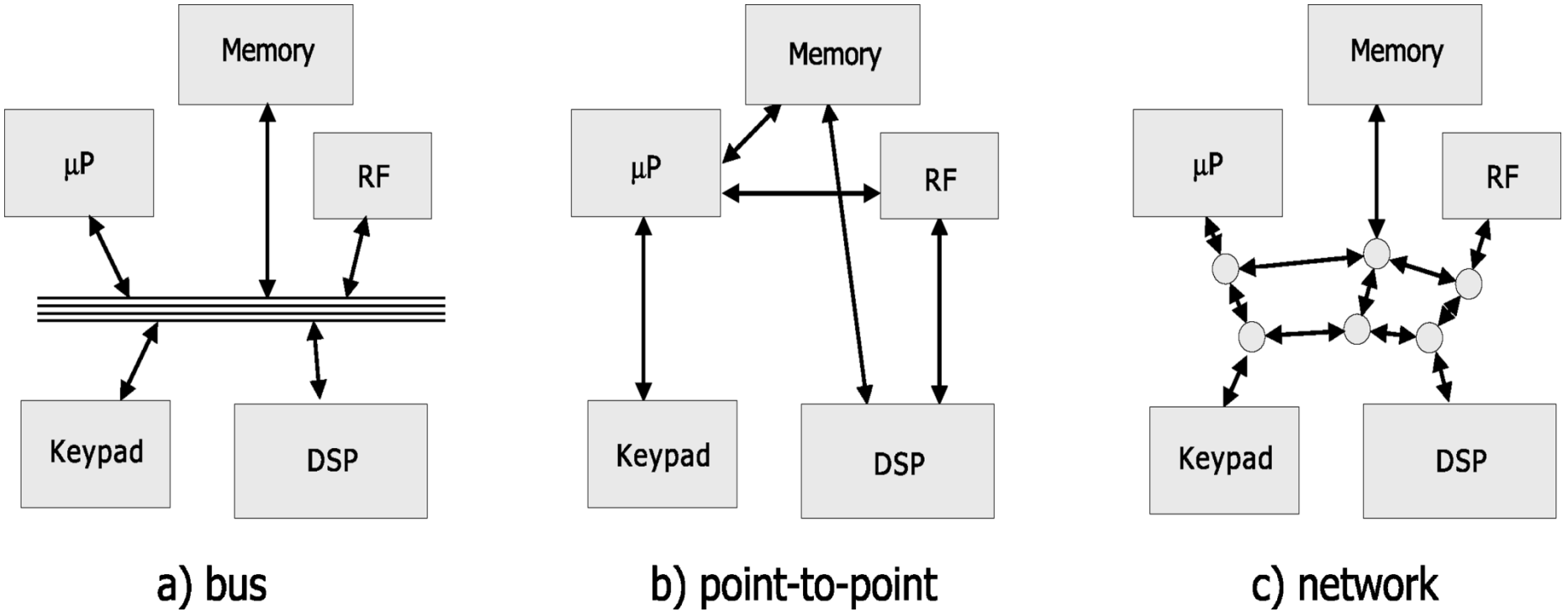
NOC (On-Chip)

- Sensitive to cost:
  - area and power
- Wires are relatively cheap
- Latency is critical
- Traffic is known a-priori
- Design time specialization
- Custom NoCs are possible

Off-Chip Networks

- Cost is in the links
- Latency is tolerable
- Traffic/applications unknown
- Changes at runtime
- Adherence to networking standards

On-Chip Communication Structures



(a) Bus based Communication

(b) Point to Point Communication

(c) Network based Communication

On-Chip Communication Structures

Dedicated Point-to-Point

- **Advantages**
  - **Optimal in terms of bandwidth, availability, latency and power usage**
  - **Simple to design and verify as well as easier to model**
- **Disadvantages**
  - **Number of links may increase exponentially with the increase in number of cores**
  - **Hardware Area**
  - **Routing Problems**

On-Chip Communication Structure
Network on Chip

Advantages

- Structured architecture – Lower complexity and cost of SoC design
- Reuse of components, architectures, design methods and tools
- Efficient and high-performance interconnect
- Scalability of communication architecture

Disadvantages

- Internal network contention can cause a latency
- Bus oriented IPs need smart wrapping hardware
- Software needs clear synchronization in multiprocessor systems

Networks-on-Chip

Interconnect for SoCs, MPSoC and FPGAs

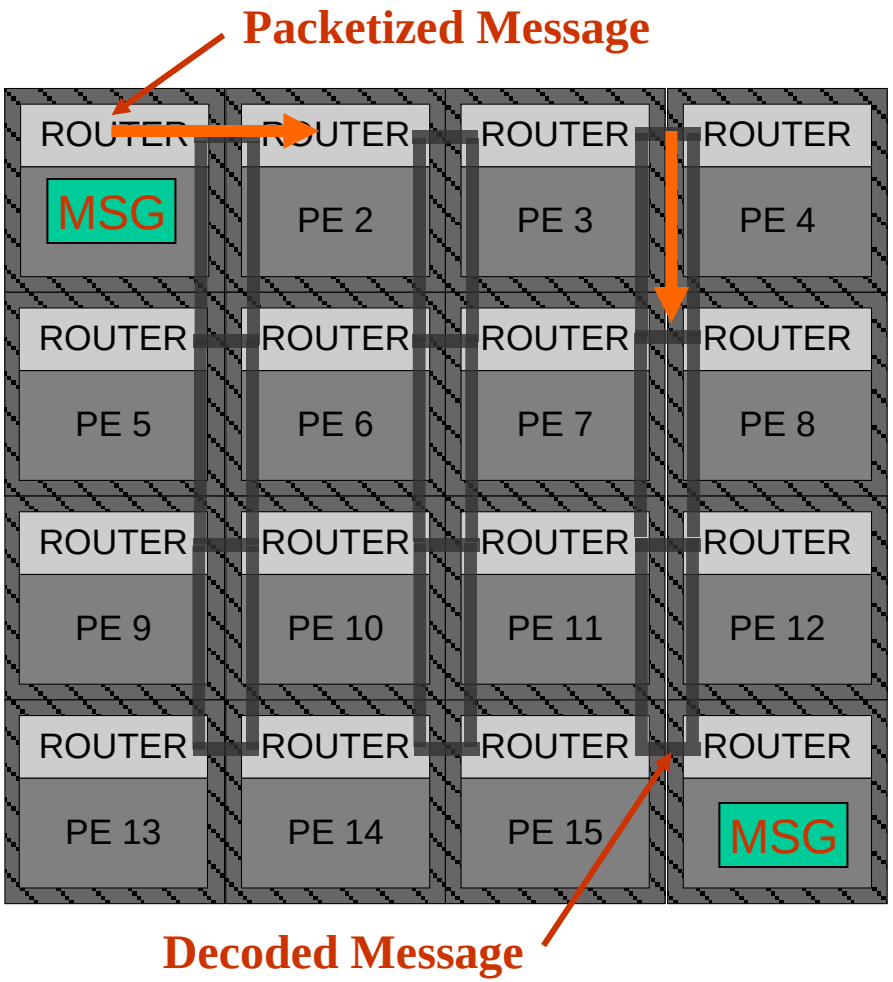
- Multi-hop, packet-based communication
- Efficient resource sharing

Scalable communication provide scalable performance/efficiency in

- Power
- Hardware Area
- Design productivity

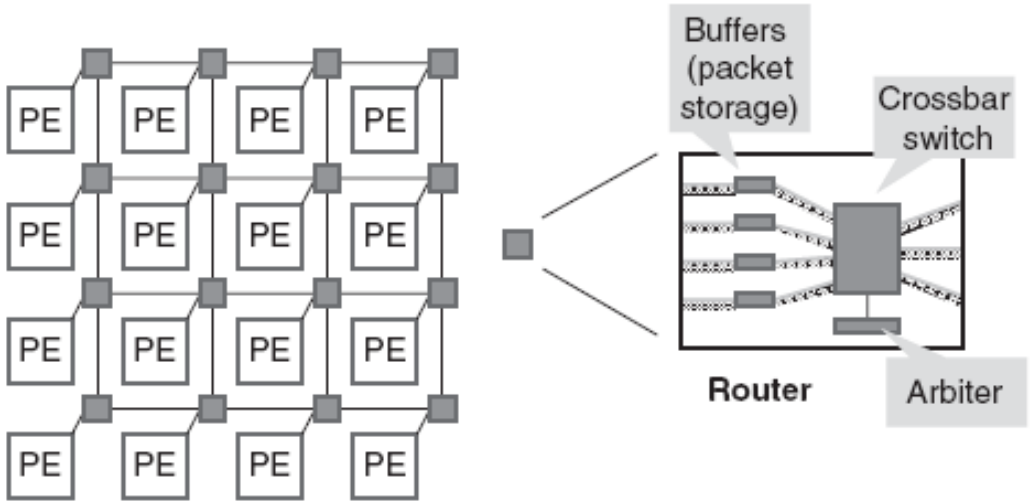
NoC ?

A chip-wide network: Processing Elements (PEs) or other cores are inter-connected via a packet-based network.



What is an NoC?

- Network-on-chip (NoC) is a packet switched on-chip communication network designed using a layered methodology
  - "routes packets, not wires"
- NoCs use packets to route data from the source to the destination PE via a network fabric that consists of
  - switches (routers)
  - interconnection links (wires)

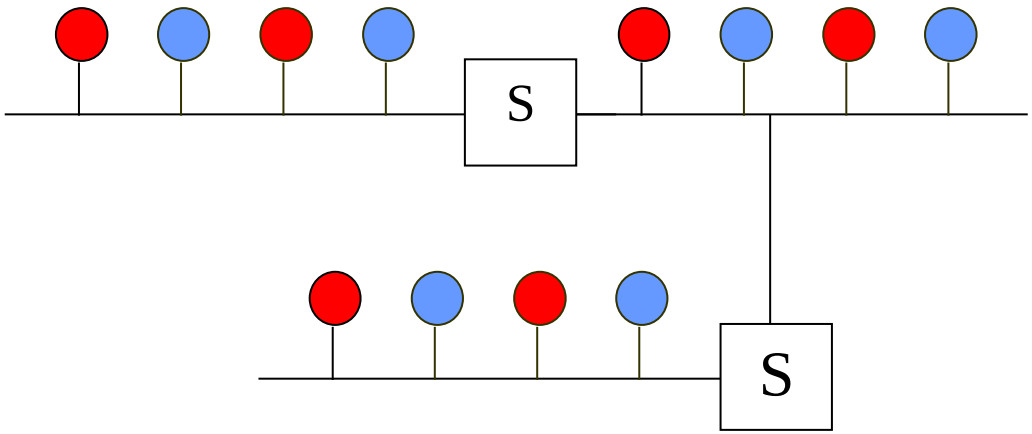


NoC: Buses to Networks

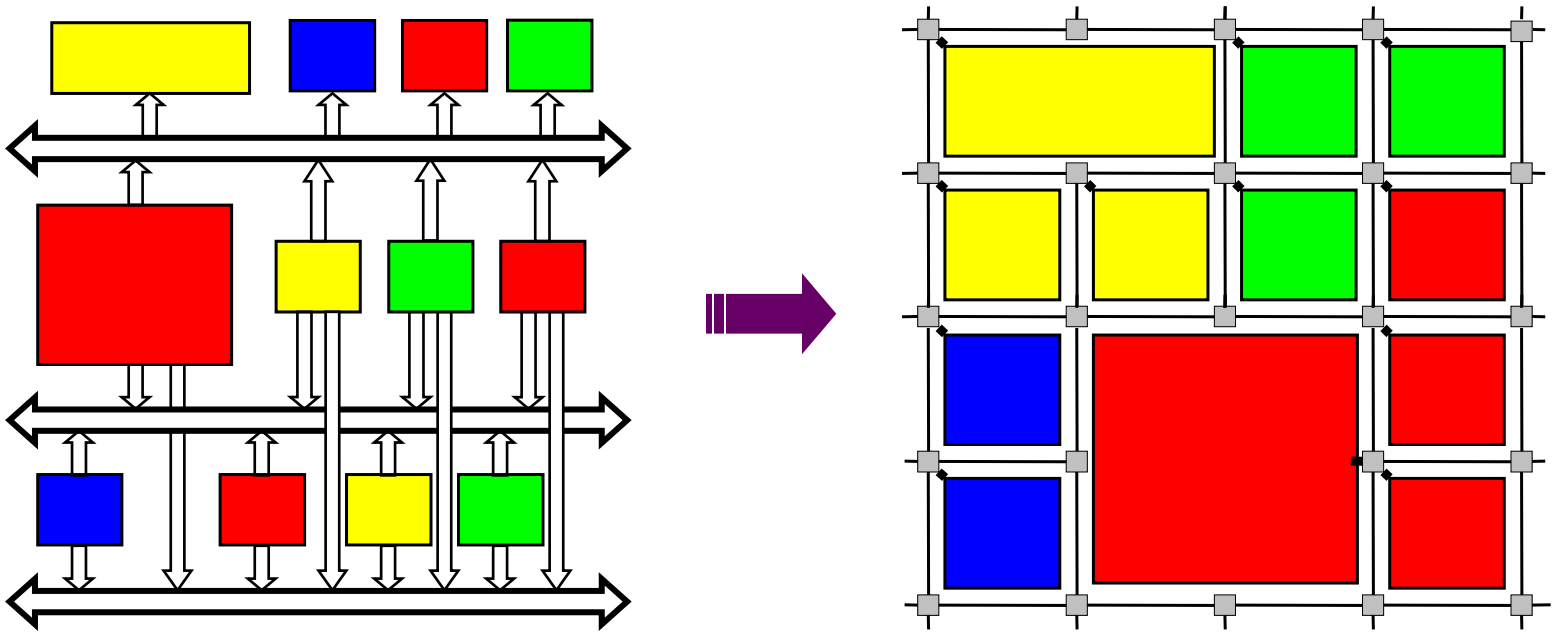
Original Bus Features

- One transaction at a time
- Central Arbiter
- Limited bandwidth
- Synchronous
- Low cost

Shared Bus to Segmented Bus



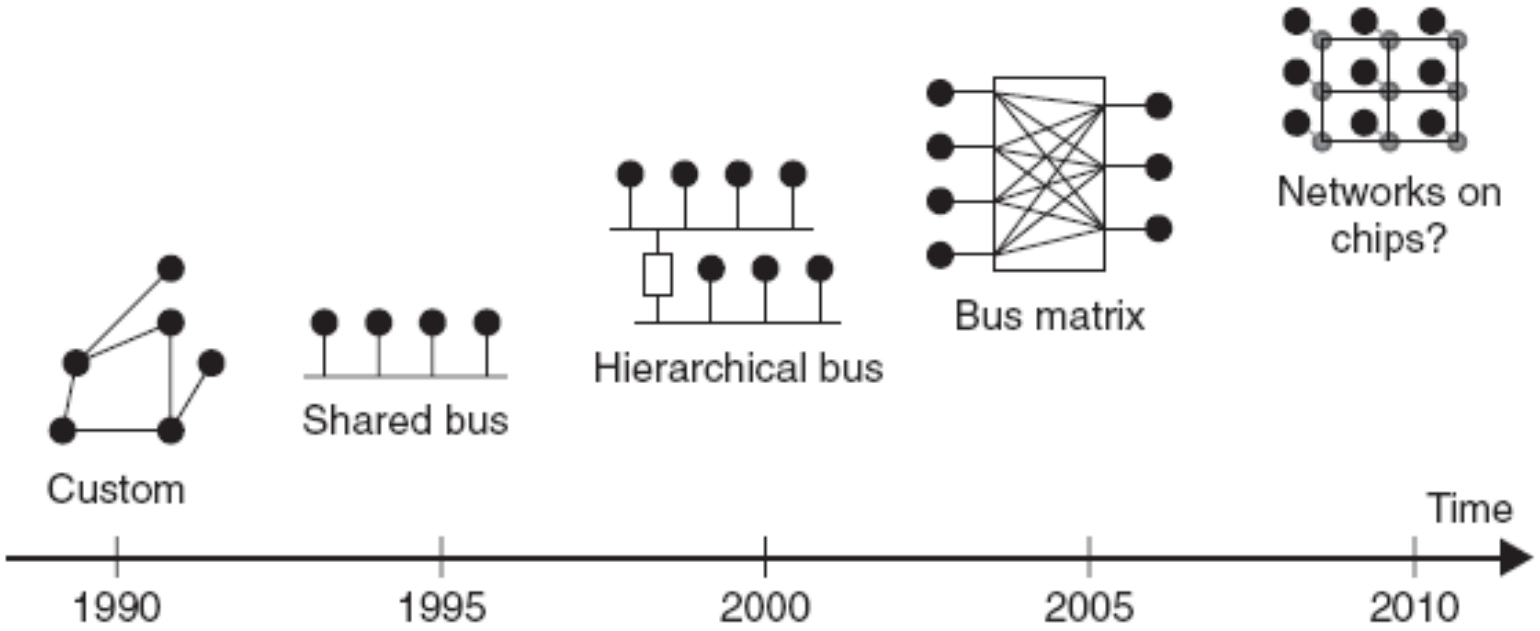
Buses to NoC



- Architectural paradigm shift: Replace wire spaghetti by network
- Usage paradigm shift: Pack everything in packets
- Organizational paradigm shift
  - Confiscate communications from logic designers
  - Create a new discipline, a new infrastructure responsibility

NoC Evolution

- Progress of on-chip communication architectures



Network-on-Chip Properties

- NoCs are an attempt to scale down the concepts of large-scale networks, and apply them to the embedded system-on-chip (SoC) domain
- NoC Properties
  - Regular geometry that is scalable
  - Flexible QoS guarantees
  - Higher bandwidth
  - Reusable components
    - Buffers, arbiters, routers, protocol stack
  - No long global wires (or global clock tree)
    - No problematic global synchronization
    - GALS: Globally asynchronous, locally synchronous design
  - Reliable and predictable electrical and physical properties

NoC Related Main Problems

Global interconnect design problems:

- Delay
- Power
- Noise
- Scalability
- Reliability

System integration:

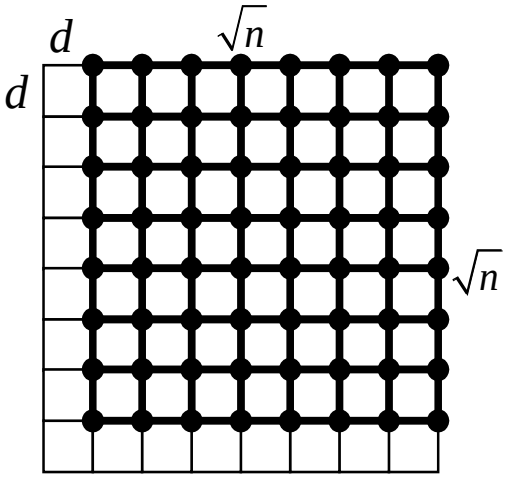
Productivity problem

NoC Scalability

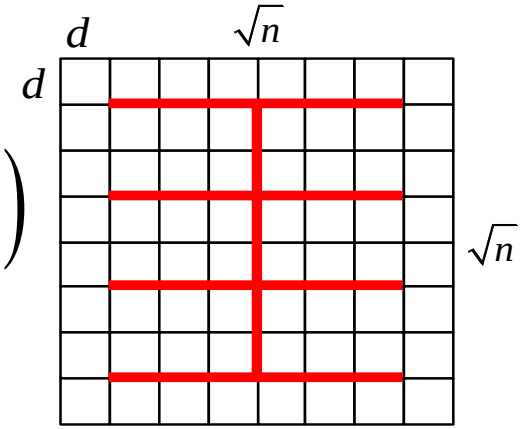
Compare the wire-area for same performance.

Wire-area cost for achieving same performance

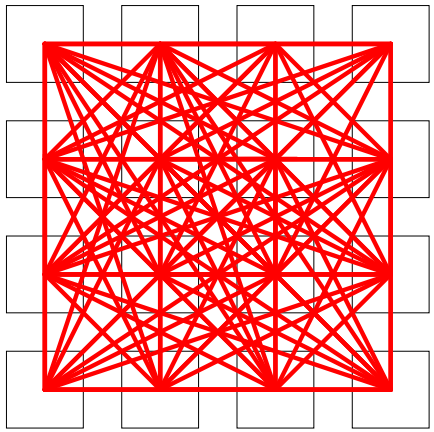
NoC: $O(n)$



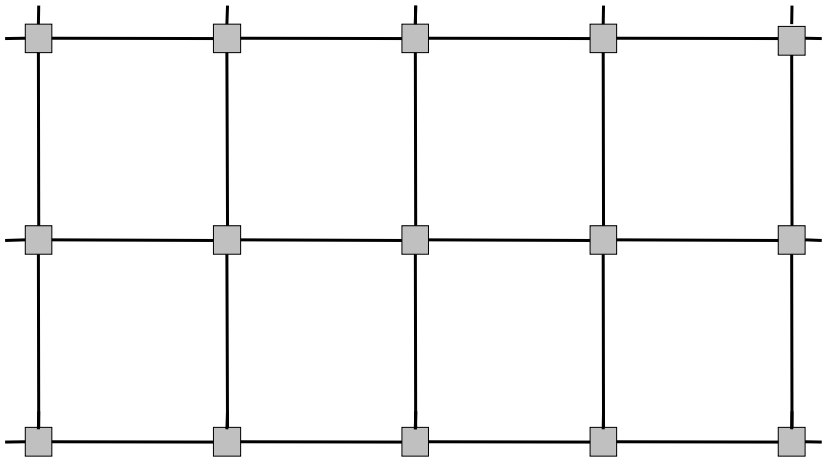
Bus $O(n^3\sqrt{n})$



Pt-to-Pt: $O(n^2\sqrt{n})$



NoC Wiring Design



- NoC links:
  - Regular
  - Point-to-point -- no fan-out tree (problem)
  - Can use transmission-line layout
  - Well-defined current return path
- Can be optimized for noise / speed / power
  - Low swing, current mode, ….

On-Chip Network Terminology

Network interface: Connects endpoints (cores) to network. Decouples computation/communication

Links: Bundle of wires that carries a signal

Switch/Router: It connects fixed number of input channels to fixed number of output channels

Channel: A single logical connection between routers/switches

Node: A network endpoint connected to a router/switch

Message: Unit of transfer for clients (e.g., cores, memory, etc.)

Packet: Unit of transfer for network

Flit: Flow control digit, where flow control is within the on-chip network

On-Chip Network Basics

Topology

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

Routing

- How does a message get from source to destination
- Static or adaptive

Flow Control and Buffering

- Storing message/data in the network.
- Storing the entire packets or flits (parts of packet), etc.
- Managing the buffer space.
- How to throttle during over subscription?

  Depends on the routing strategy also.

NoC Topology: Overview

NoC topology is the arrangement of channels and nodes in an on-chip network. Similar to a map of roads.

It can be a first thing for an on-chip network design

Significant impact on the cost-performance of an on-chip network.

Switch/Router (Node) Degree: number of links at a node used to estimate the **cost.**

Higher degree->more links/ports of each router.

Hop Count: number of hops a message takes from source to destination. Good for estimating **latency.**

Implement **complexity:** Node degree, ease of layout.

Network Diameter: Large min hop count in the on-chip network

Path Diversity: Multiple shortest paths between source and destination pair.

Useful for fault tolerance, better load balancing in the network and Routing algorithm can also exploit path diversity

NoC Topologies

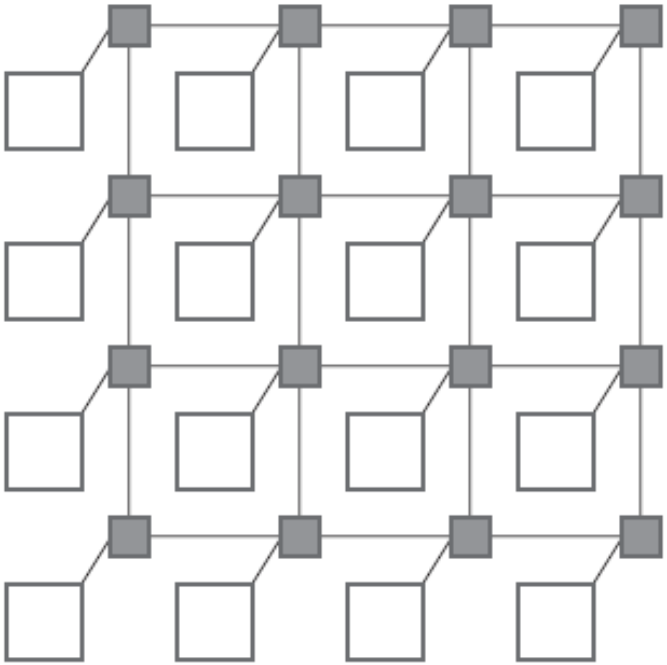
Direct Topologies

Node is both a switch and a core.

- each node has direct point-to-point link to a subset of other nodes in the system called neighboring nodes.
- nodes consist of computational blocks and/or memories, as well as a NI block that acts as a router e.g., Nostrum, SOCBUS, Proteo, etc.
  - as the number of nodes in the system increases, the total available communication bandwidth also increases.
  - fundamental trade-off is between connectivity and cost.

2D-mesh Topology

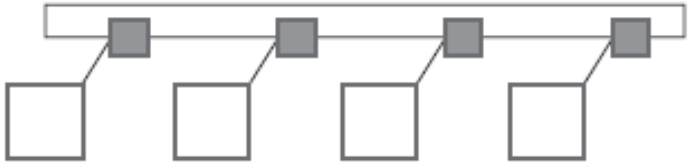
- Most direct network topologies have an orthogonal implementation, where nodes can be arranged in an n-dimensional orthogonal space
  - Routing for such networks is a bit simple
  - e.g., n-dimensional mesh, torus, hypercube, octagon
- 2D mesh is most popular topology
  - All links have the same length
    - eases physical design
  - Chip area grows linearly with the number of nodes
  - Must be designed in such a way as to avoid traffic accumulating in the center of the mesh



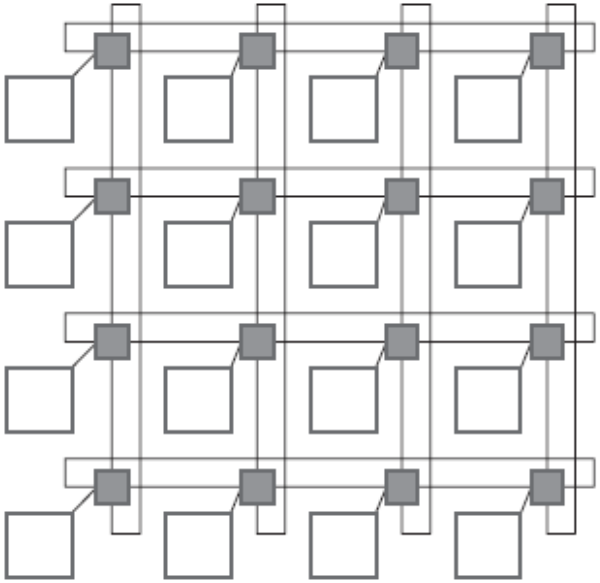
NoC Torus Topology

Torus topology (k-ary n-cube) is n-dimensional grid with k nodes in each dimension

- k-ary 1-cube (1-D torus) is essentially a **ring** network with k nodes
  - * Cheap to implement
  - * High latency as More cores = more delay to traverse the ring
  - * Not easy to scale



- k-ary 2-cube (**2-D torus**) topology is analogous to a regular mesh
  - Except that nodes at the edges are connected to switches at the opposite edge via wrap-around channels
  - Long end-around connections can, however, lead to excessive delays

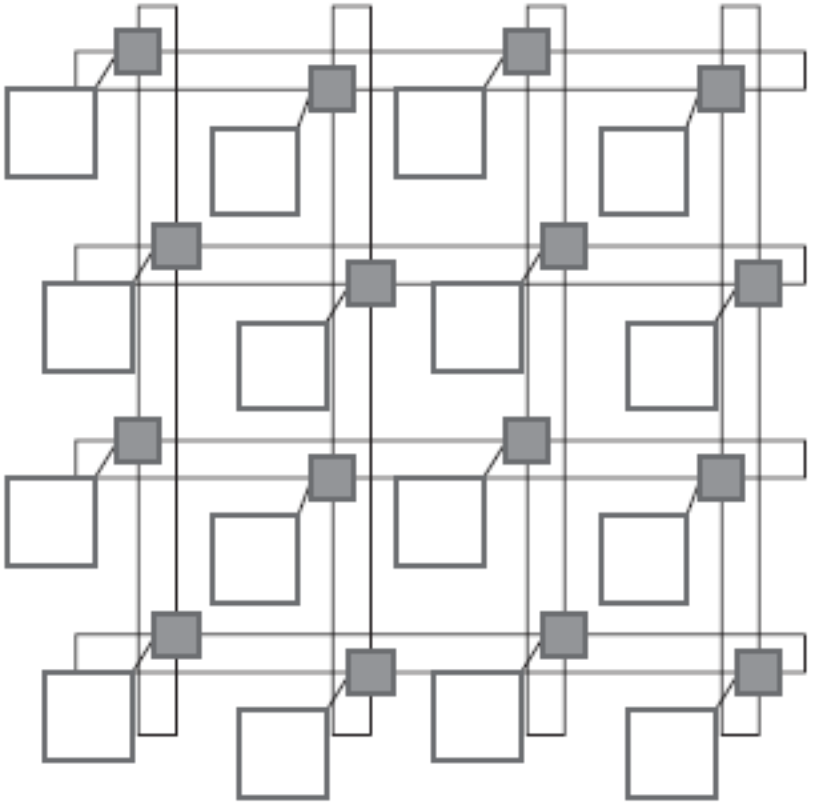


Folding Torus Topology

Folding torus topology overcomes the long link limitation of a 2-D torus having same size links.

- Map well to planar substrate for on-chip
- Topologies in Torus Family
  e.g., Ring -- k-ary 1-cube
- Edge Symmetric
  - Good for load balancing
  - Removing wrap-around links for mesh loses edge symmetry.

  Then More traffic concentrated on center channels
- Exploit locality for near-neighbor traffic



Meshes and tori can be extended by adding bypass links to increase performance at the cost of higher area

Indirect Topologies

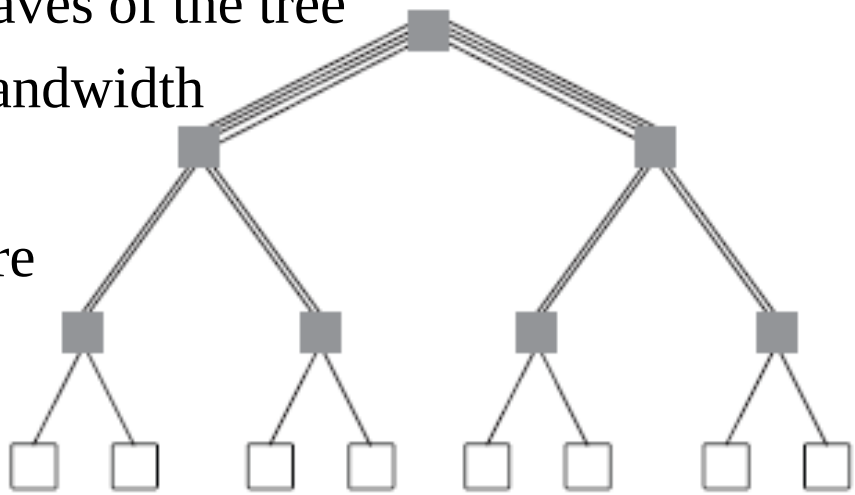
Indirect Topologies

Nodes can be **either** a switch **or** a core

- each node is connected to an external switch, and switches have point-to-point links to other switches
- switches do not perform any information processing, and correspondingly nodes do not perform any packet switching. e.g., crossbar-based topologies

Fat tree topology (Hierarchical Topology)

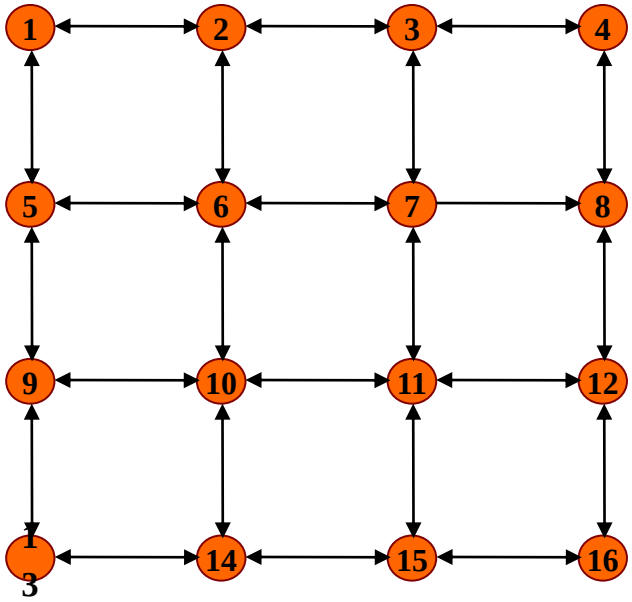
- cores are connected only to the leaves of the tree
- more links near root, for higher bandwidth
- Good for local traffic
- Easy to layout and low in hardware
- Disadvantage »
  Root can become a bottleneck

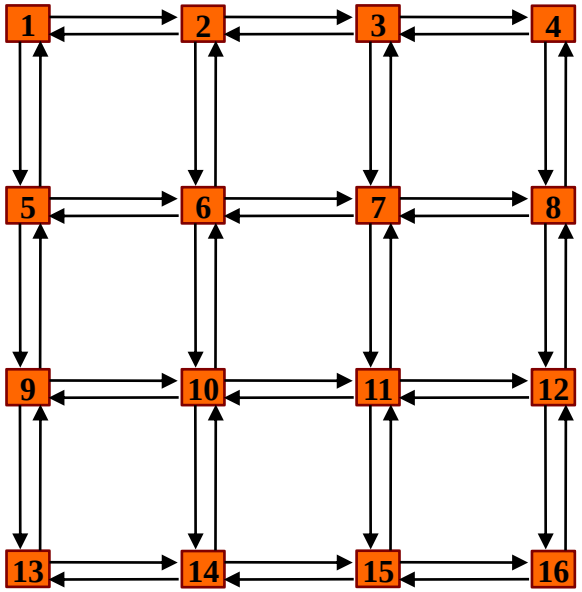


Irregular NoC Topologies

- Based on the concept of using only what is necessary.
- Application-specific topologies.
- Eliminate unneeded resources and bandwidth from the system.
- Leads to reduced power and area use.
- Requires additional design work.

Mesh Topology





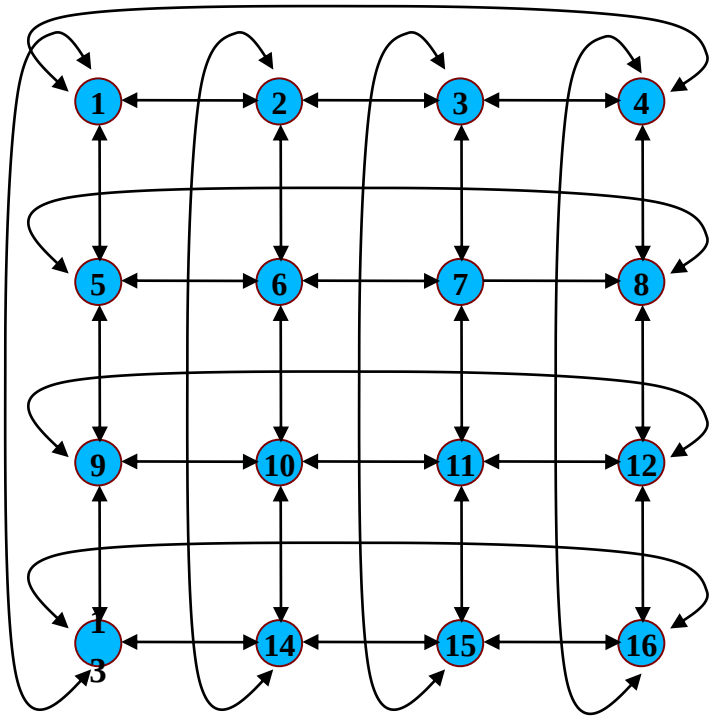
Physical (layout) implementation

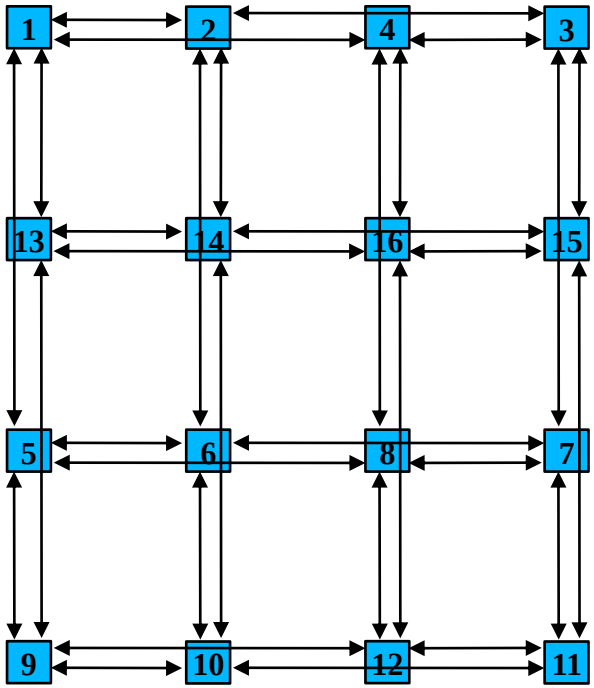
Easy to layout on-chip

- Equal and regular length links

Path diversity – many ways to get from one core to another

2D-Torus Topology





Physical Layout implementation

- Mesh performance disadvantage: Traversing from one edge to other. Torus solves this problem.
- Higher path diversity than mesh topology.
- Higher cost and hard to layout due to different link lengths.

Circuit and Packet Switching

Circuit Switching sets up full path

- Establish the route and then send data
  no one else can use those links
- Fast and high bandwidth.
- Set-up and bringing down links is slow.

Packet Switching routes packets

- Route each packet individually
  may be through different paths
- If link is free – it can be used.
- Slower (must dynamically switch).
- No setup OR bring down time.

Routing Algorithms

Responsible for correctly and efficiently routing packets from source to the destination

- Choice of a routing algorithm depends on trade-offs between several potentially conflicting metrics
  - Minimizing power and logic hardware (routing tables) for lower area footprint
  - Higher performance by lower delay and maximizing traffic utilization.
  - Improving robustness to better adapt to changing traffic needs

Routing schemes classified based on several categories

- Static or dynamic routing
- Distributed or source routing
- Minimal or non-minimal routing

Static and Dynamic Routing

Static Routing: fixed paths are used to transfer data between a particular source and destination

- does not consider the current state of the network
- easy to implement, since very little additional router logic is required
- in-order packet delivery if single path is used

Dynamic Routing: Routing decisions are made according to the current state of the network

Main factors considered are the availability and load on links

- Path between source and destination may change over time with the traffic conditions and requirements of the application
- More resources needed to monitor the network state and dynamically changing the routing paths.
- Better distribution of traffic in the network.

Distributed and Source Routing

Static/dynamic routing can be further classified depending on where the routing information is stored, and where the routing decisions are made.

Distributed Routing: Each packet carries the destination address

- XY co-ordinates or ID of the destination node/router
- Routing decisions are made in each router by using the destination address in a routing table (LUT) or by executing a function.

Source Routing: Packet carries the routing information

- Pre-computed routing tables are stored at the node's NI.
- When a packed is released, the routing information is taken from the source NI and added to the header of the packet (large packet size)
- When the packet arrives at a router, the routing information is extracted from the packet header to continue packet routing.
- No need of a destination address in the packet, routing tables, or functions needed to calculate the route.

Minimal and Non-minimal Routing

Minimal Routing: Routing path from the source to the destination is the shortest possible between two nodes.

- For mesh topology (each node can be identified by its XY co-ordinates) if source node is at (0, 0) and destination node is at (i, j), then the minimal path length is |i| + |j|
- source start sending a packet if a minimal path is available

Non-minimal Routing: It can use longer paths when a minimal path is not available.

- By allowing non-minimal paths, there are more alternative paths that is good for avoiding network congestion.
- Main overhead is the additional power consumption.

Deadlock, Livelock, and Starvation

Deadlock: A packet does not reach its destination, because it is blocked at some intermediate resource.

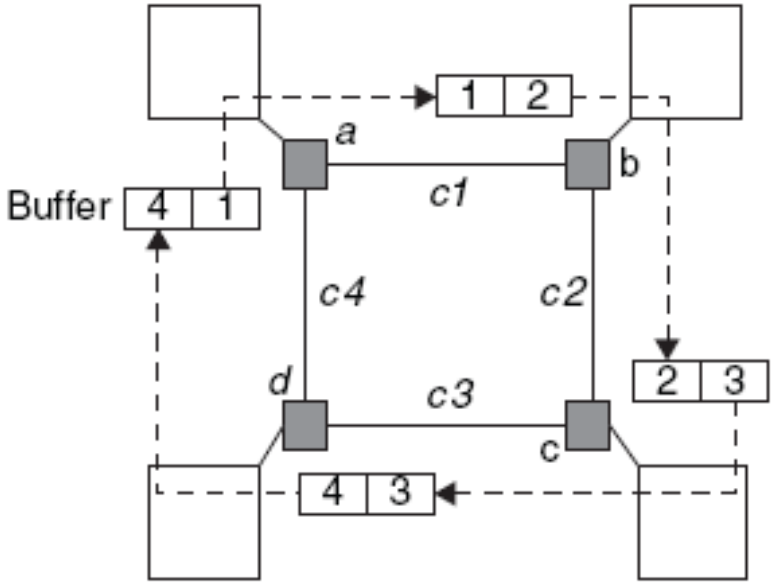
Livelock: A packet does not reach its destination, because it enters a cyclic path.

Starvation: A packet does not reach its destination, because some resource does not grant access (while it grants access to other packets).

Routing Algorithms

Routing algorithm must ensure freedom from deadlocks

- Common in WH routing. e.g., cyclic dependency shown below

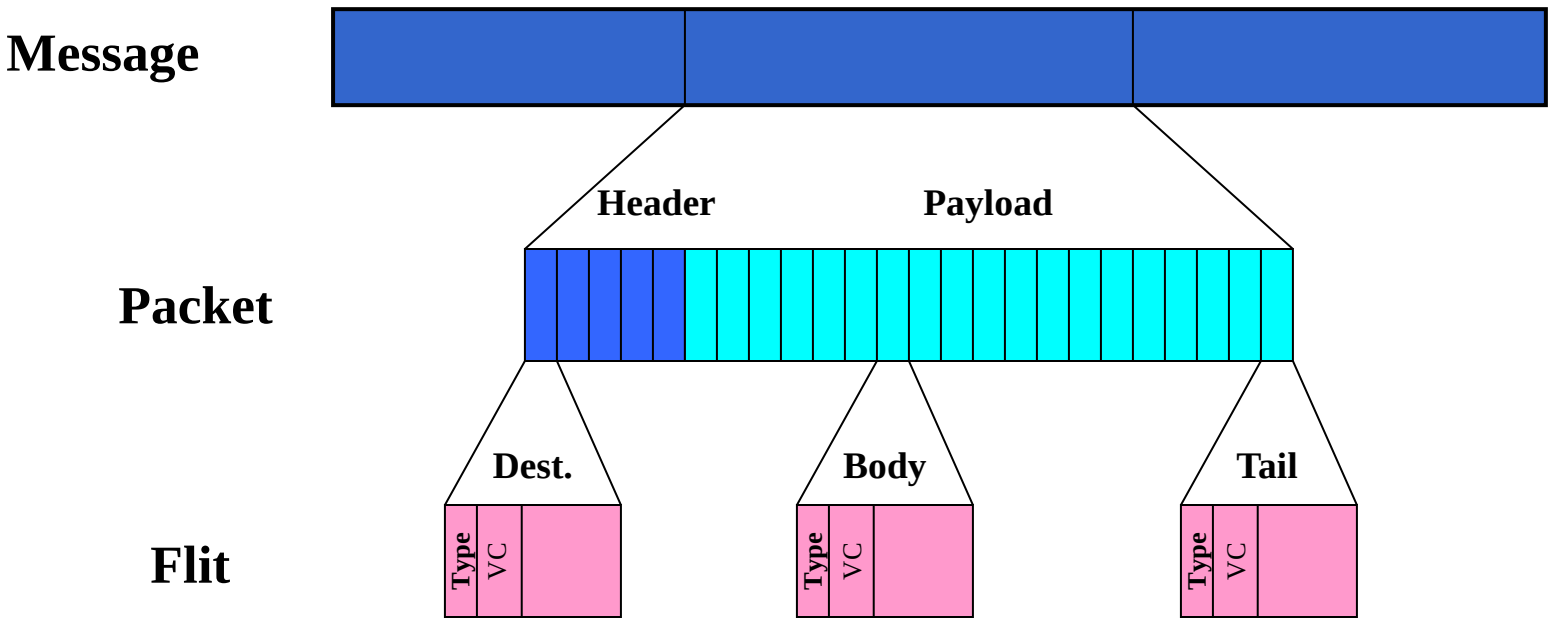


- freedom from deadlocks can be ensured by allocating additional hardware resources or imposing restrictions on the routing
- Usually dependency graph of the shared network resources is built and analyzed either statically or dynamically

Message Abstraction

Packet: An element of information that a processing element (PE) sends to another PE. A packet may consist of a variable number of flits."

Flit: The elementary unit of information exchanged in the communication network in a clock cycle.



Definitions and Terminology

Switch: The component of the network that is in charge of flit routing.

Flit Latency: The time needed for a FLIT to reach its target PE from its source PE.

Packet Latency: The time needed for a PACKET to reach its target PE from its source PE.

Packet Spread: The time from the reception of the first flit of a packet to the reception of the last one.

Packet Switching

Packets are transmitted from source and make their way independently to destination.

Possibly along different routes and with different delays

- zero start-up time, followed by a variable delay due to contention in routers along the path
- QoS guarantees are harder to make in packet switching as compared to circuit switching.

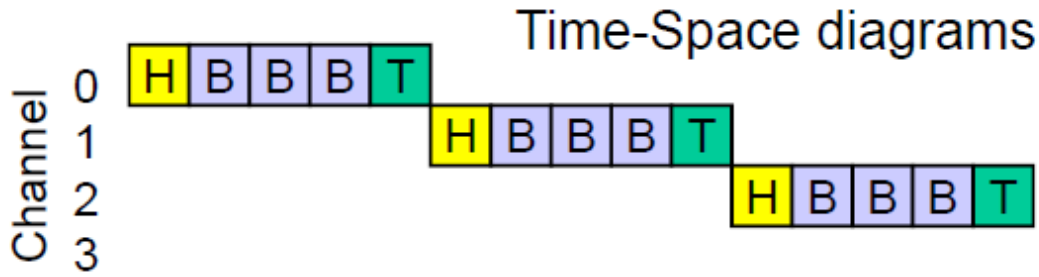
Three types of main packet switching schemes

- **SAF (Store and Forward) Switching**
- **Virtual Cut Through Switching**
- **Wormhole Switching**

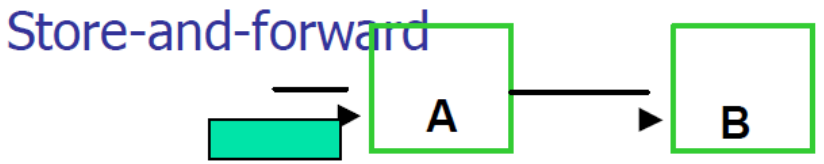
Store and Forward (SAF) Switching

ENTIRE packet gets buffered by a router before moving to next router/node. Excessive buffer requirements

- Each router requires buffering for an entire packet
- Packet size affects the cost, latency and buffering requirements.
- Stalling happens at the nodes and the link between them.

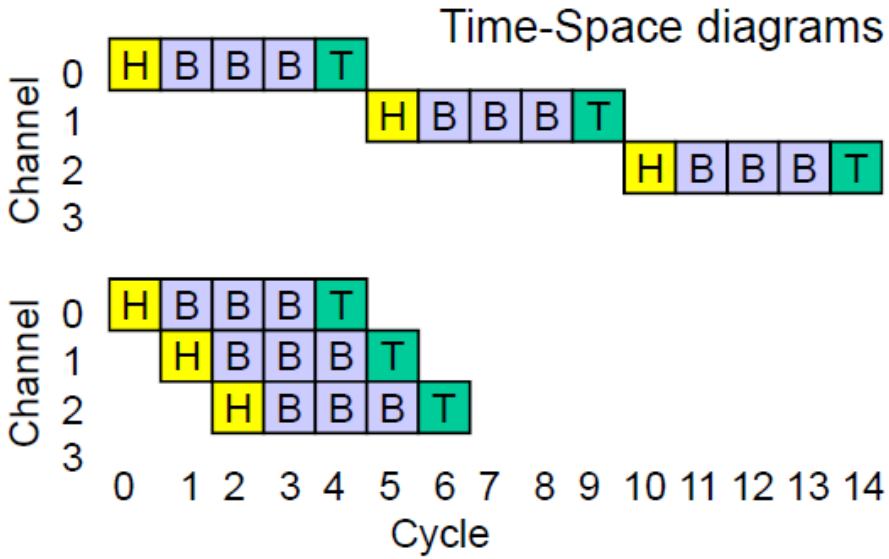


- After A finishes receiving entire packet, A sends B a request to receive entire packet
- B ack A
- A sends packet to B



Virtual Cut-through Switching

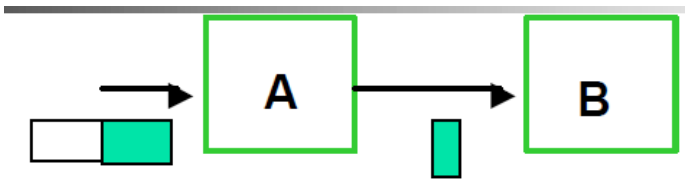
Similar to SAF. Divides packets into flits.
Starts forwarding flits once header is received (& buffer space available).



- Reduces router latency over SAF switching.
- Same buffering requirements as SAF switching
- Router cost depends on header and packet size.
- Stalling at the local nodes level.

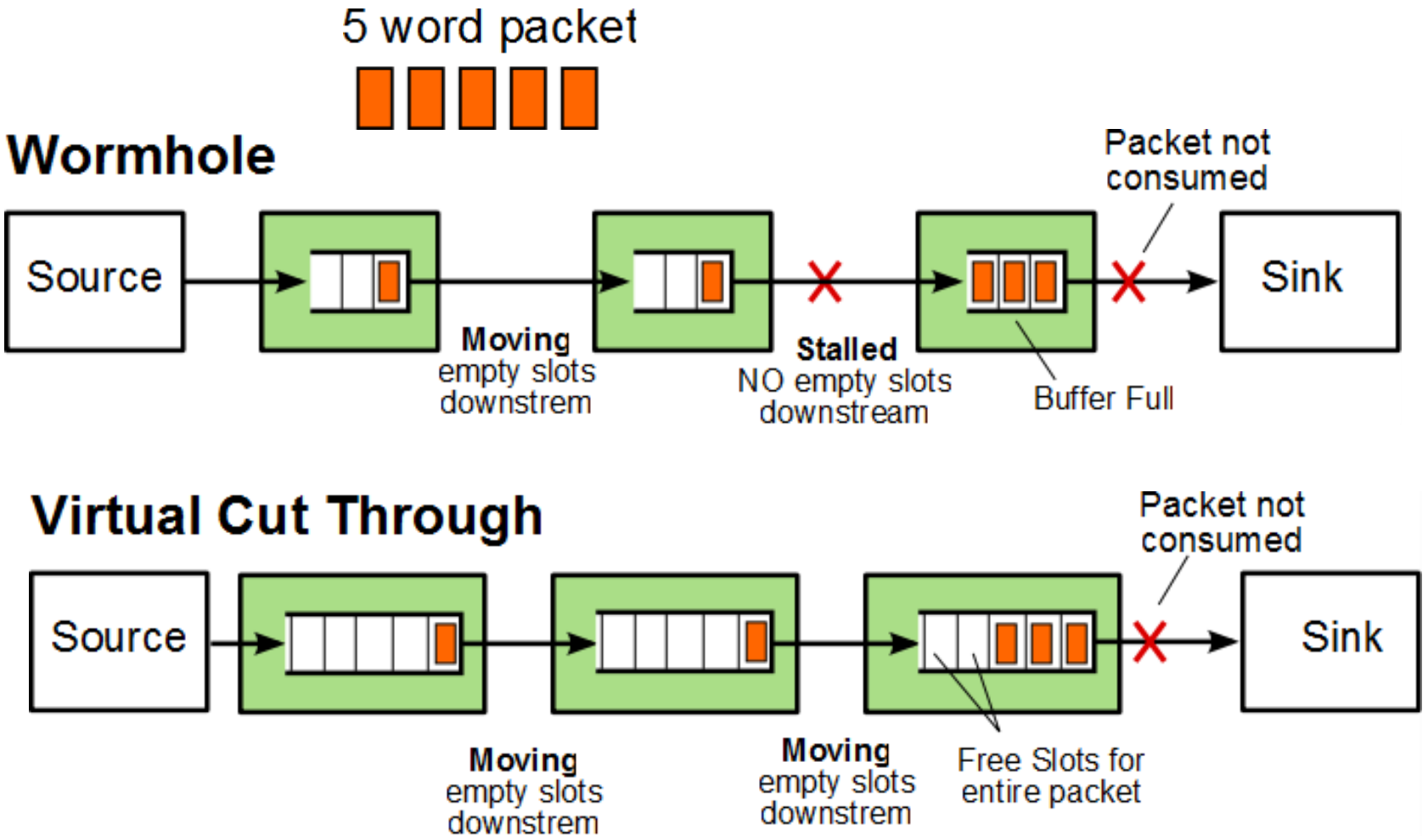
Wormhole Switching

- Packets divided into flits. A flit is forwarded to the next router if space exists for a flit.
- Parts of the packet is distributed among two or more routers.
- Susceptible to deadlocks - usage dependencies among links.
  - Send packet **flits** across on-chip network like a “worm”
  - **A** receives a flit of the packet
  - **A** inquires with **B** if it is ready to receive a flit
  - **B** sends ack to **A** when ready
  - **A** sends flit to **B**



Delay incurred for the header flit

VCT and Wormhole Switching



Wormhole *vs.* VCT and SAF

- Wormhole switching has advantages over SAF and VCT:
  - Lower latency
  - More efficient buffer utilization
- Limitations:
  - Suffers from head-of-line blocking
    - ✓ If head flit of one packet does not move due to contention, it can block other packet worms from proceeding as well

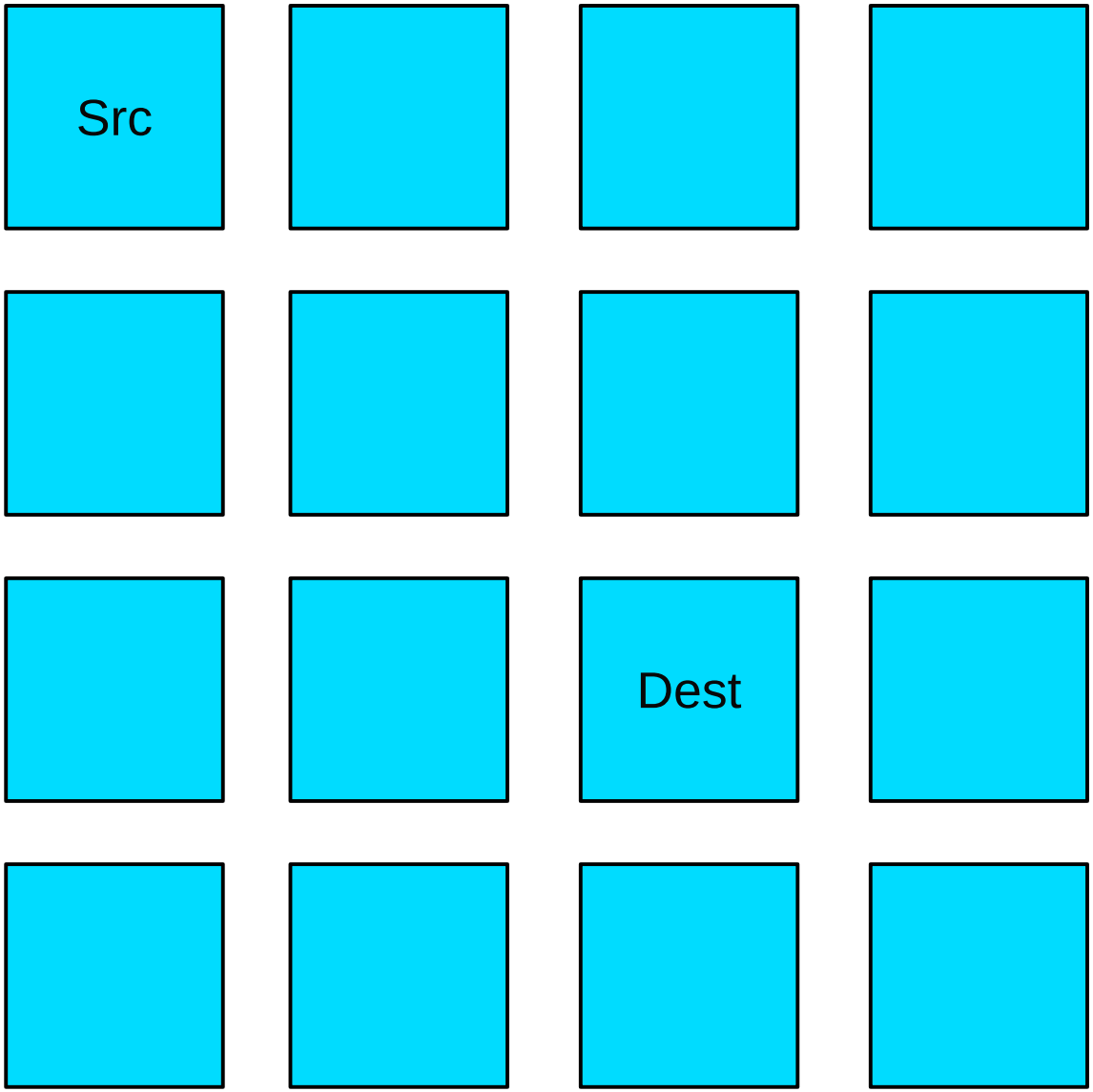
Relevant Parameters of Routing

- Minimum latency is of paramount importance in NoCs (for inter-process communication).
- Ideally: One clock cycle latency per switch/router (flit enters at time t and exits at t+1)
- Maximum switch clock frequency
  (technology + routing logic limits)
- Deadlock free
- No flits are ever lost; once a flit is injected in the NoC, it must reach to its destination
  may be after a long time.

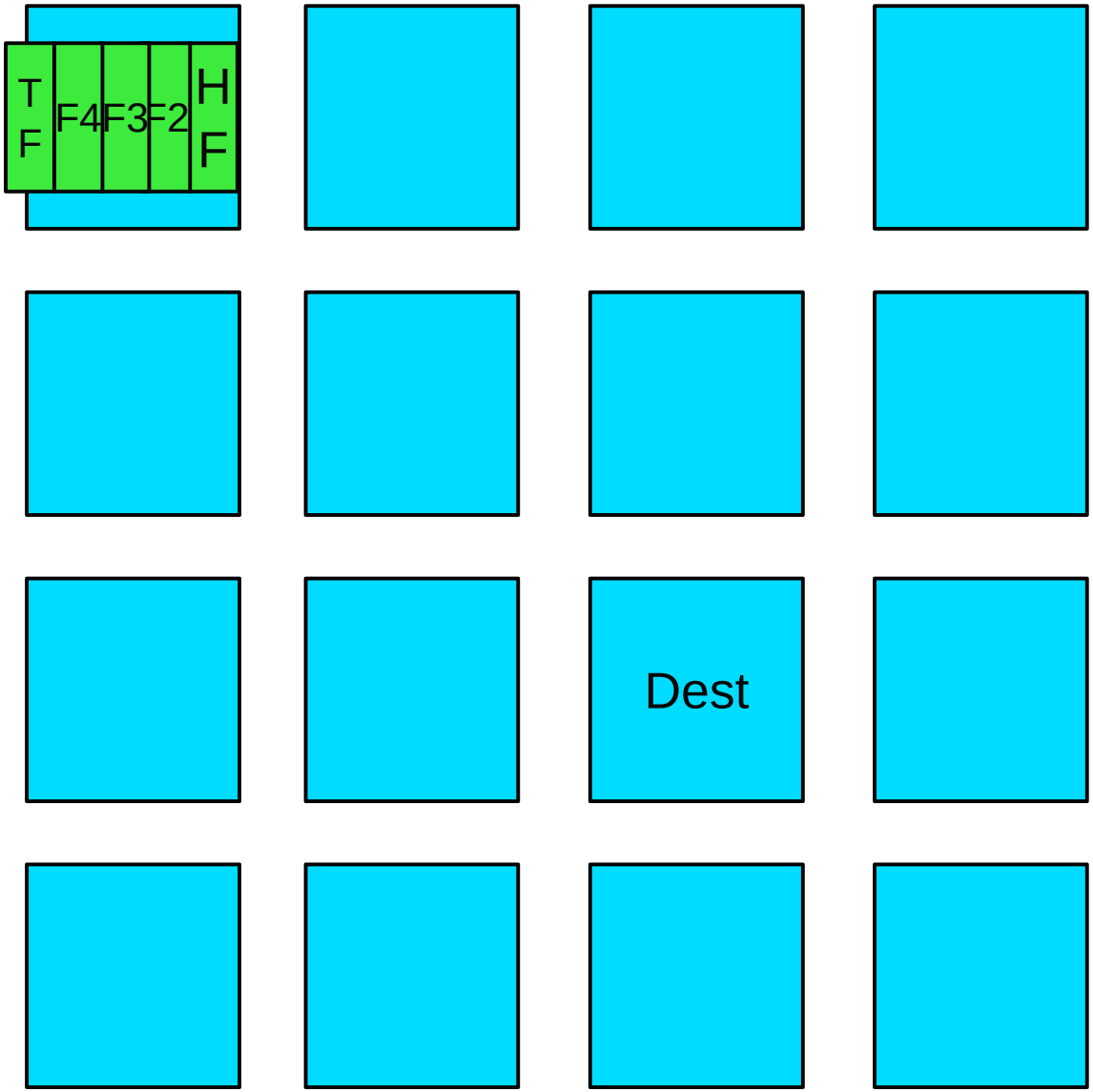
Wormhole Routing

- The header flit "digs" the path in the network to the destination
  - And holds the path, essentially creating a "worm" for the other flits to follow with no delay
- Body flits follow the head flit, and tail follows body
- Pipelined
- If head flit blocked, rest of the packet stops.

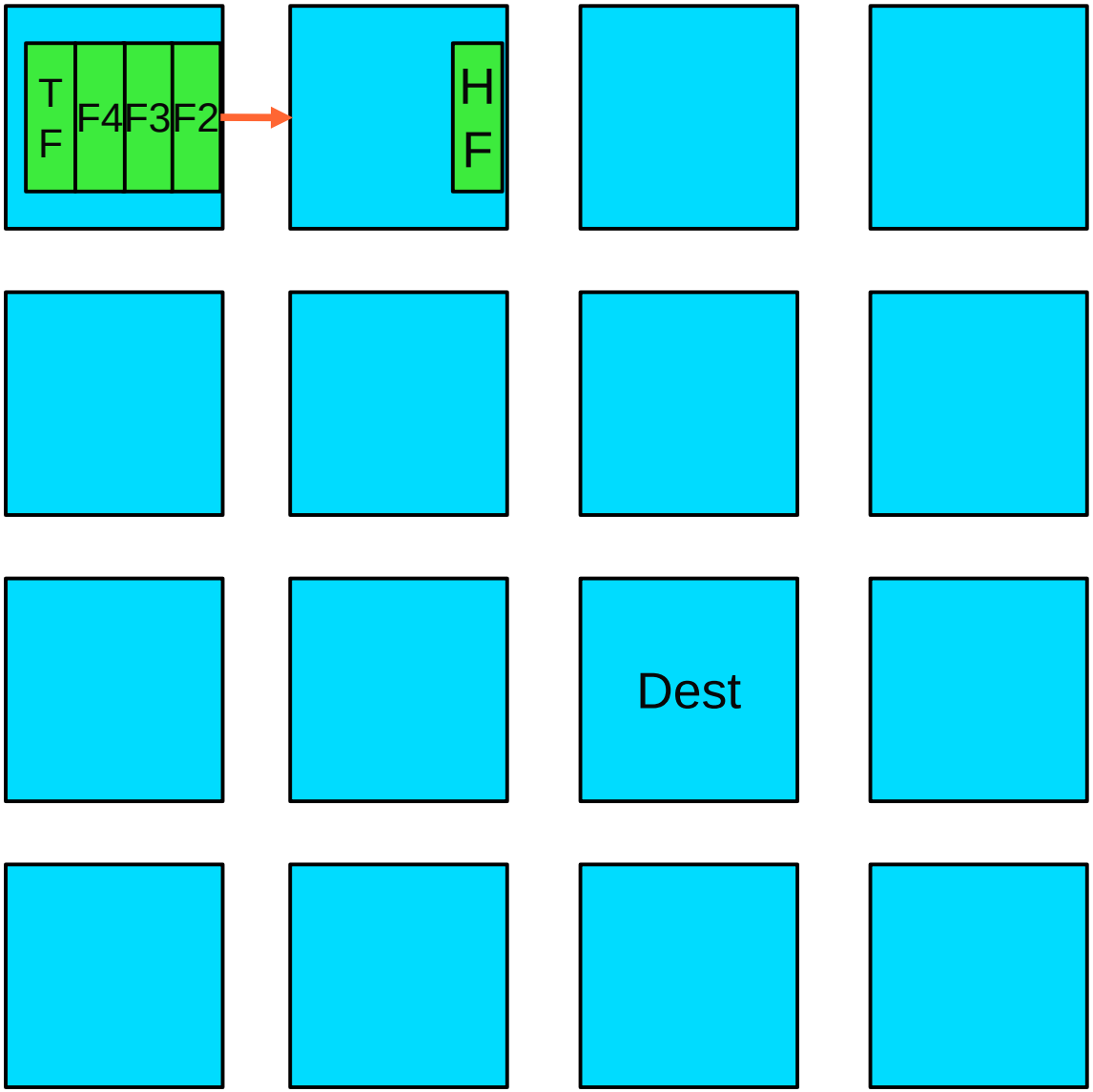
Wormhole



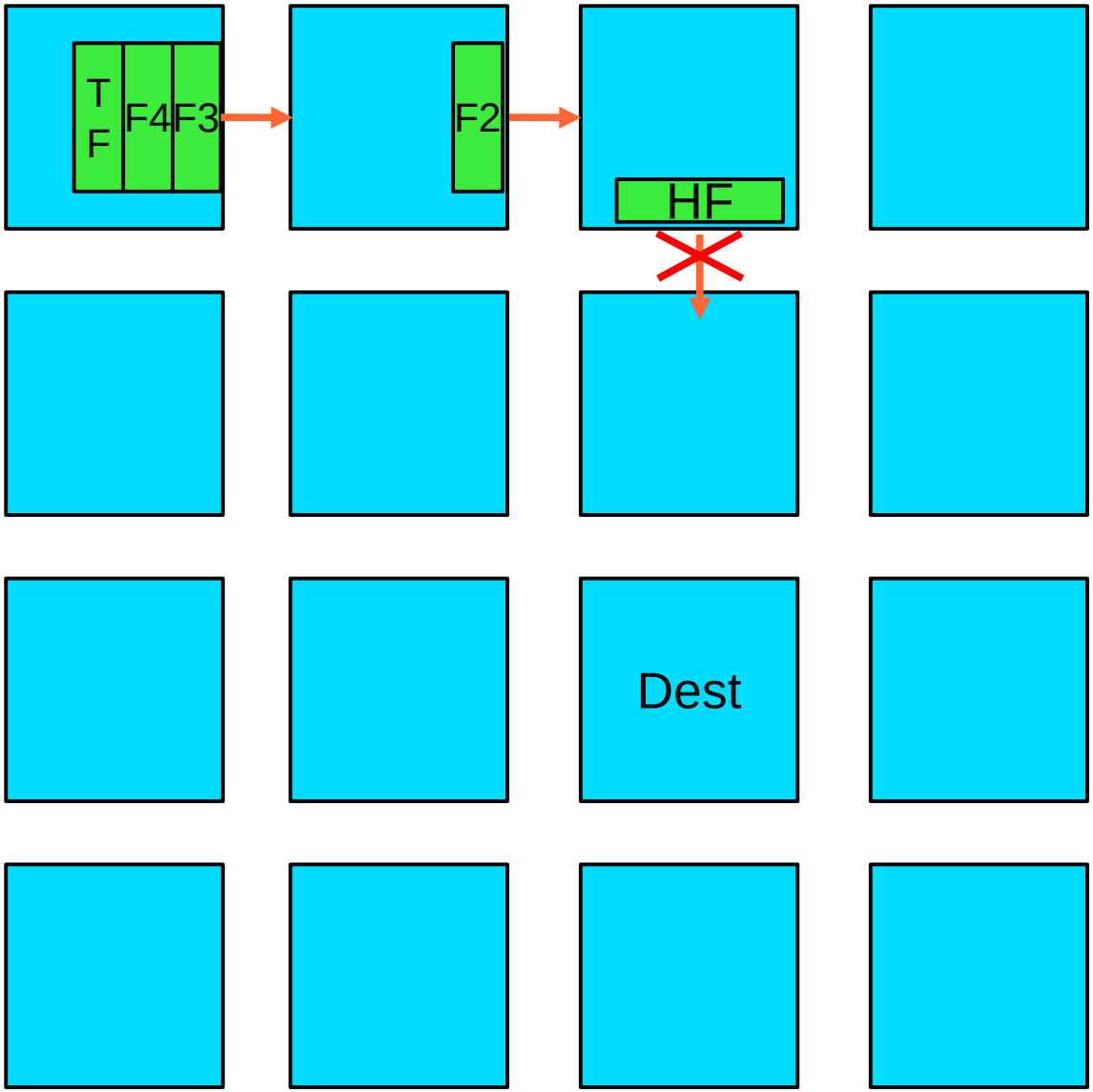
Wormhole



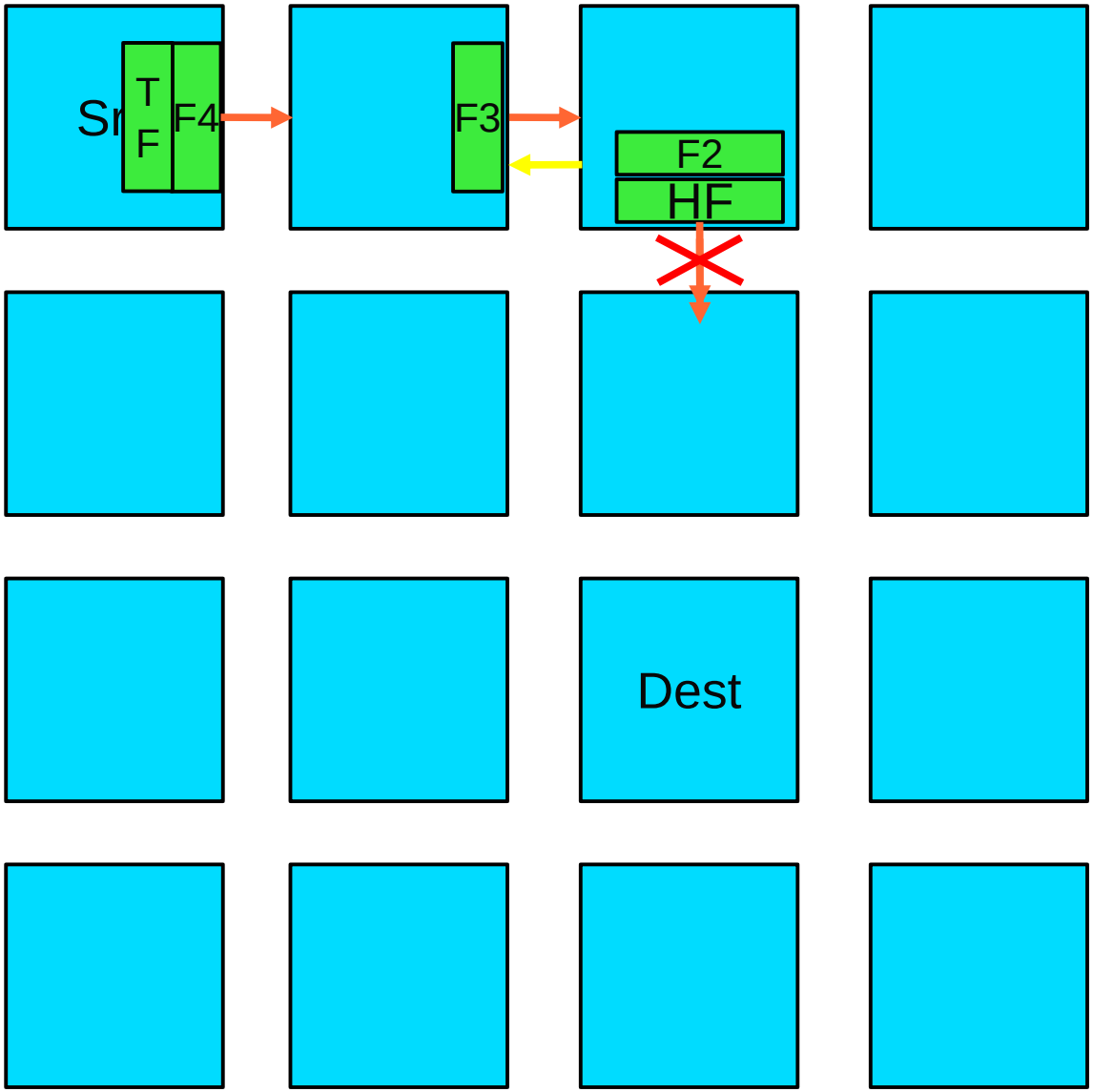
Wormhole



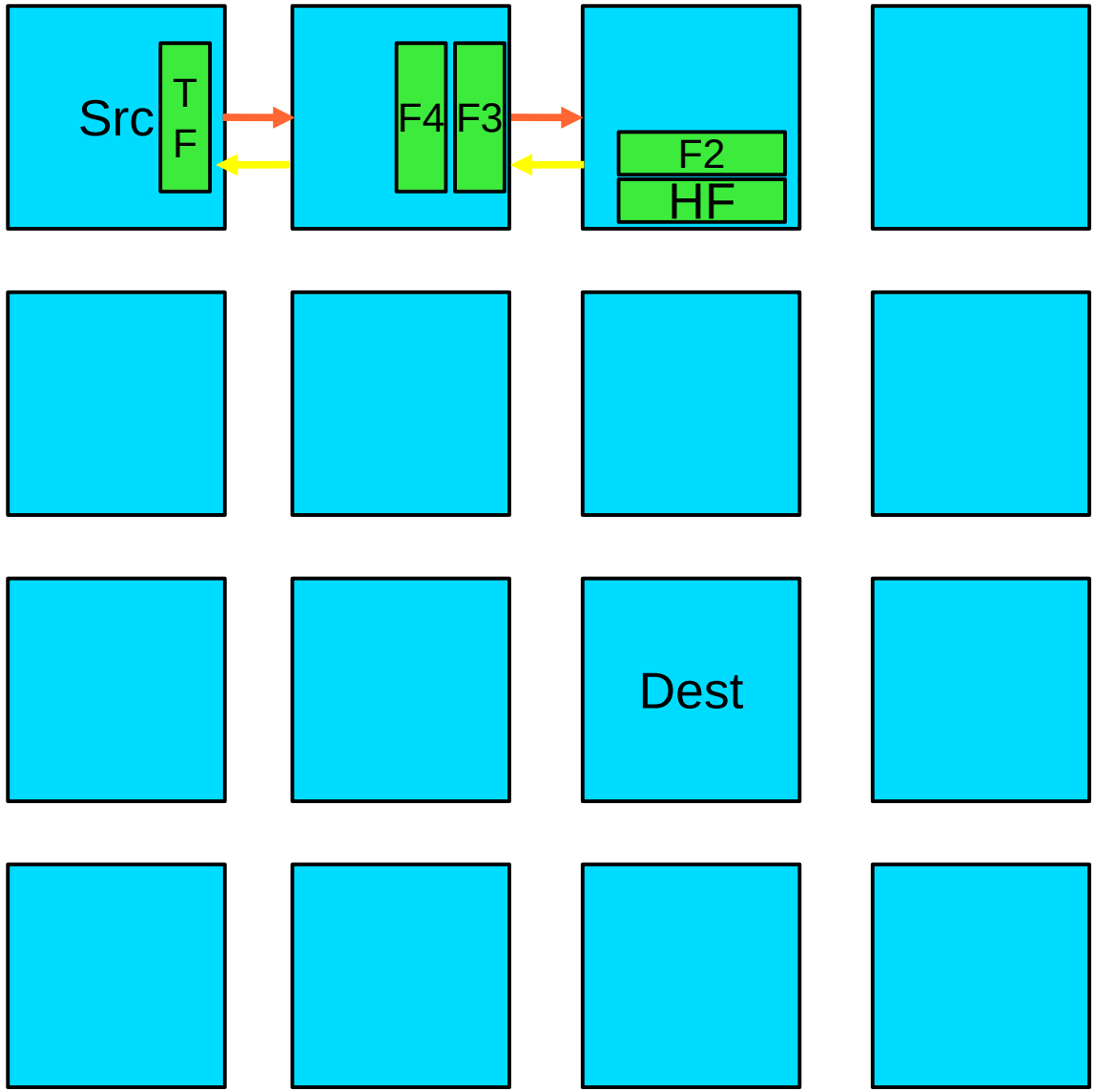
Wormhole Routing



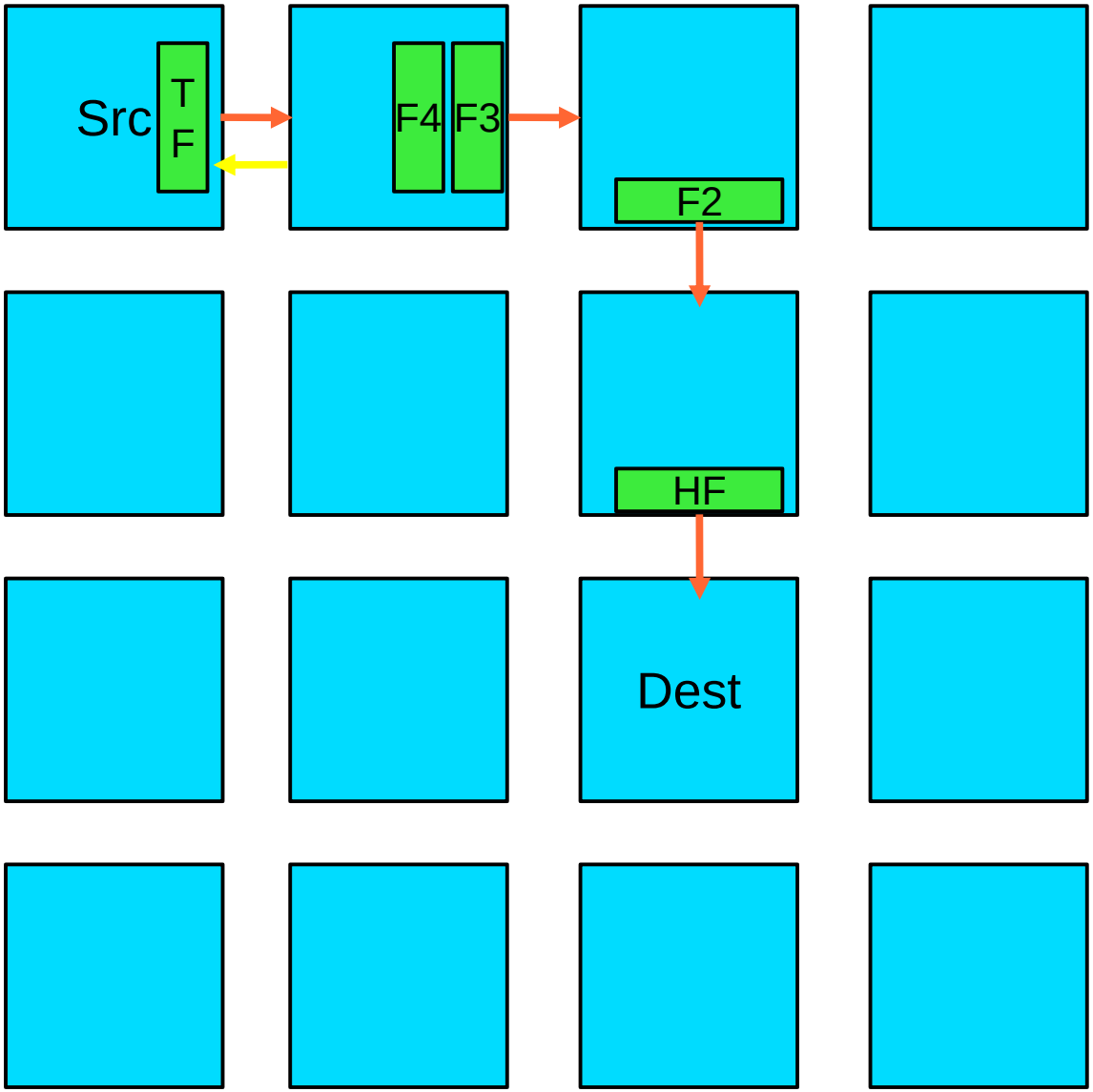
Wormhole



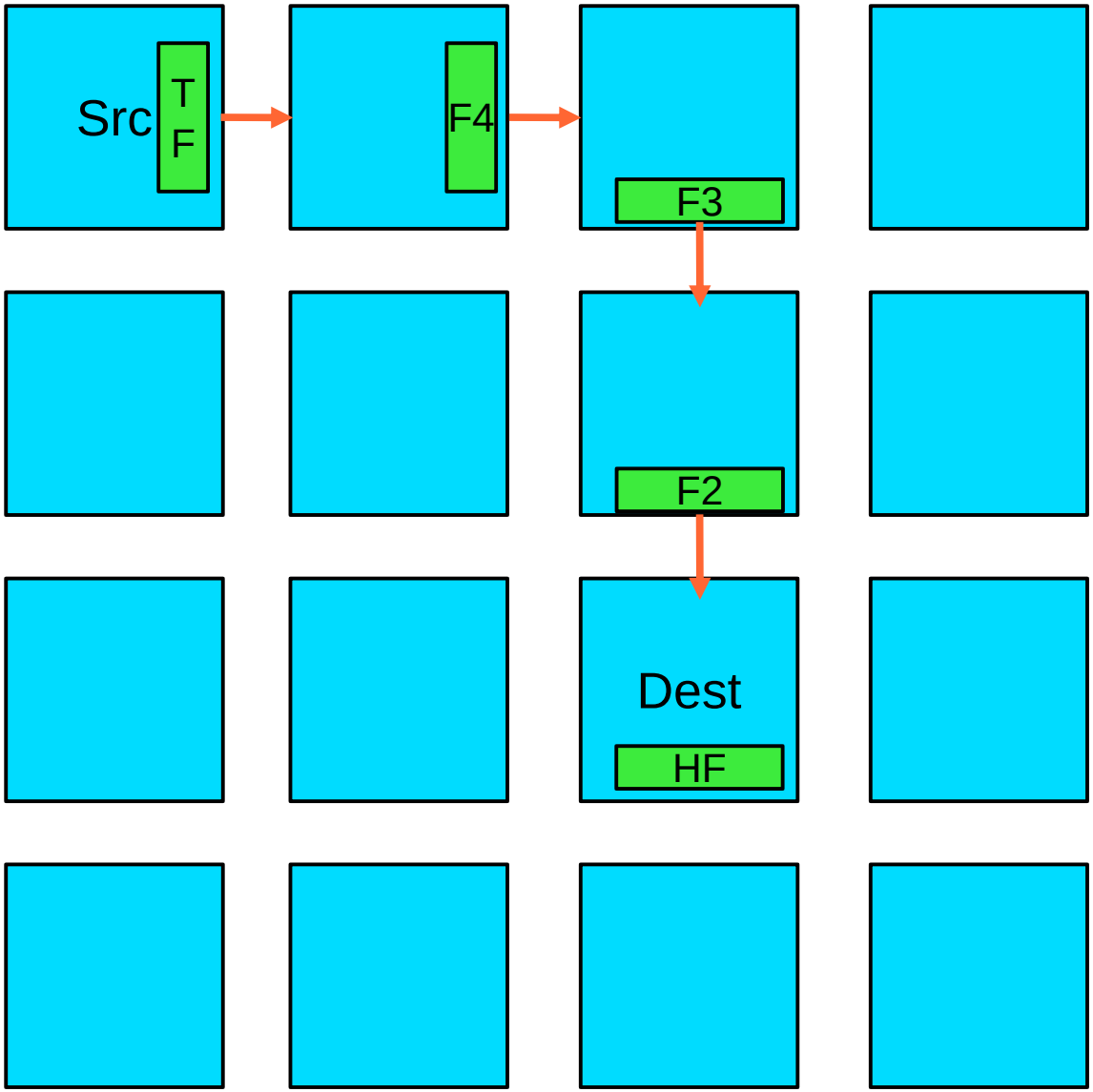
Wormhole



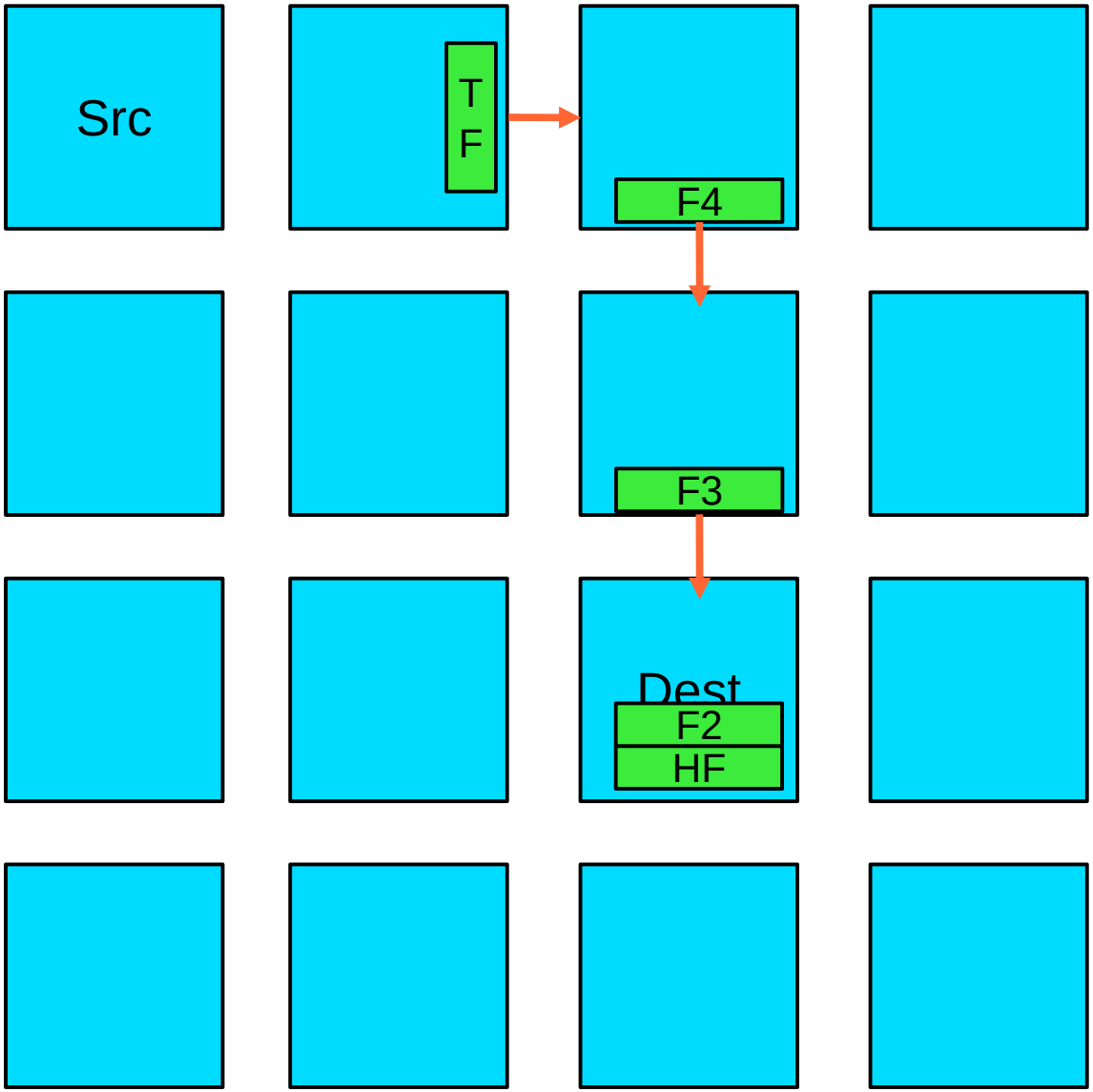
Wormhole



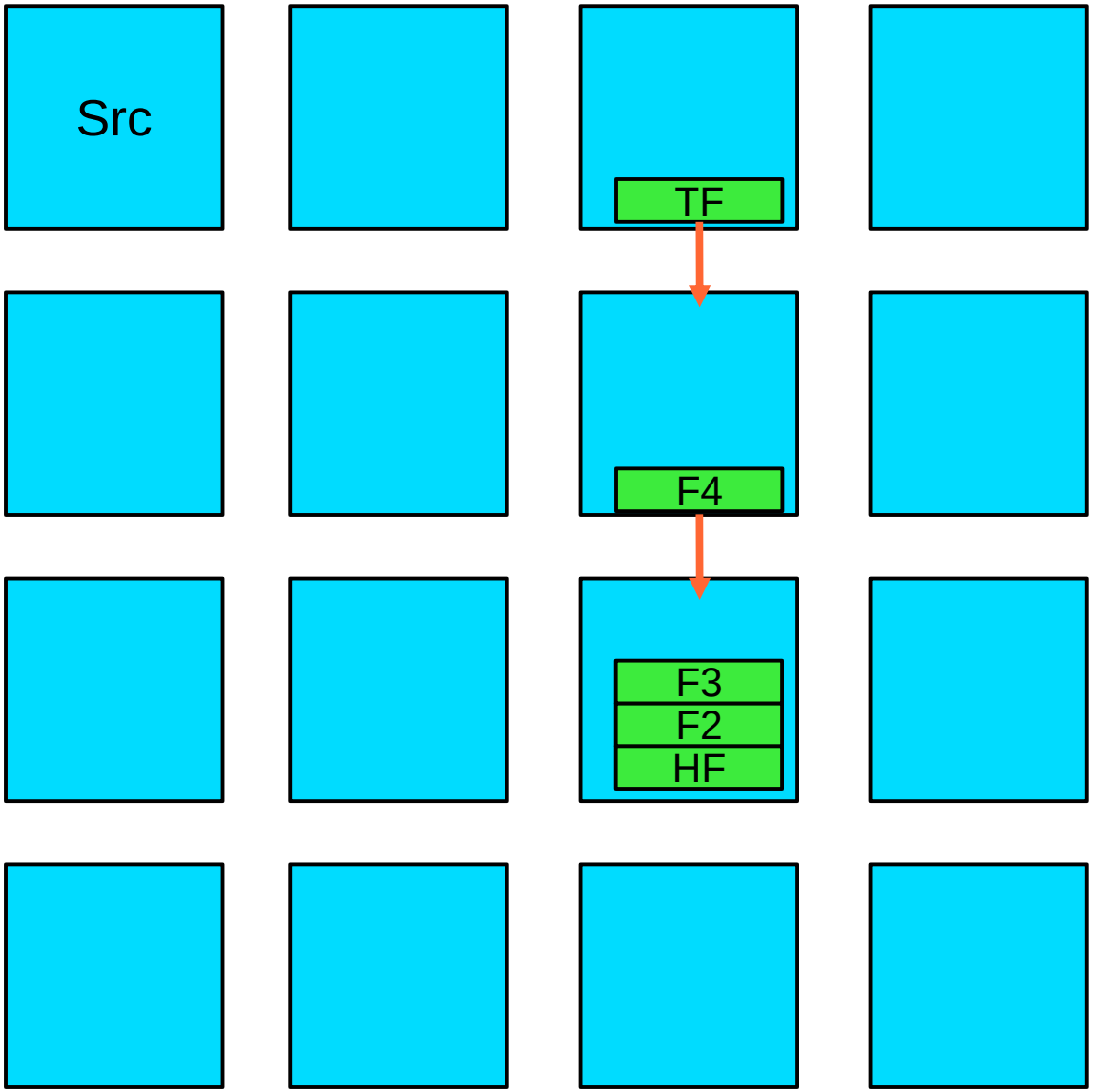
Wormhole



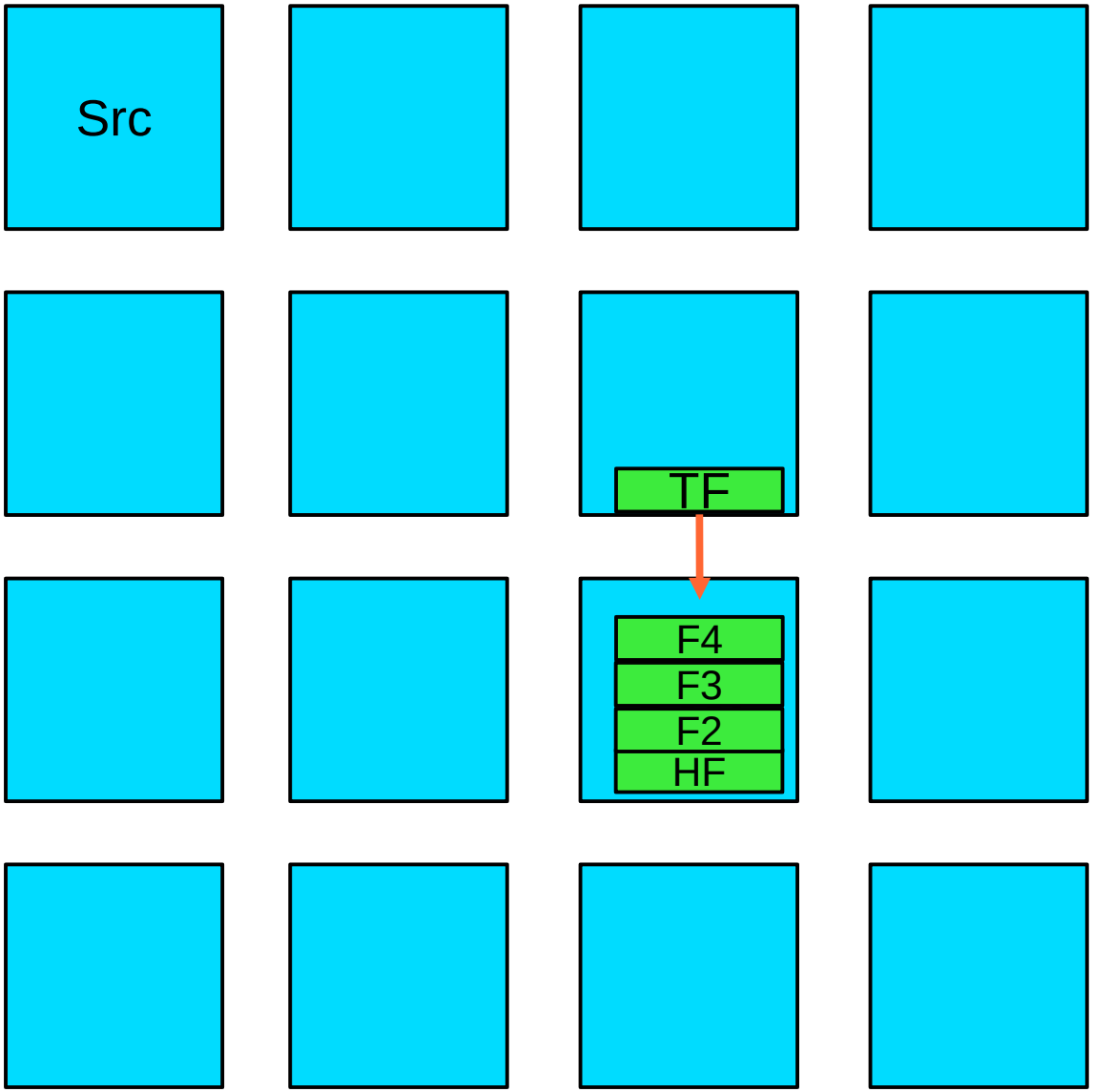
Wormhole



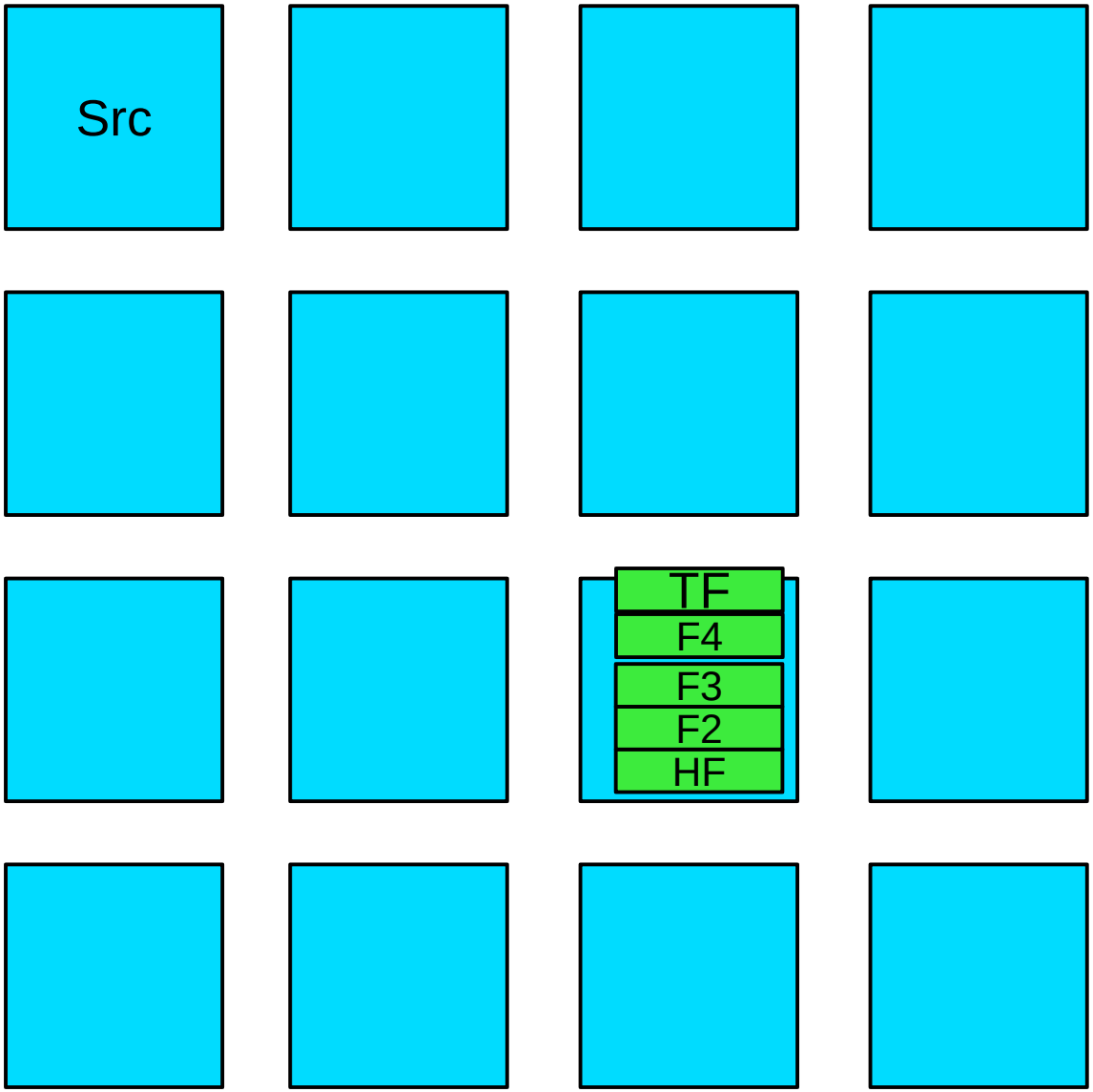
Wormhole



Wormhole



Wormhole



Deflection Routing

Hot Potato – Deadlock Free Routing

Every flit can be routed to different directions
(no packet notion at the switch level)

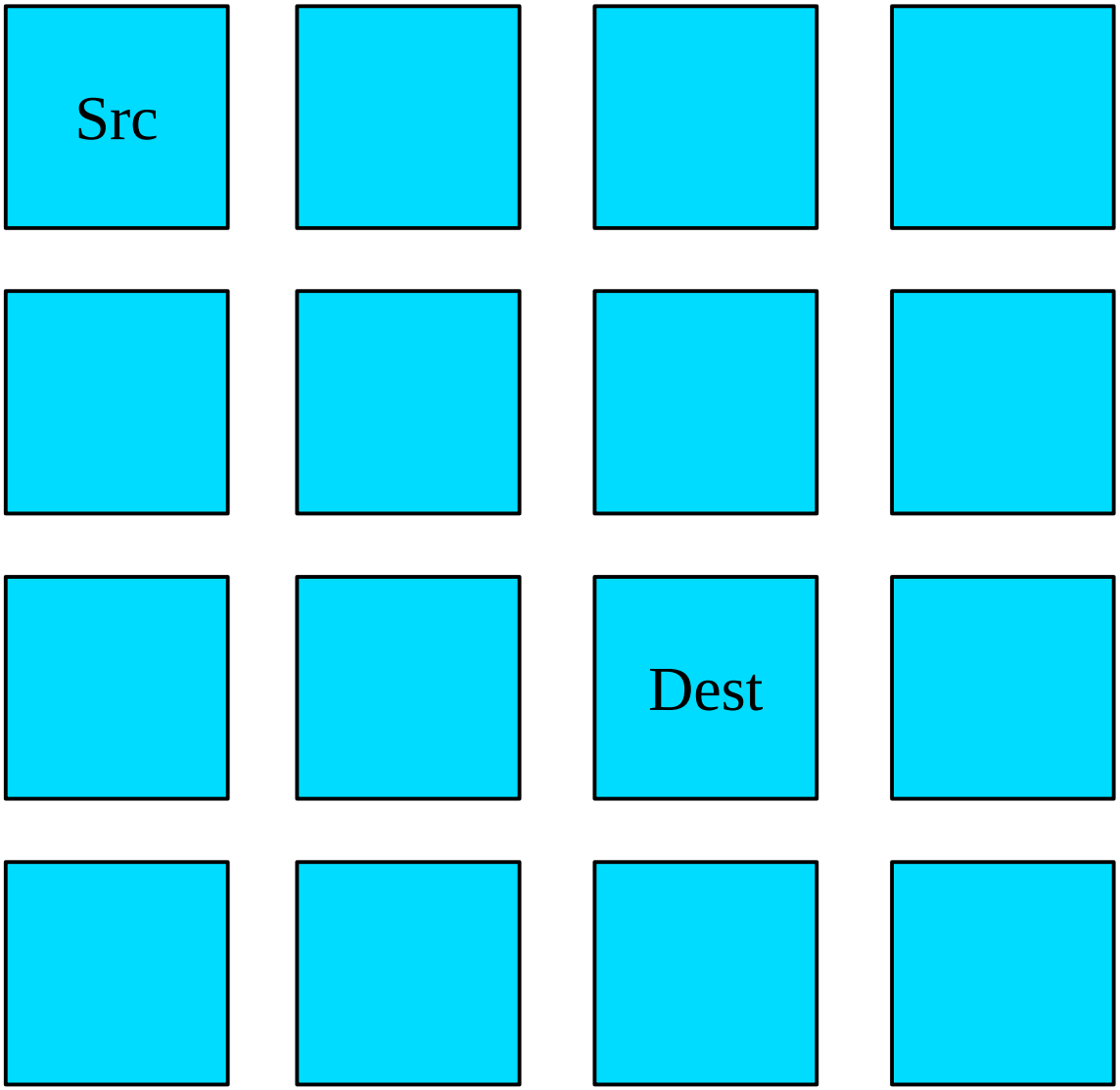
- ***If the optimal direction is blocked, the flit is "deflected" to another direction***
  - **Switch latency of 1 clock cycle whatever the level of congestion**
  - **Minimum buffer requirements**

- Packets reordering
- Adaptive routing
- No buffering
- No back pressure
- Works with Torus/Mesh

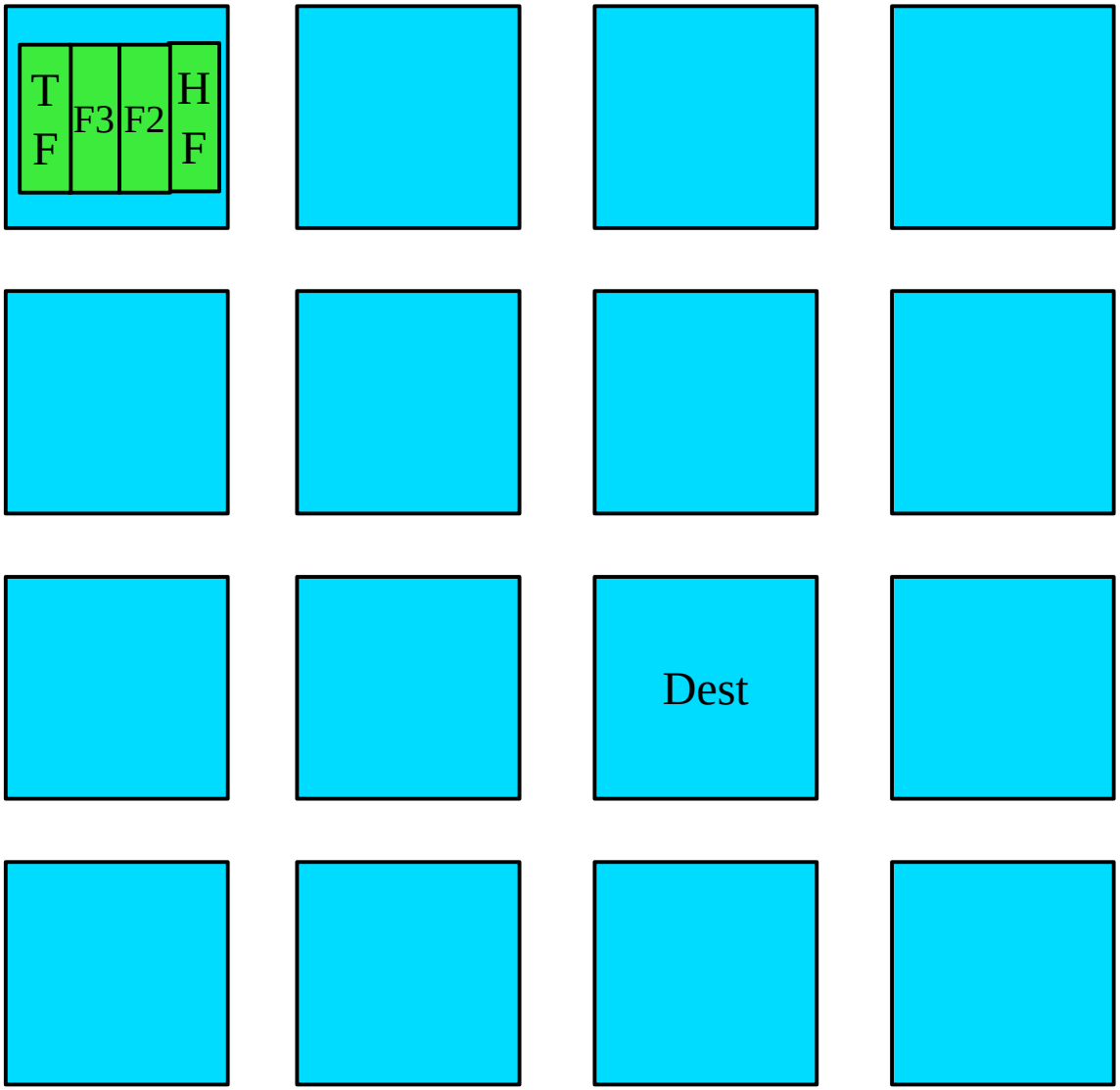
Wormhole Routing

- No packets reordering
- Static routing
- Buffering (≥ 2 flits/port)
- Back pressure
- XY routing needs mesh

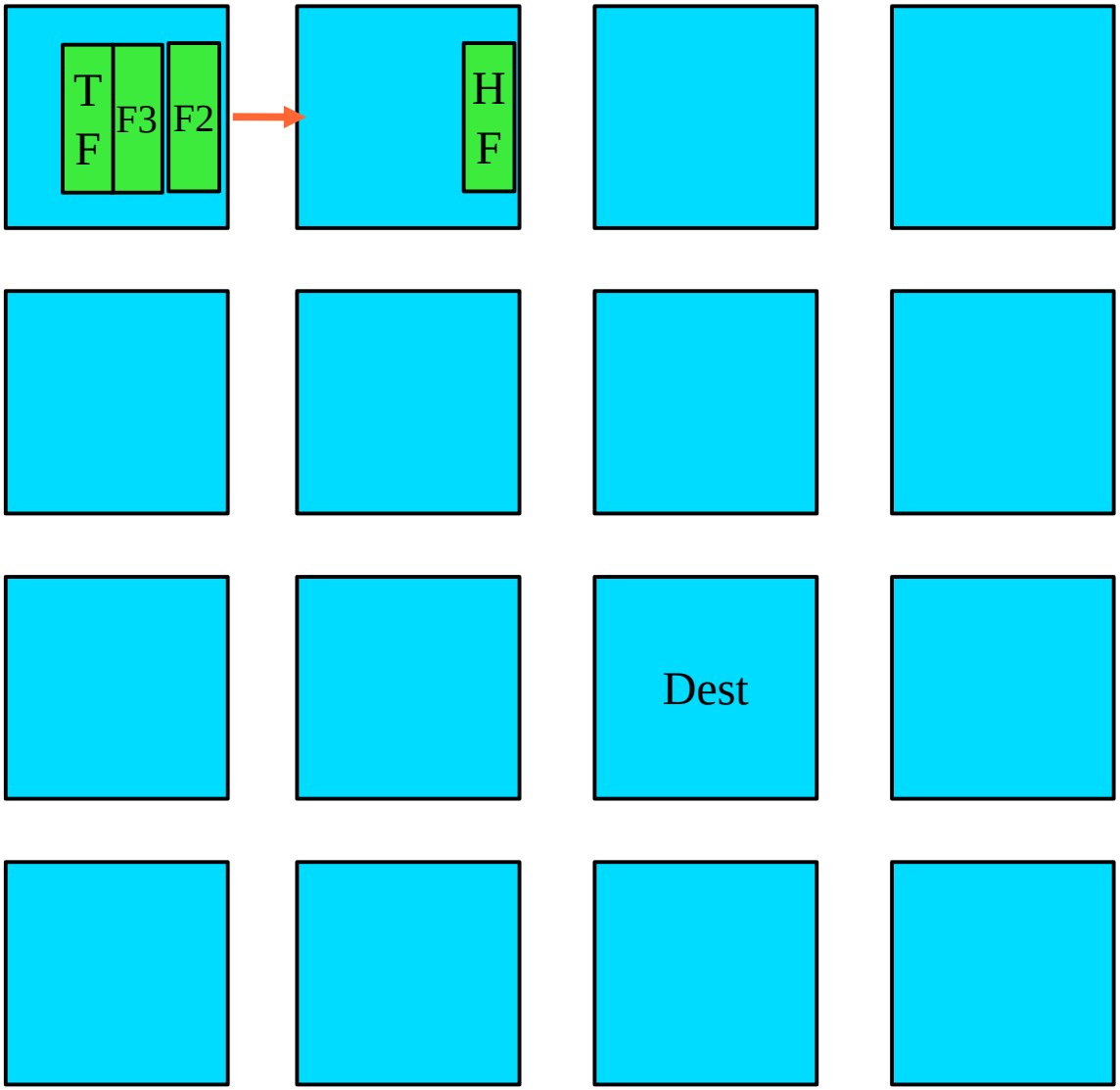
Hot-Potato



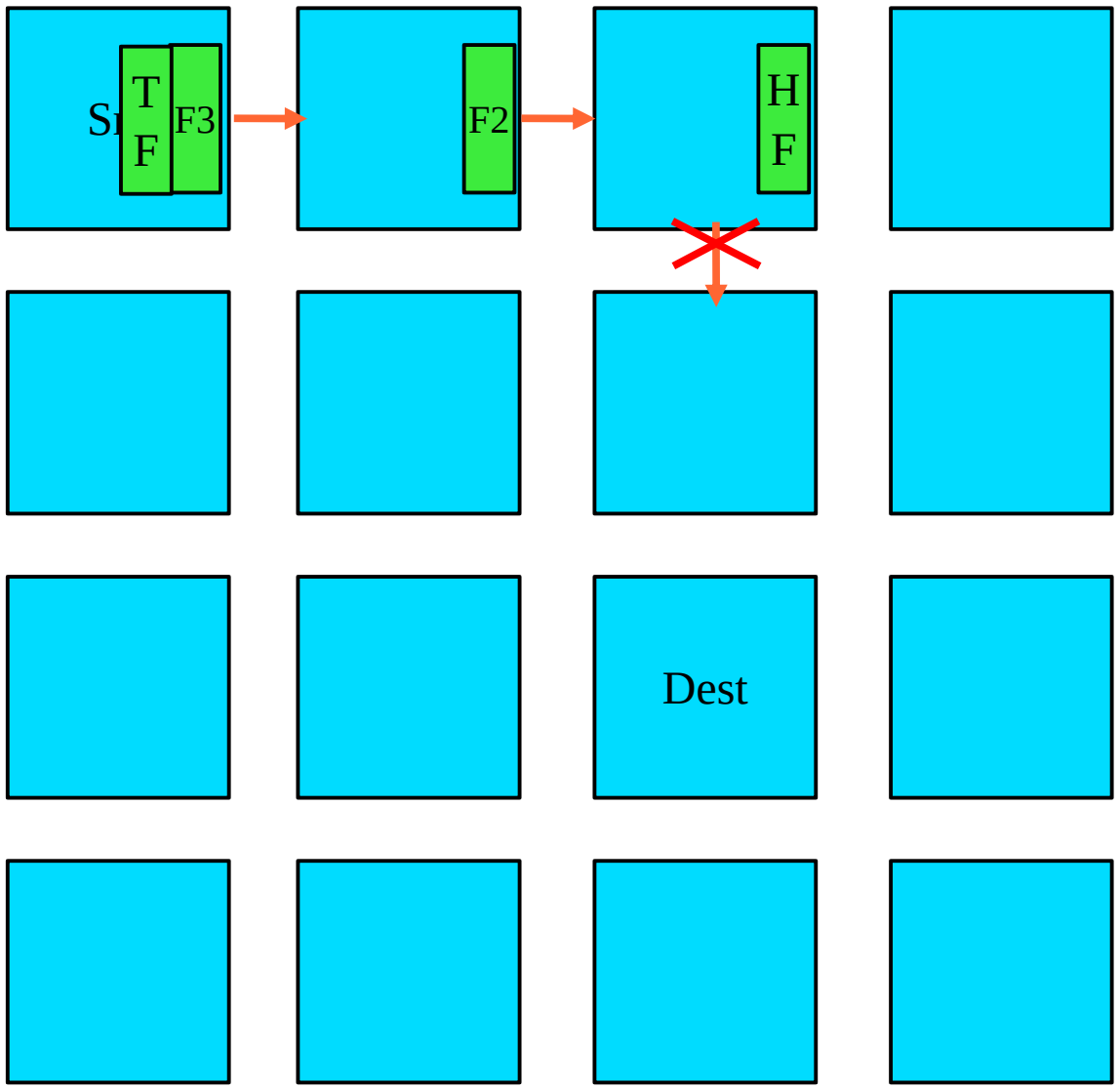
Hot-Potato



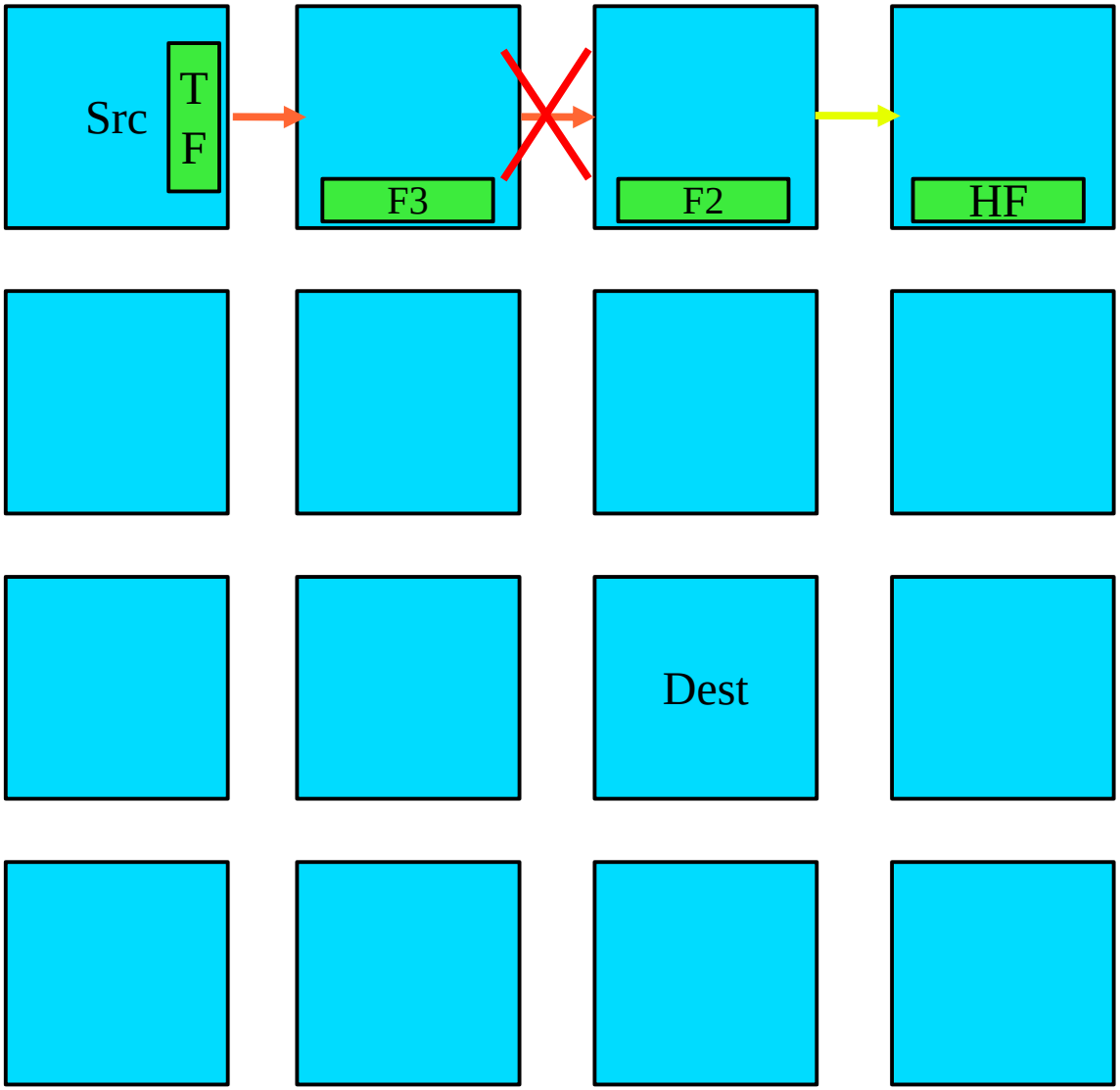
Hot-Potato



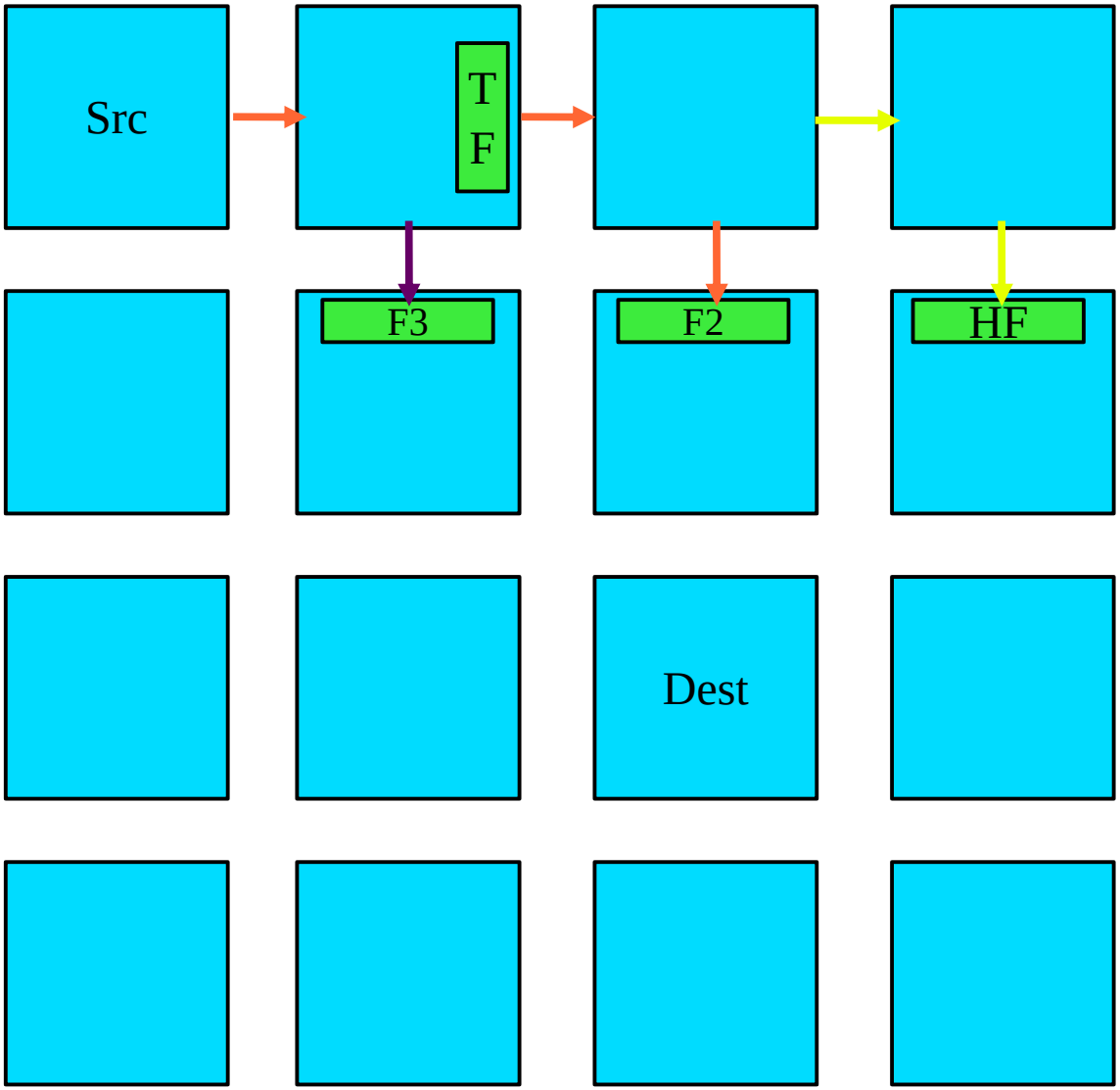
Hot-Potato



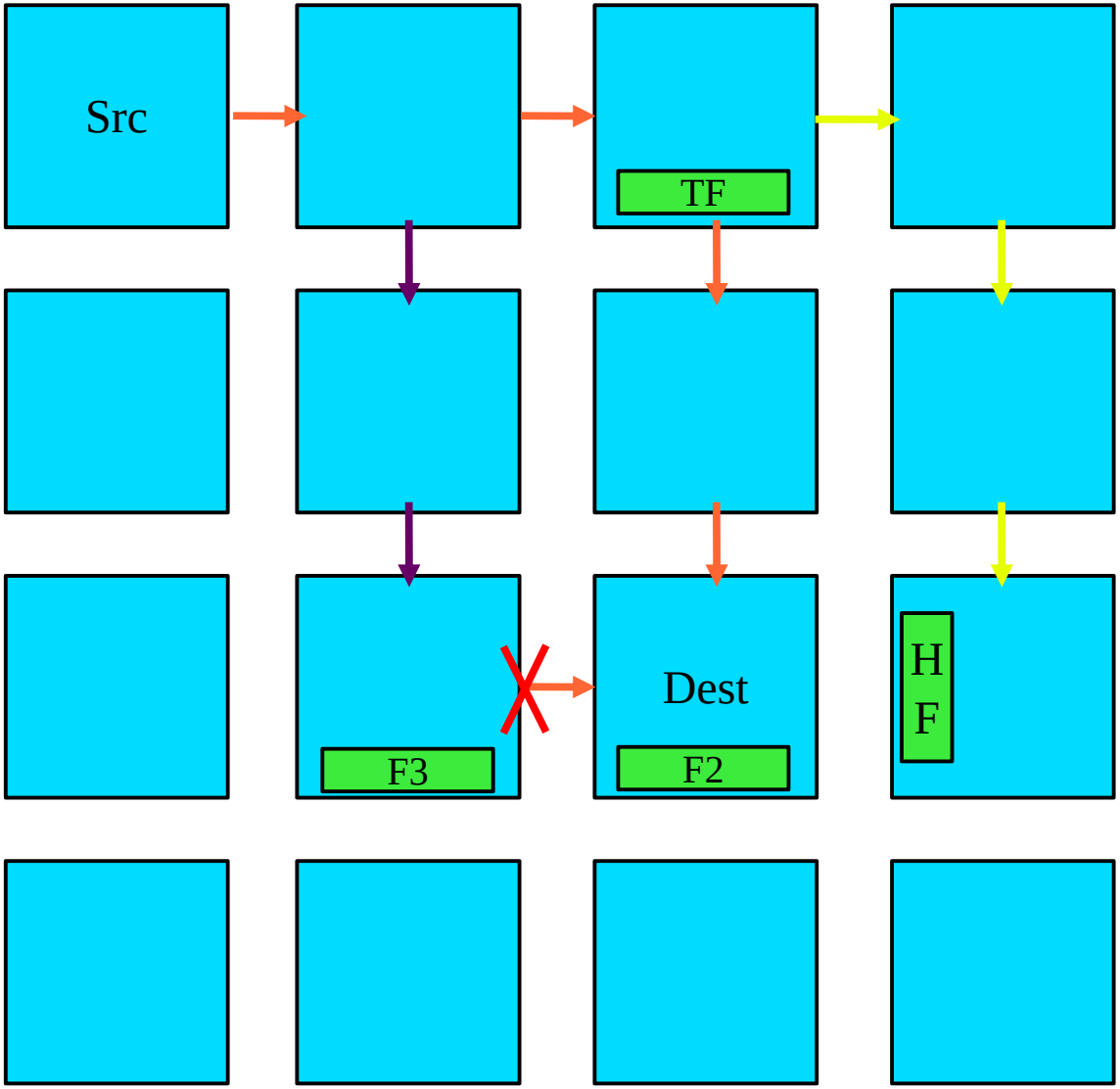
Hot-Potato



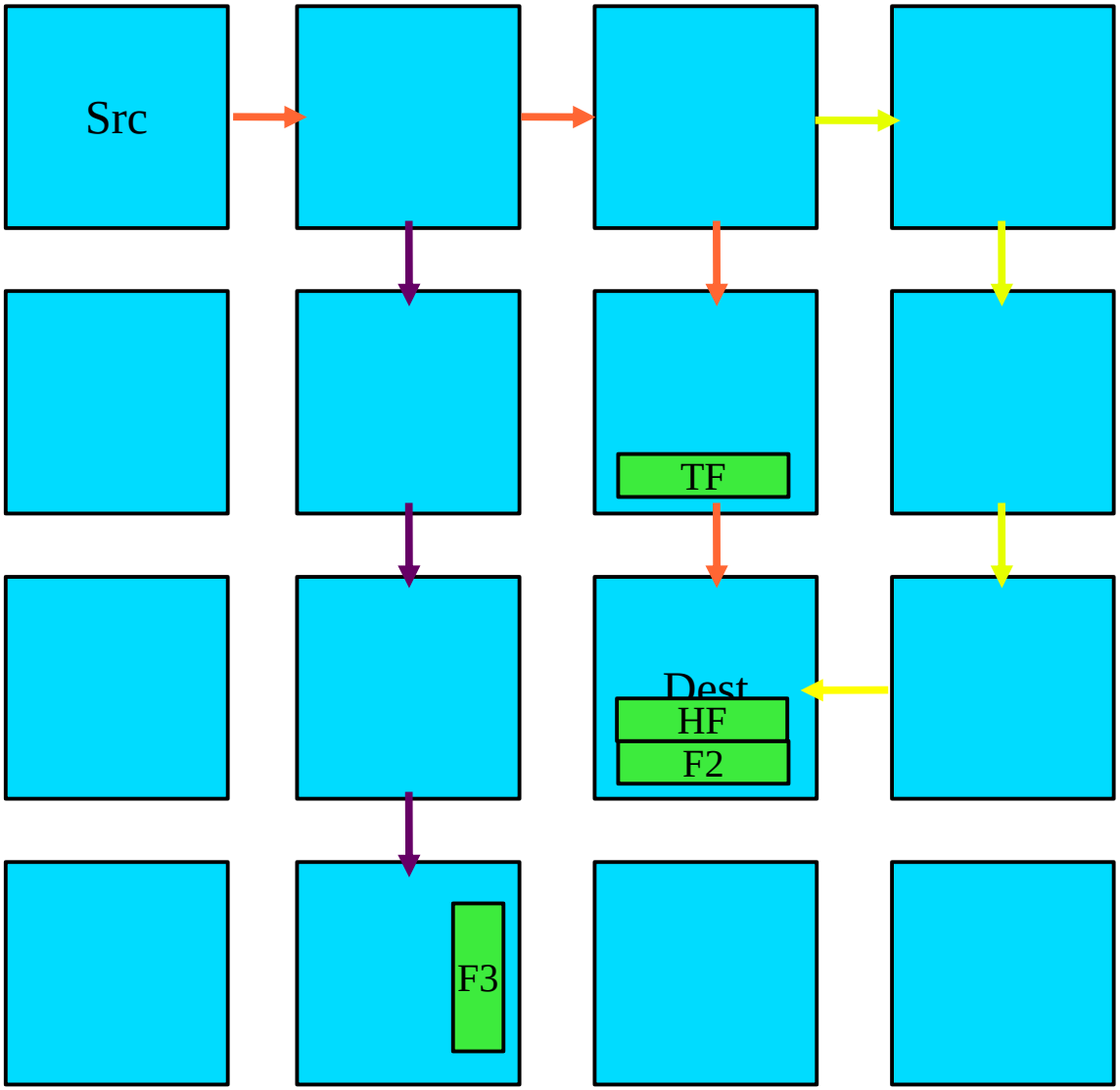
Hot-Potato



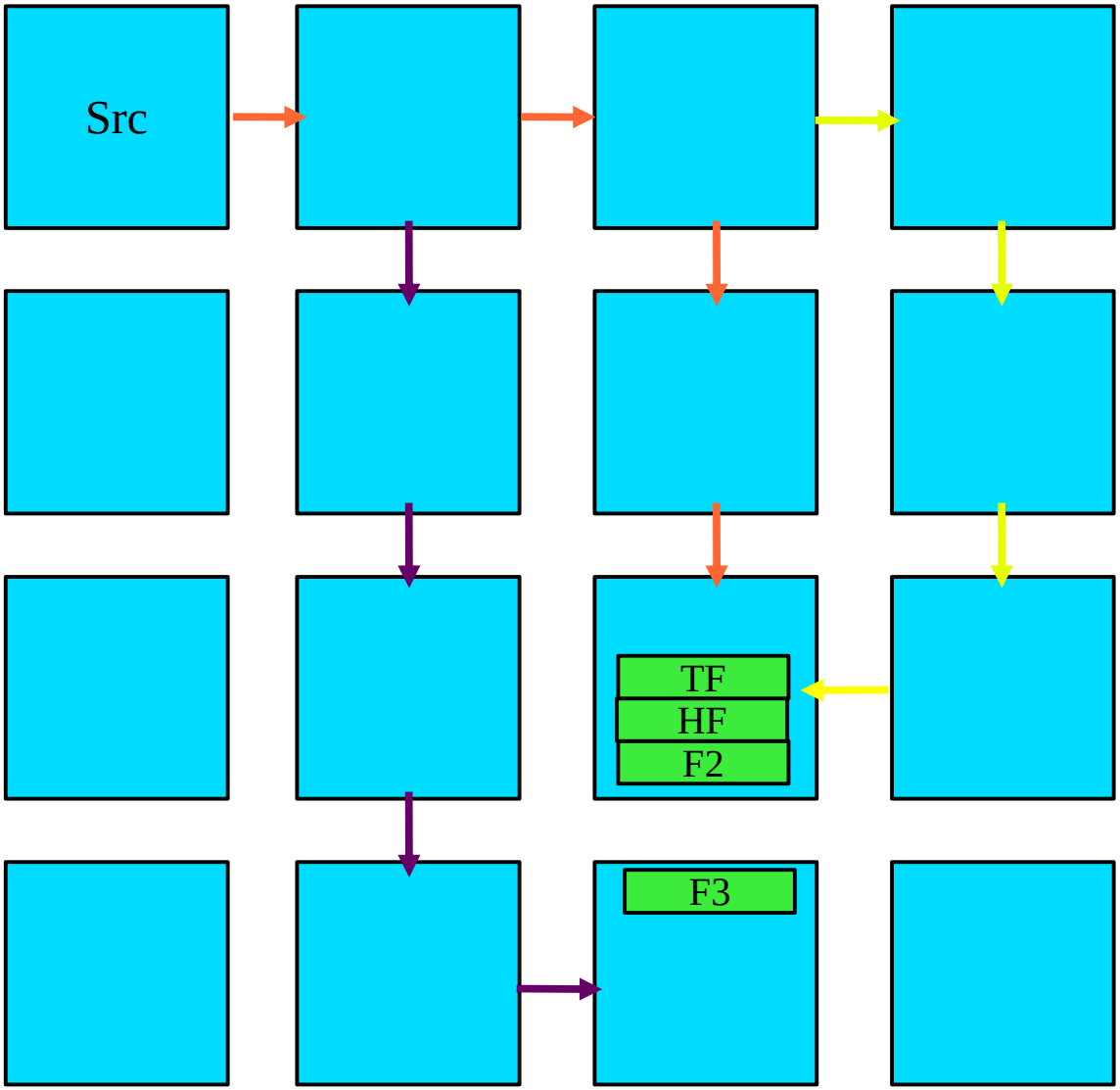
Hot-Potato



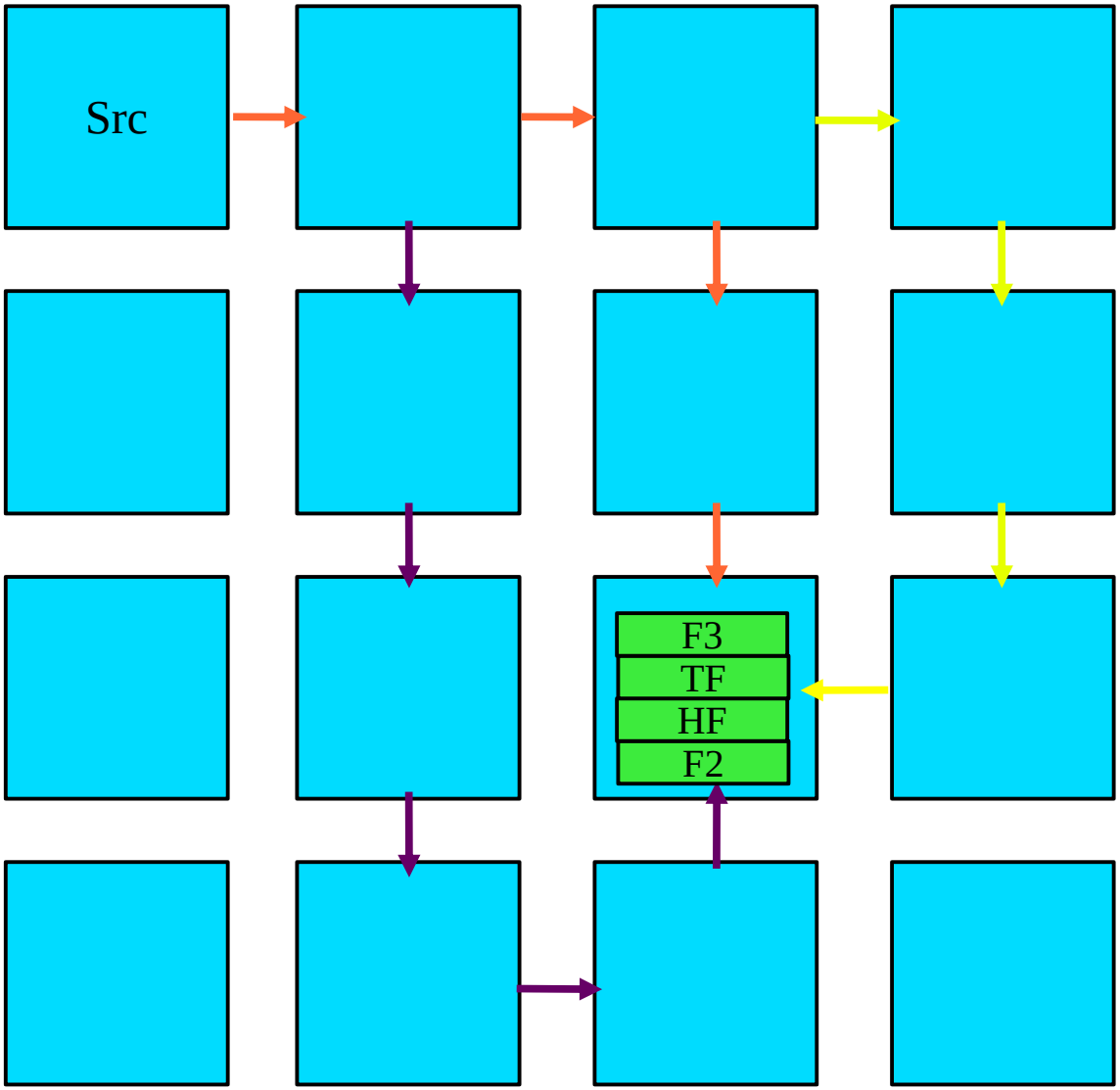
Hot-Potato



Hot-Potato



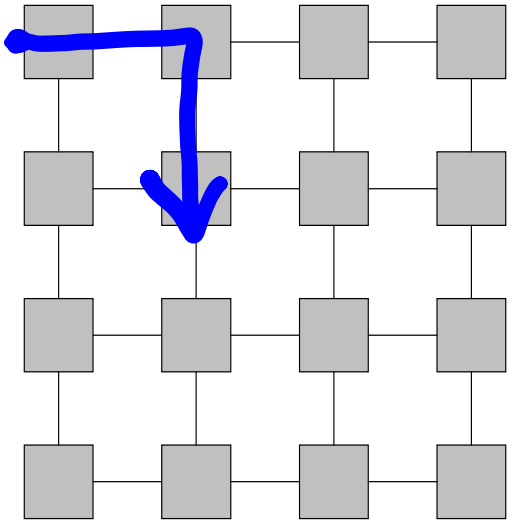
Hot-Potato



Fixed Shortest Path Routing

Suitable for Regular Topologies

e.g. Mesh, Torus, Tree, etc.



X-Y Routing

(first x then y direction)

- **Simple Router**
- **No deadlock scenario**
- **No retransmission**
- **No reordering of messages**
- **Power-efficient**

Flow Control Schemes

- Objective is to allocate network resources for packets to traverse the NoC.
- It can also be viewed as a problem of resolving contention during packet traversal
- Manages congestion
- Provide error recovery mechanism for retransmission in network

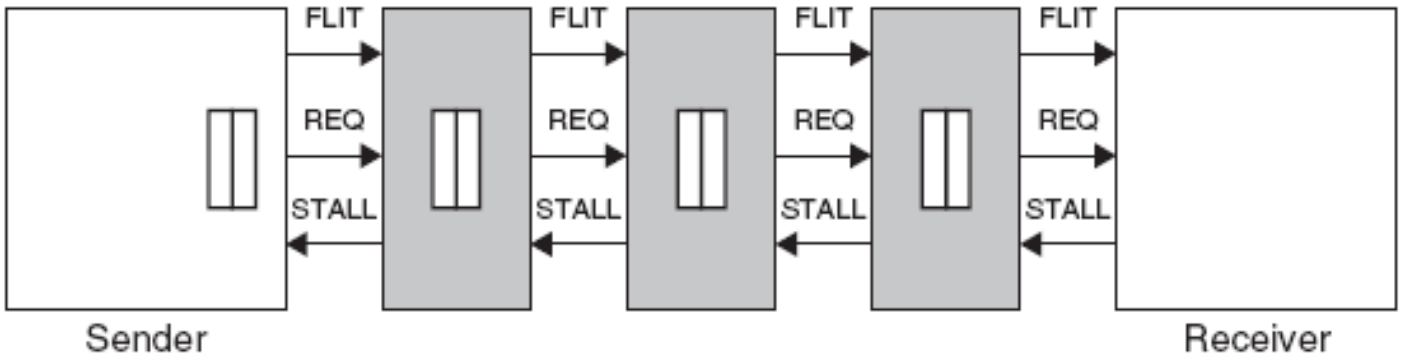
Flow Control Schemes

- Flow control mechanism also support recovery from the transmission errors at the data link level.
  - e.g., if a corrupted packet needs to be retransmitted, flow of packets from the sender must be stopped, and request signaling must be performed to reallocate buffer and bandwidth resources
- Flow control techniques can manage link congestion.
- However, all the schemes cannot (by themselves) reallocate all the resources required for retransmission when errors occur
  - either error correction or a scheme to handle reliable transfers must be implemented at a higher layer

Buffer Backpressure and Overflow

- Flow control WITH resource reservation needs mechanism to prevent buffer overflow
  - Avoid dropping packets
  - Upstream nodes need to know buffer availability at downstream routers
    - Upstream = closer to source
    - Downstream = closer to destination
- Significant impact on throughput achieved by flow control

STALL/GO Flow Control

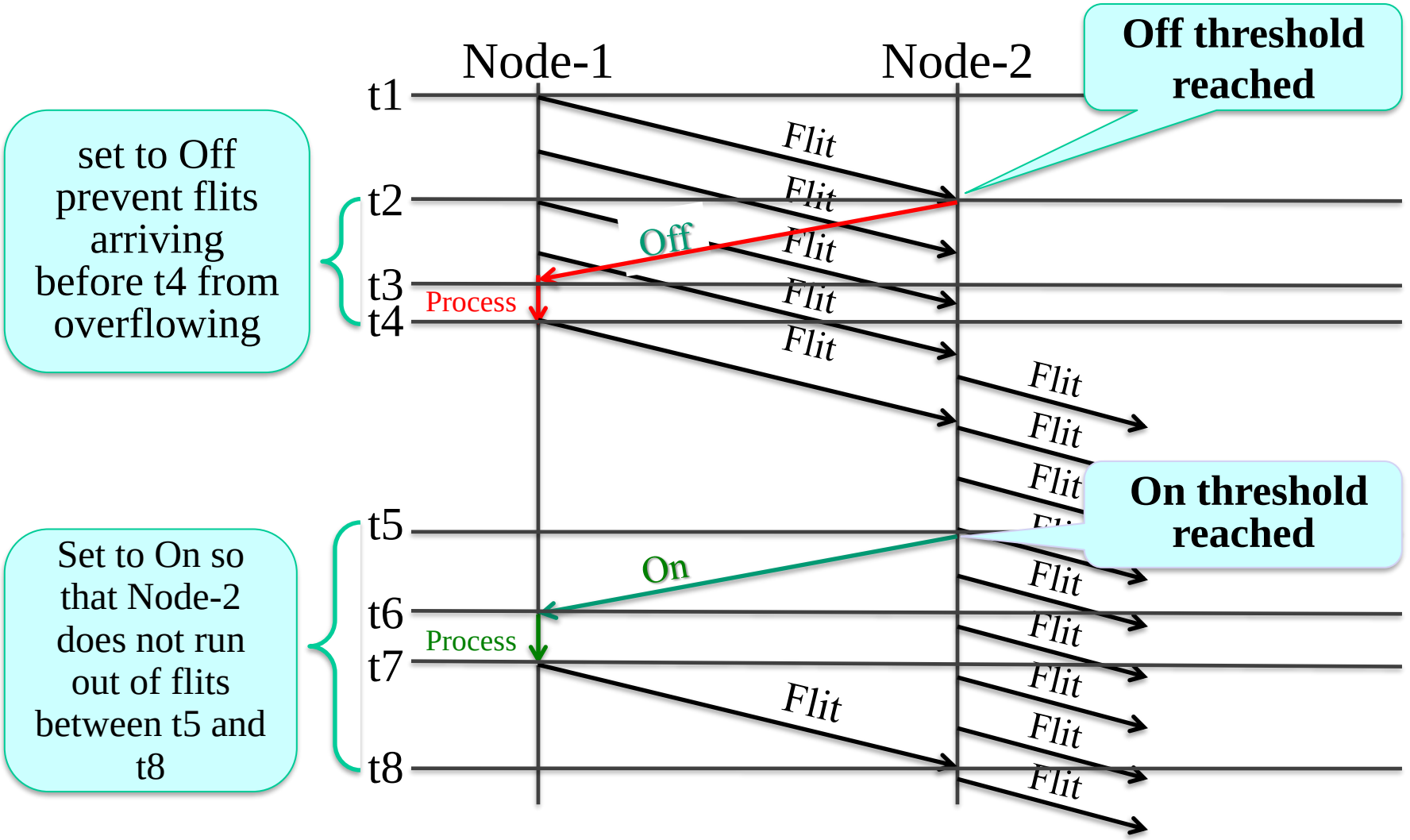


STALL/GO (A flow control with resource reservation)

- Low overhead scheme and requires only two control wires.
  - REQ (Request) - going forward and signaling data availability
  - STALL/GO - going backward and signaling either a condition of buffers filled (STALL) or buffers free (GO)
- Implemented with distributed buffering (pipelining) along link.
- good performance – fast recovery from congestion.
- No provision for fault handling.

  Higher-level protocols responsible for handling flit interruption

Stall/Go Variation – On/Off Scheme (with thresholds)



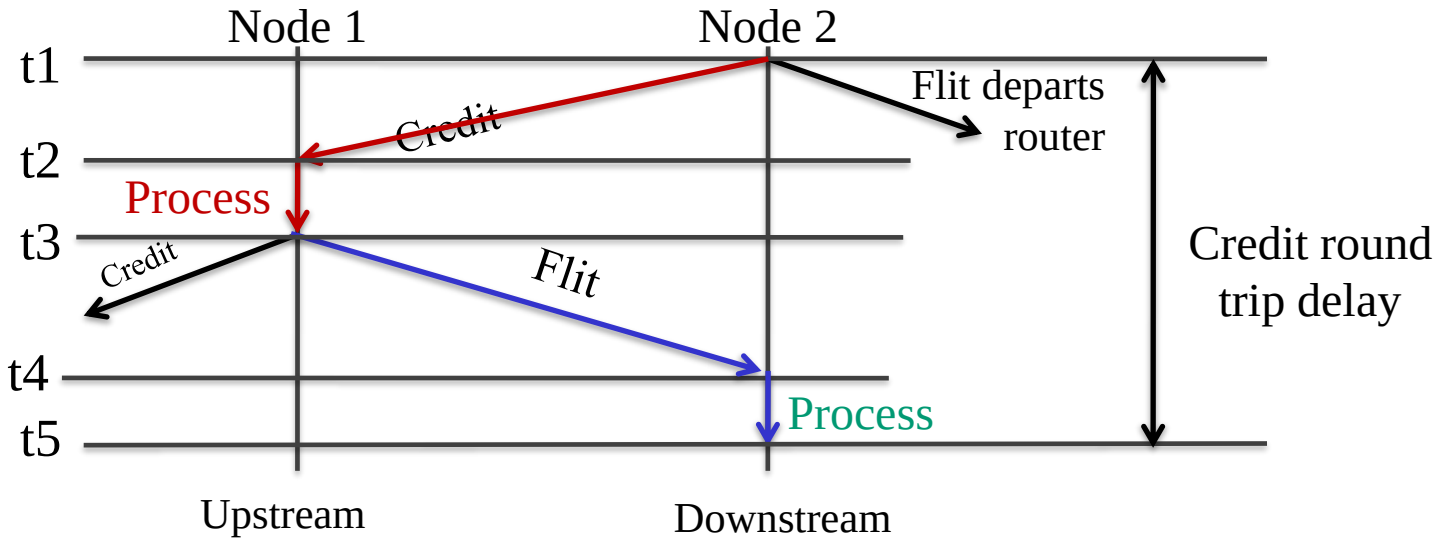
Credit-Based Flow Control

Upstream router stores credit counts for each downstream channel.

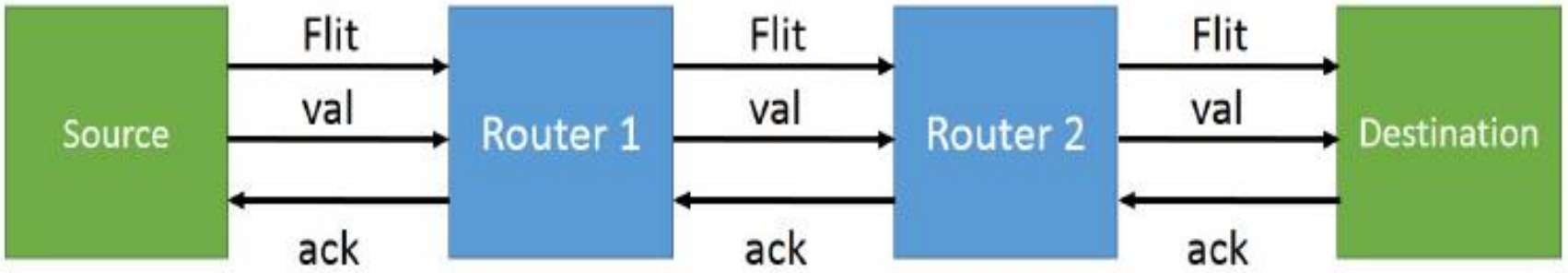
Upstream Router role: When flit forwarded to the next router, decrement the credit count. Count == 0, buffer full, stop sending

Downstream Router role: When flit forwarded & buffer slot freed, send credit to upstream router.

Upstream increments credit count, knows that there is now buffer space



ACK/NACK Flow Control



Flits are sent on a link, and a local copy is kept in a buffer by sender

- When ACK received, sender deletes copy of the flit from the buffer
- When NACK is received, sender rewinds its output queue and starts resending flits, starting from the corrupted one

Implemented either end-to-end or switch-to-switch

Sender needs to have a buffer of size $2N + k$

- N is number of buffers encountered between source and destination and k depends on latency of logic at the sender and receiver

- Fault handling at the cost of greater power and chip area overhead

Flow Control Schemes

Flow Control without Resource Reservation

- Technique #1: drop packets when receiver NI full.

  Improves congestion in short term. However, increases it in the long run.
- Technique #2: return packets that do not fit into the receiver buffers to the sender

  Avoid deadlock, rejected packets must be accepted by the sender
- Technique #3: Deflection routing (Hot-Potato Routing)
  - when packet cannot be accepted at receiver, it is sent back into network
  - packet does not go back to sender, but keeps hopping from router to router till it is accepted at receiver

Flow Control with Resource Reservation

- credit-based flow control with resource reservation
- credit counter at sender NI tracks free space available in the receiver NI buffers
- credit packets can piggyback on response packets

Flow Control Summary

- On-chip networks require techniques with lower buffering requirements
  - Wormhole or Virtual Channel flow control
- Avoid dropping packets in on-chip environment
  - Requires buffer backpressure mechanism
- Complexity of flow control impacts router architecture.
  - Discuss it in the next lecture