

# Tutorial: Networks on Chip

The 1st ACM/IEEE International Symposium on Networks-on-Chip

Princeton, New Jersey, May 6, 2007

- **Network Layer Communication Performance in Network-on-Chips**  
A. Jantsch (Royal Institute of Technology, Sweden)  
10:00 - 11:45 am
- **Power, Energy and Reliability Issues in NoC**  
R. Marculescu (Carnegie Mellon University, USA)  
1:15 - 3:00 pm
- **Tooling, OS Services and Middleware**  
L. Benini (University of Bologna, Italy)  
3:15 - 5:00 pm

# Network Layer Communication Performance in Network-on-Chips

Introduction

Communication Performance

Organizational Structure

Interconnection Topologies

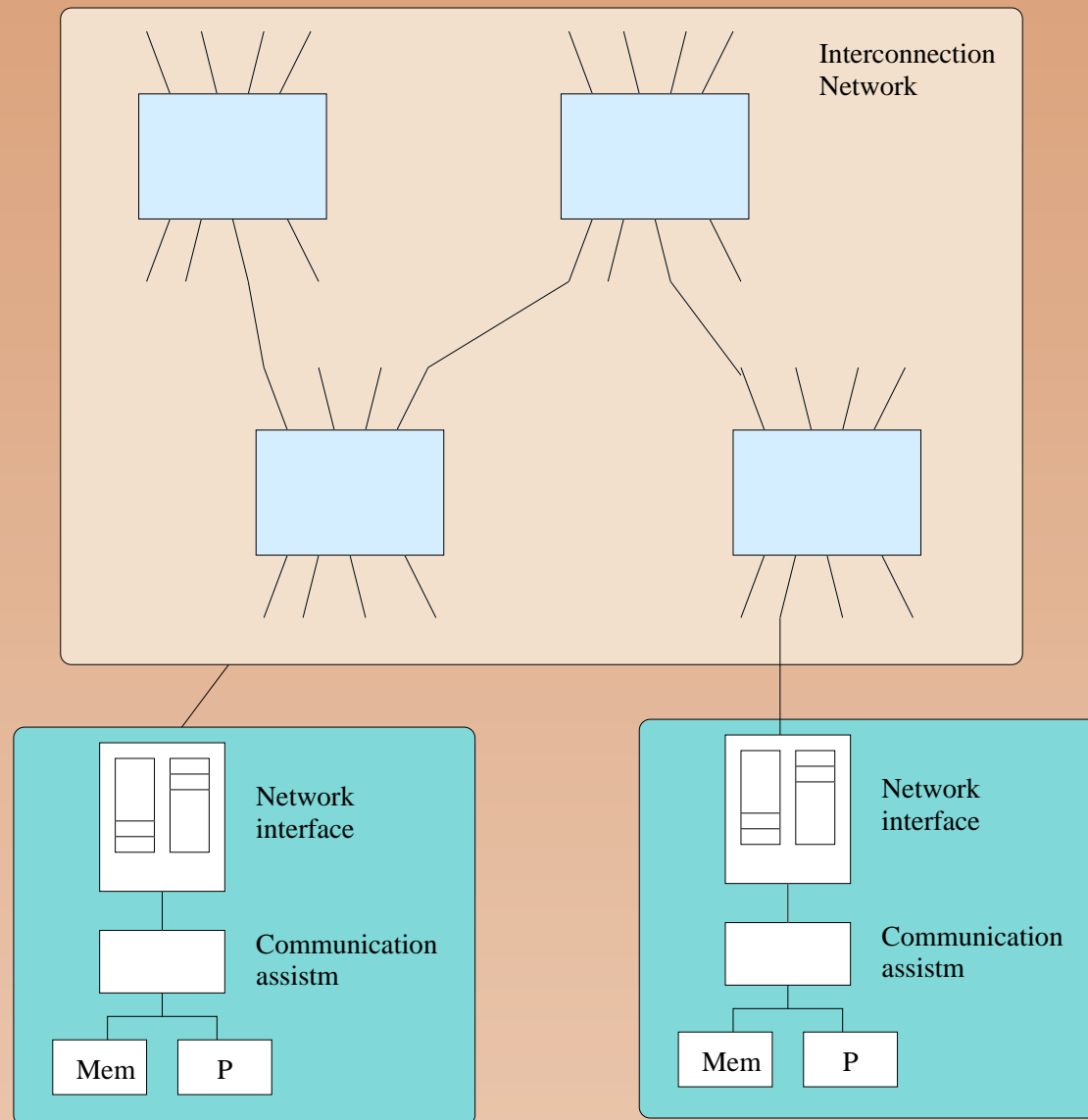
Trade-offs in Network Topology

Routing

Quality of Service



# Introduction



- **Topology:** How switches and nodes are connected
- **Routing algorithm:** determines the route from source to destination
- **Switching strategy:** how a message traverses the route
- **Flow control:** Schedules the traversal of the message over time

# Basic Definitions



## Basic Definitions

**Message** is the basic communication entity.

**Flit** is the basic flow control unit. A message consists of 1 or many flits.

**Phit** is the basic unit of the physical layer.



## Basic Definitions

**Message** is the basic communication entity.

**Flit** is the basic flow control unit. A message consists of 1 or many flits.

**Phit** is the basic unit of the physical layer.

**Direct network** is a network where each switch connects to a node.

**Indirect network** is a network with switches not connected to any node.



## Basic Definitions

**Message** is the basic communication entity.

**Flit** is the basic flow control unit. A message consists of 1 or many flits.

**Phit** is the basic unit of the physical layer.

**Direct network** is a network where each switch connects to a node.

**Indirect network** is a network with switches not connected to any node.

**Hop** is the basic communication action from node to switch or from switch to switch.



## Basic Definitions

**Message** is the basic communication entity.

**Flit** is the basic flow control unit. A message consists of 1 or many flits.

**Phit** is the basic unit of the physical layer.

**Direct network** is a network where each switch connects to a node.

**Indirect network** is a network with switches not connected to any node.

**Hop** is the basic communication action from node to switch or from switch to switch.

**Diameter** is the length of the maximum shortest path between any two nodes measured in hops.

**Routing distance** between two nodes is the number of hops on a route.

**Average distance** is the average of the routing distance over all pairs of nodes.





# Basic Switching Techniques

**Circuit Switching** A real or virtual circuit establishes a direct connection between source and destination.



## Basic Switching Techniques

**Circuit Switching** A real or virtual circuit establishes a direct connection between source and destination.

**Packet Switching** Each packet of a message is routed independently. The destination address has to be provided with each packet.



## Basic Switching Techniques

**Circuit Switching** A real or virtual circuit establishes a direct connection between source and destination.

**Packet Switching** Each packet of a message is routed independently. The destination address has to be provided with each packet.

**Store and Forward Packet Switching** The entire packet is stored and then forwarded at each switch.



## Basic Switching Techniques

**Circuit Switching** A real or virtual circuit establishes a direct connection between source and destination.

**Packet Switching** Each packet of a message is routed independently. The destination address has to be provided with each packet.

**Store and Forward Packet Switching** The entire packet is stored and then forwarded at each switch.

**Cut Through Packet Switching** The flits of a packet are pipelined through the network. The packet is not completely buffered in each switch.



## Basic Switching Techniques

**Circuit Switching** A real or virtual circuit establishes a direct connection between source and destination.

**Packet Switching** Each packet of a message is routed independently. The destination address has to be provided with each packet.

**Store and Forward Packet Switching** The entire packet is stored and then forwarded at each switch.

**Cut Through Packet Switching** The flits of a packet are pipelined through the network. The packet is not completely buffered in each switch.

**Virtual Cut Through Packet Switching** The entire packet is stored in a switch only when the header flit is blocked due to congestion.



## Basic Switching Techniques

**Circuit Switching** A real or virtual circuit establishes a direct connection between source and destination.

**Packet Switching** Each packet of a message is routed independently. The destination address has to be provided with each packet.

**Store and Forward Packet Switching** The entire packet is stored and then forwarded at each switch.

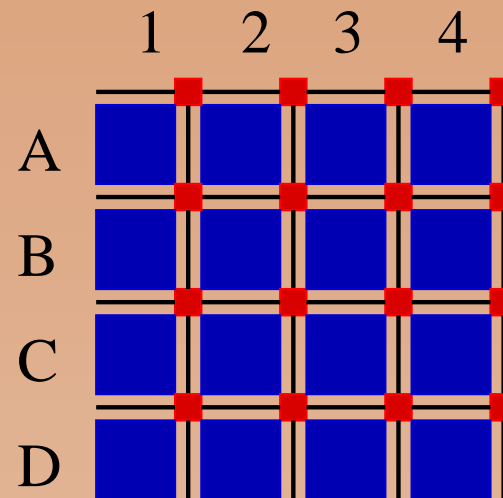
**Cut Through Packet Switching** The flits of a packet are pipelined through the network. The packet is not completely buffered in each switch.

**Virtual Cut Through Packet Switching** The entire packet is stored in a switch only when the header flit is blocked due to congestion.

**Wormhole Switching** is cut through switching and all flits are blocked on the spot when the header flit is blocked.



## Latency



$$\text{Time}(n) = \text{Admission} + \text{RoutingDelay} + \text{ContentionDelay}$$

**Admission** is the time it takes to emit the message into the network.

**RoutingDelay** is the delay for the route.

**ContentionDelay** is the delay of a message due to contention.

## Routing Delay

Store and Forward:

$$T_{sf}(n, h) = h\left(\frac{n}{b} + \Delta\right)$$

$n$  ... message size in bits

$n_p$  ... size of message fragments in bits

$h$  ... number of hops

$b$  ... raw bandwidth of the channel

$\Delta$  ... switching delay per hop





## Routing Delay

Store and Forward:

$$T_{sf}(n, h) = h\left(\frac{n}{b} + \Delta\right)$$

Circuit Switching:

$$T_{cs}(n, h) = \frac{n}{b} + h\Delta$$

$n$  ... message size in bits

$n_p$  ... size of message fragments in bits

$h$  ... number of hops

$b$  ... raw bandwidth of the channel

$\Delta$  ... switching delay per hop



## Routing Delay

Store and Forward:

$$T_{sf}(n, h) = h\left(\frac{n}{b} + \Delta\right)$$

Circuit Switching:

$$T_{cs}(n, h) = \frac{n}{b} + h\Delta$$

Cut Through:

$$T_{ct}(n, h) = \frac{n}{b} + h\Delta$$

$n$  ... message size in bits

$n_p$  ... size of message fragments in bits

$h$  ... number of hops

$b$  ... raw bandwidth of the channel

$\Delta$  ... switching delay per hop



## Routing Delay

Store and Forward:

$$T_{sf}(n, h) = h\left(\frac{n}{b} + \Delta\right)$$

Circuit Switching:

$$T_{cs}(n, h) = \frac{n}{b} + h\Delta$$

Cut Through:

$$T_{ct}(n, h) = \frac{n}{b} + h\Delta$$

Store and Forward with  
fragmented packets:

$$T_{sf}(n, h, n_p) = \frac{n - n_p}{b} + h\left(\frac{n_p}{b} + \Delta\right)$$

$n$  ... message size in bits

$n_p$  ... size of message fragments in bits

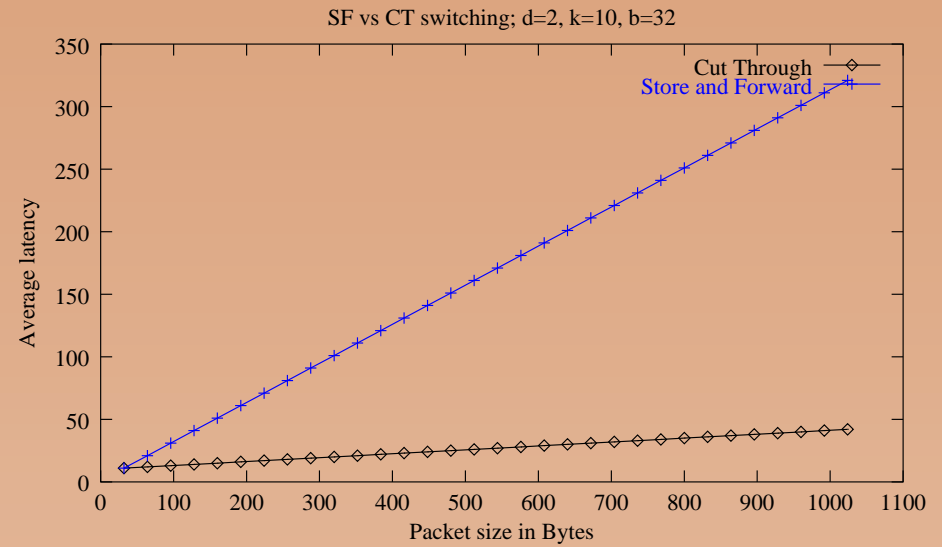
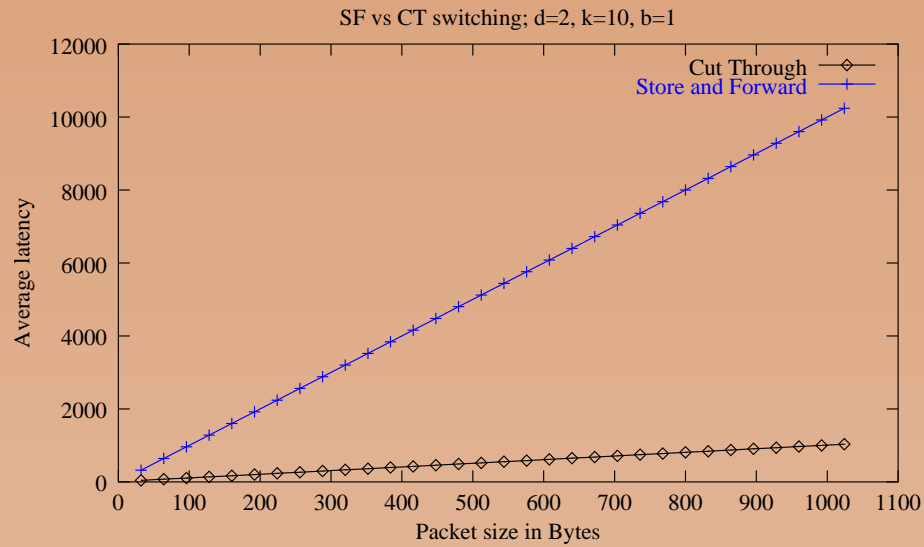
$h$  ... number of hops

$b$  ... raw bandwidth of the channel

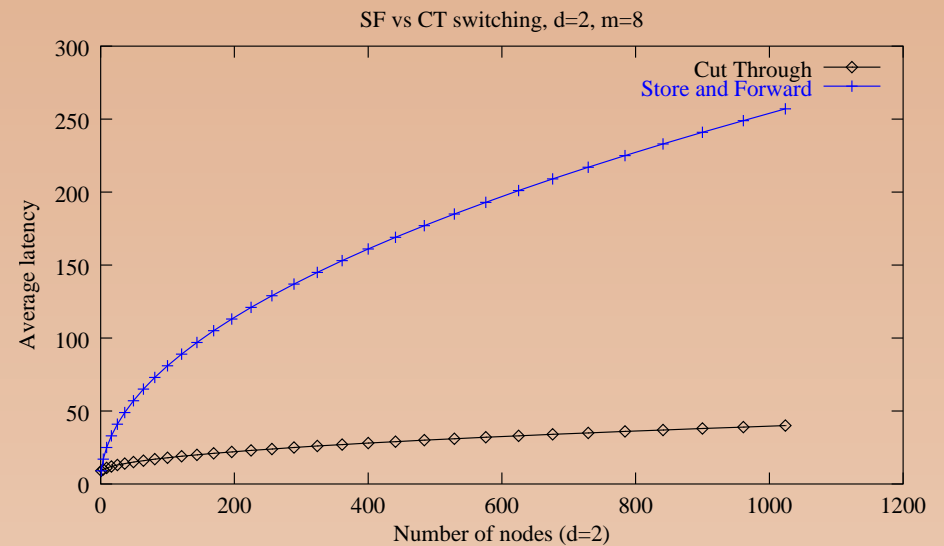
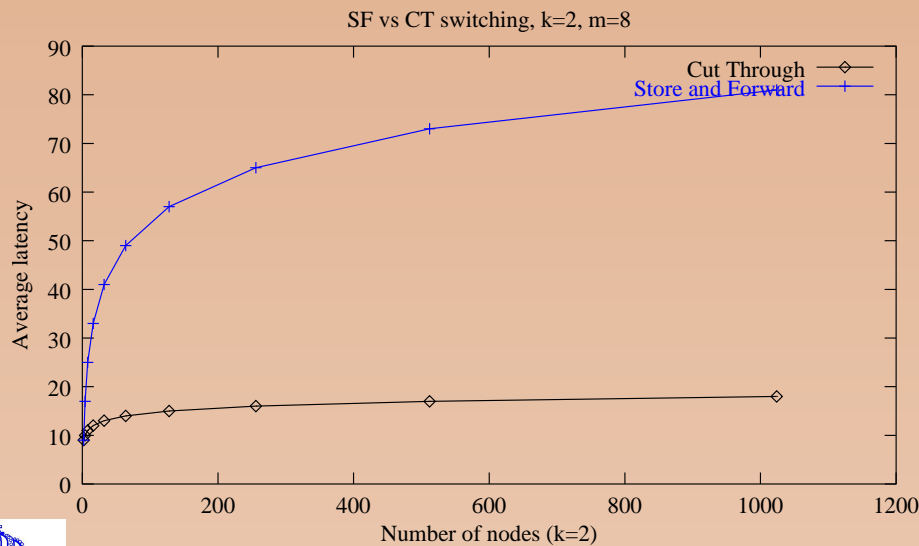
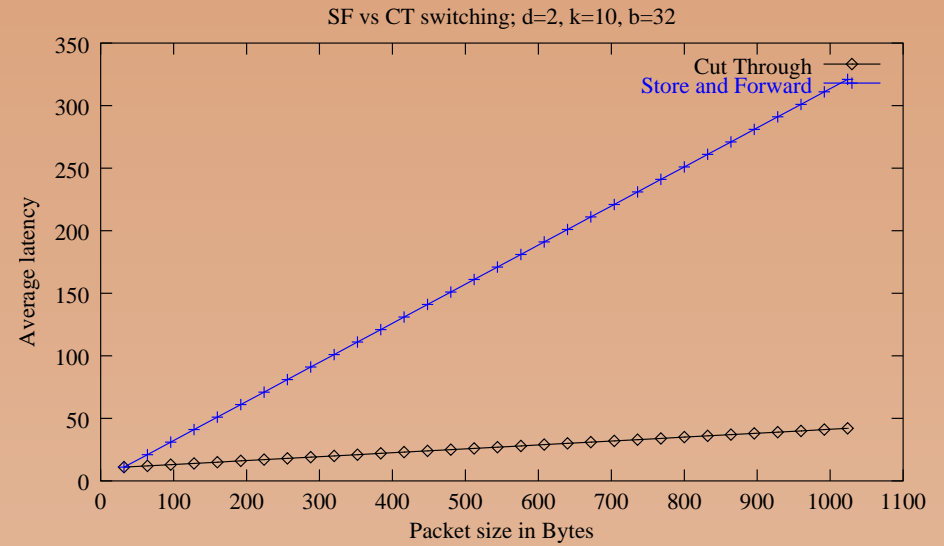
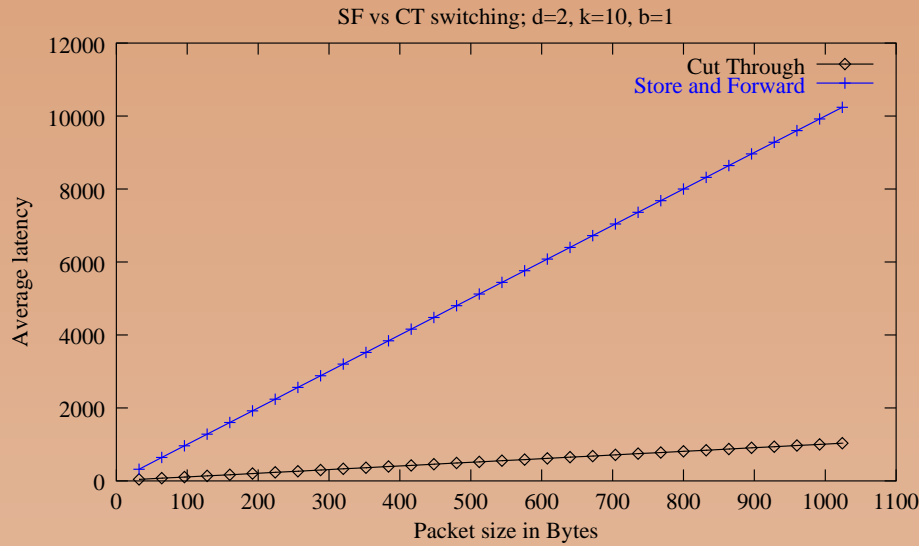
$\Delta$  ... switching delay per hop



# Routing Delay: Store and Forward vs Cut Through



# Routing Delay: Store and Forward vs Cut Through



# Local and Global Bandwidth

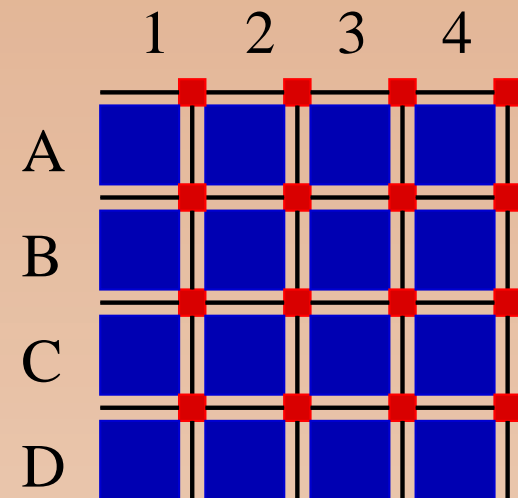
- Local bandwidth** =  $b \left( \frac{n}{n+n_E+w\Delta} \right)$
- Total bandwidth** =  $Cb[\text{bits/second}] = Cw[\text{bits/cycle}] = C[\text{phits/cycle}]$
- Bisection bandwidth** ... minimum bandwidth to cut the net into two equal parts.

$b$  ... raw bandwidth of a link;  
 $n$  ... message size;  
 $n_E$  ... size of message envelope;  
 $w$  ... link bandwidth per cycle;

$\Delta$  ... switching time for each switch in cycles;  
 $w\Delta$  ... bandwidth lost during switching;  
 $C$  ... total number of channels;

For a  $k \times k$  mesh with bidirectional channels:

Total bandwidth =  $(4k^2 - 4k)b$   
 Bisection bandwidth =  $2kb$



## Link and Network Utilization

**total load on the network:**  $L = \frac{Nhl}{M}$  [phits/cycle]

**load per channel:**  $\rho = \frac{Nhl}{MC}$  [phits/cycle]  $\leq 1$

$M$  ... each host issues a packet every  $M$  cycles

$C$  ... number of channels

$N$  ... number of nodes

$h$  ... average routing distance

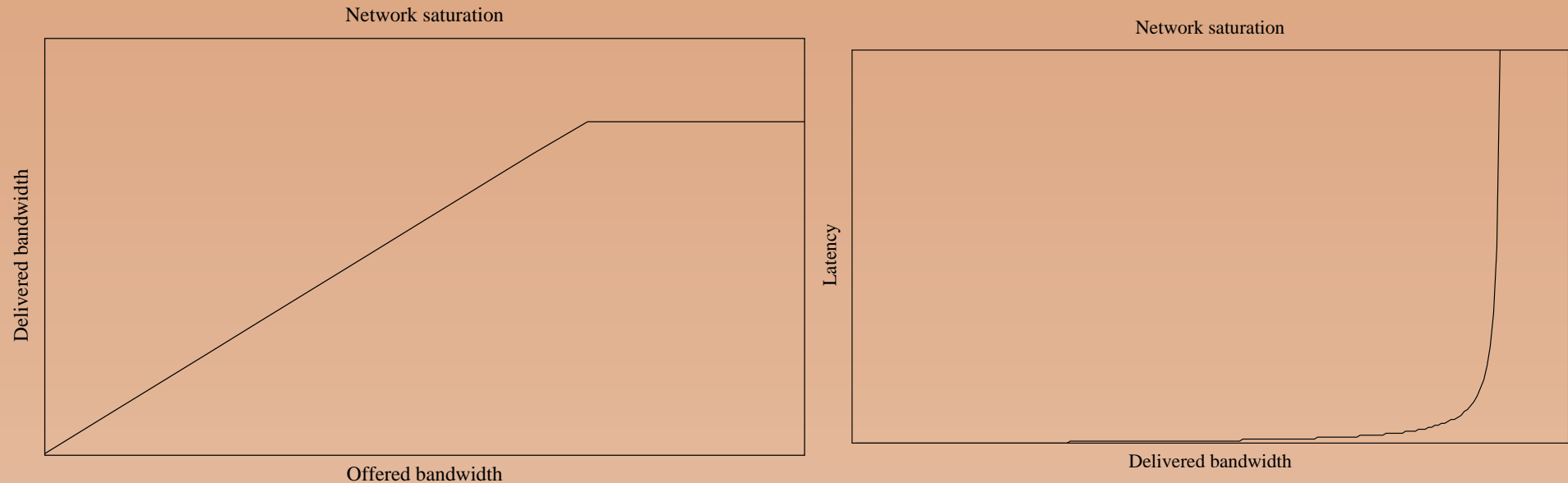
$l = n/w$  ... number of cycles a message occupies a channel

$n$  ... average message size

$w$  ... bandwidth per channel



# Network Saturation



Typical saturation points are between 40% and 70%.  
The saturation point depends on

- Traffic pattern
- Stochastic variations in traffic
- Routing algorithm





# Organizational Structure

- Link
- Switch
- Network Interface



# Link

**Short link** At any time there is only one data word on the link.

**Long link** Several data words can travel on the link simultaneously.

**Narrow link** Data and control information is multiplexed on the same wires.

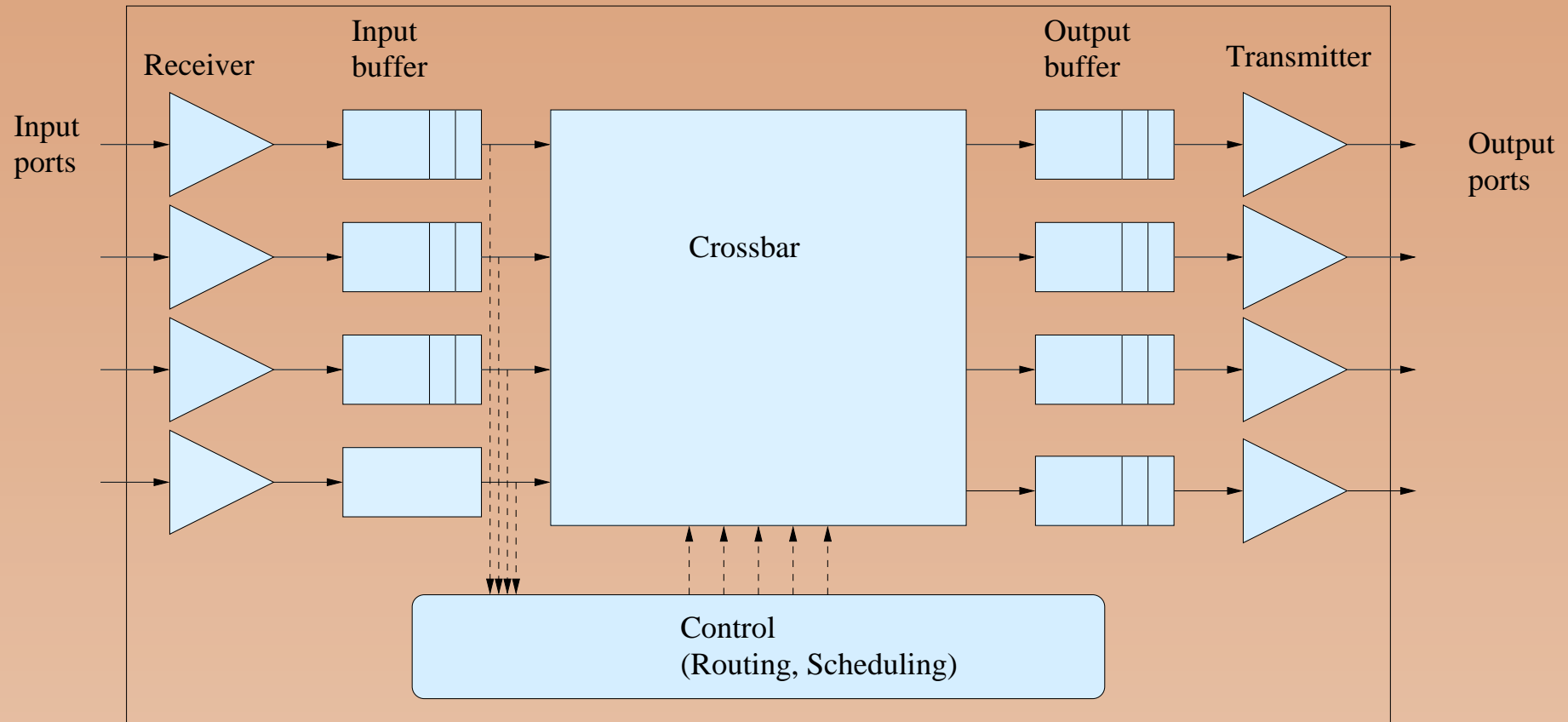
**Wide link** Data and control information is transmitted in parallel and simultaneously.

**Synchronous clocking** Both source and destination operate on the same clock.

**Asynchronous clocking** The clock is encoded in the transmitted data to allow the receiver to sample at the right time instance.



# Switch



# Switch Design Issues

**Degree:** number of inputs and outputs;

## Buffering

- Input buffers
- Output buffers
- Shared buffers

## Routing

- Source routing
- Deterministic routing
- Adaptive routing

## Output scheduling

## Deadlock handling

## Control flow



# Network Interface

- Admission protocol
- Reception obligations
- Buffering
- Assembling and disassembling of messages
- Routing
- Higher level services and protocols

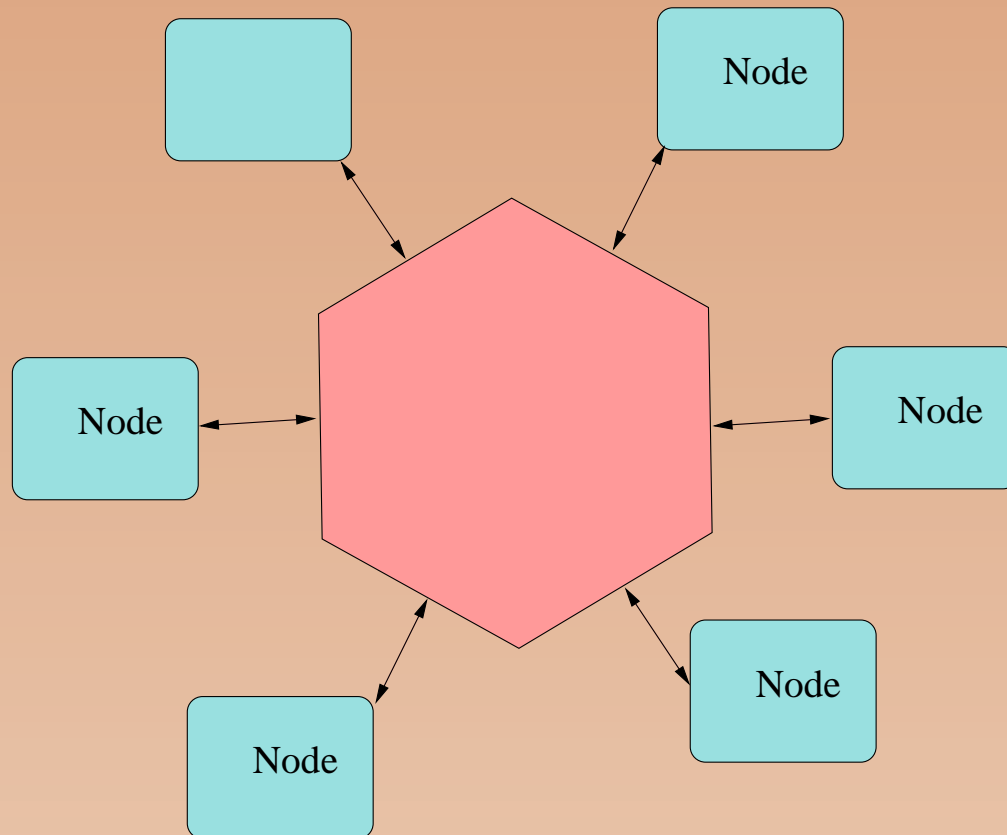


# Interconnection Topologies

- Fully connected networks
- Linear arrays and rings
- Multidimensional meshes and tori
- Trees
- Butterflies



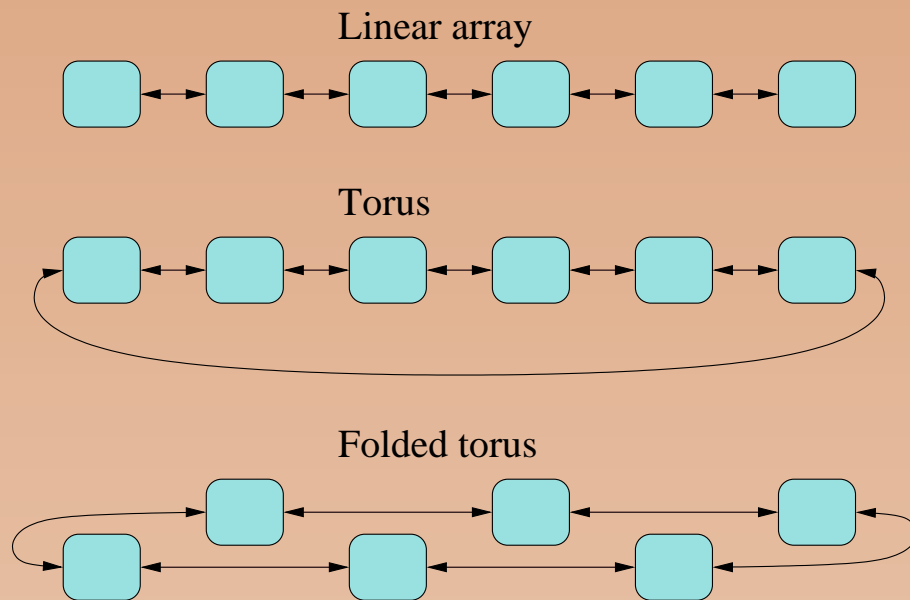
# Fully Connected Networks



Bus: switch degree =  $N$   
 diameter = 1  
 distance = 1  
 network cost =  $O(N)$   
 total bandwidth =  $b$   
 bisection =  $b$   
 bandwidth

Crossbar: switch degree =  $N$   
 diameter = 1  
 distance = 1  
 network cost =  $O(N^2)$   
 total bandwidth =  $Nb$   
 bisection =  $Nb$   
 bandwidth

# Linear Arrays and Rings



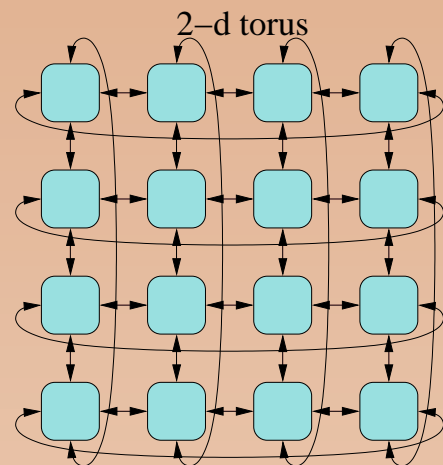
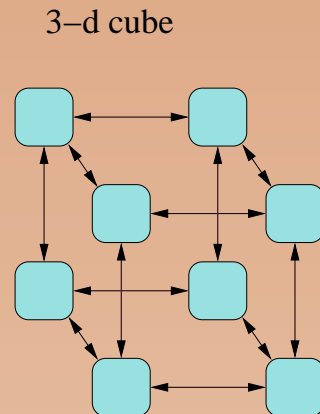
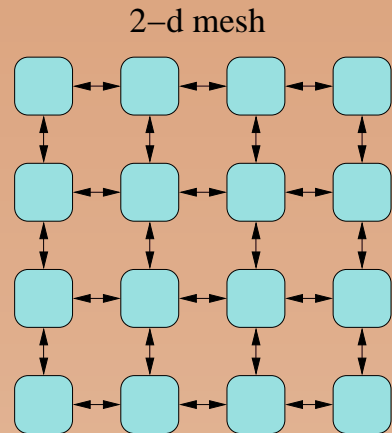
## Linear

array: switch degree	=	2
diameter	=	$N - 1$
distance	$\sim$	$2/3N$
network cost	=	$O(N)$
total bandwidth	=	$2(N - 1)b$
bisection	=	$2b$
bandwidth		

Torus: switch degree	=	2
diameter	=	$N/2$
distance	$\sim$	$1/3N$
network cost	=	$O(N)$
total bandwidth	=	$2Nb$
bisection	=	$4b$
bandwidth		



## Multidimensional Meshes and Tori



**$k$ -ary  $d$ -cubes** are  $d$ -dimensional tori with unidirectional links and  $k$  nodes in each dimension:

$$\text{number of nodes } N = k^d$$

$$\text{switch degree} = d$$

$$\text{diameter} = d(k - 1)$$

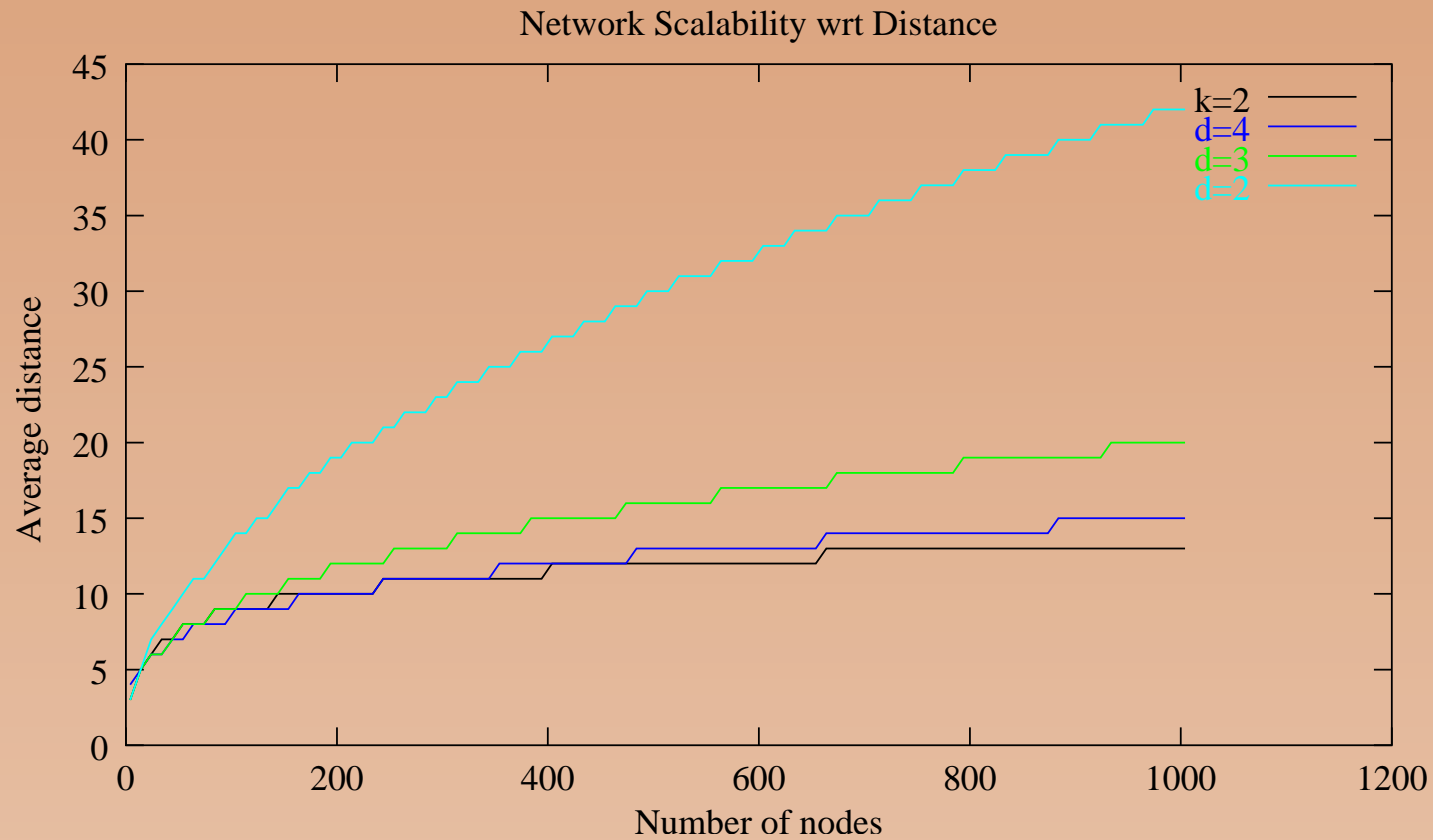
$$\text{distance} \sim d \frac{1}{2} (k - 1)$$

$$\text{network cost} = O(N)$$

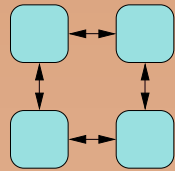
$$\text{total bandwidth} = 2Nb$$

$$\text{bisection bandwidth} = 2k^{(d-1)}b$$

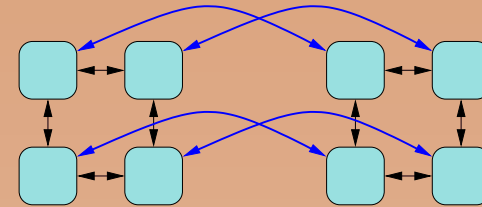
# Routing Distance in $k$ -ary $n$ -Cubes



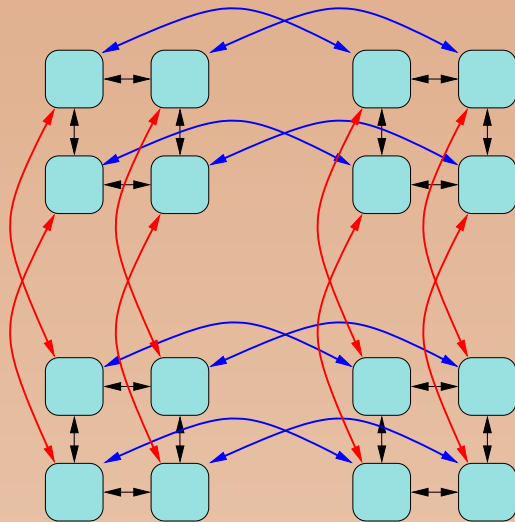
# Projecting High Dimensional Cubes



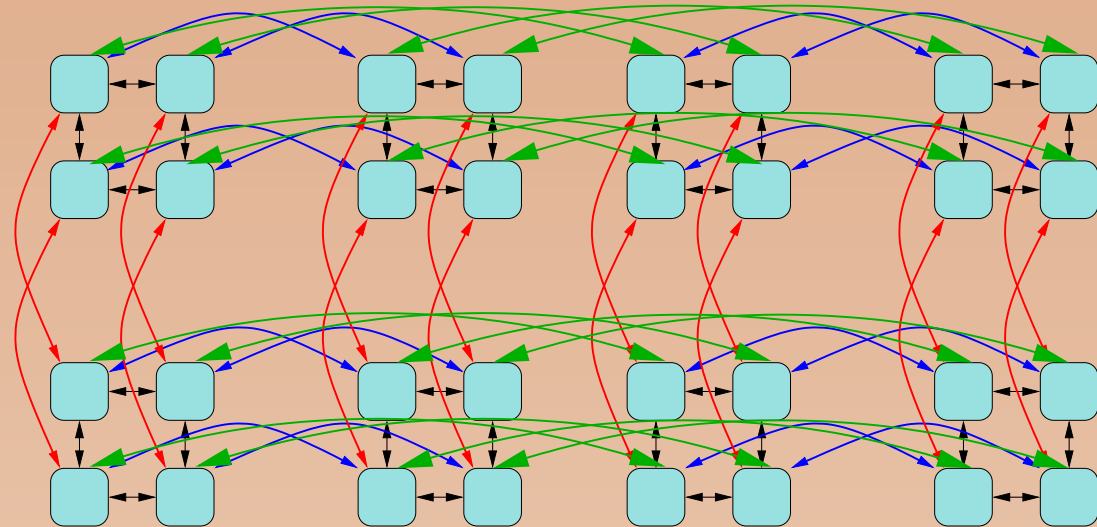
2-ary 2-cube



2-ary 3-cube



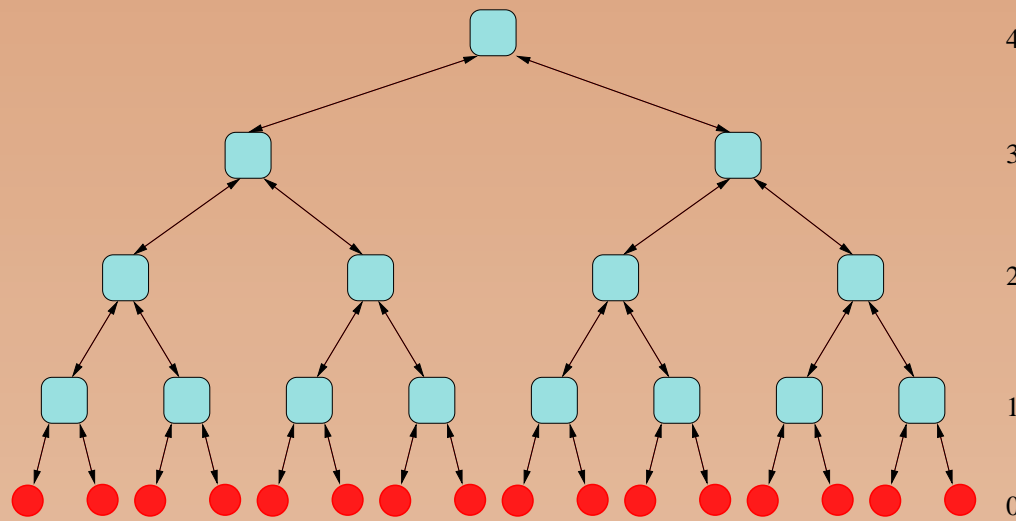
2-ary 4-cube



2-ary 5-cube

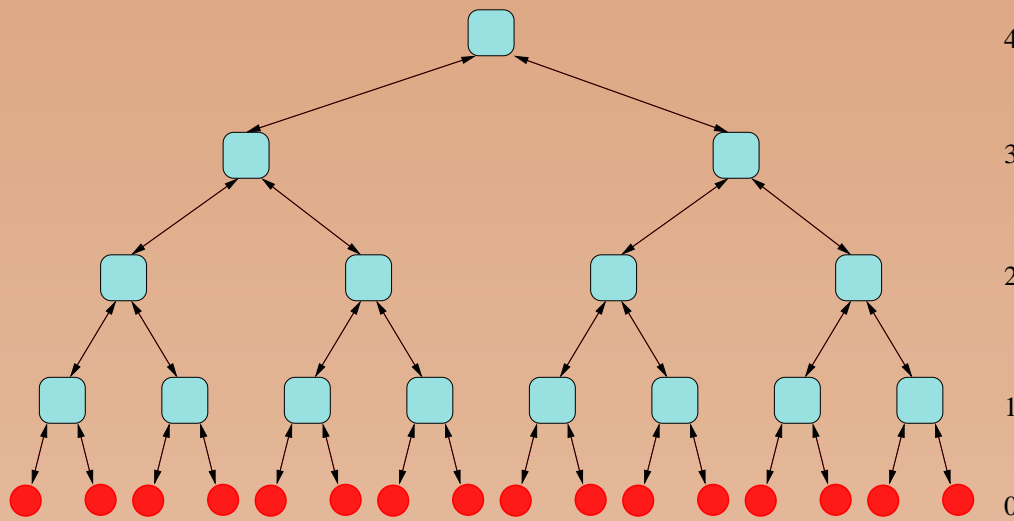


# Binary Trees



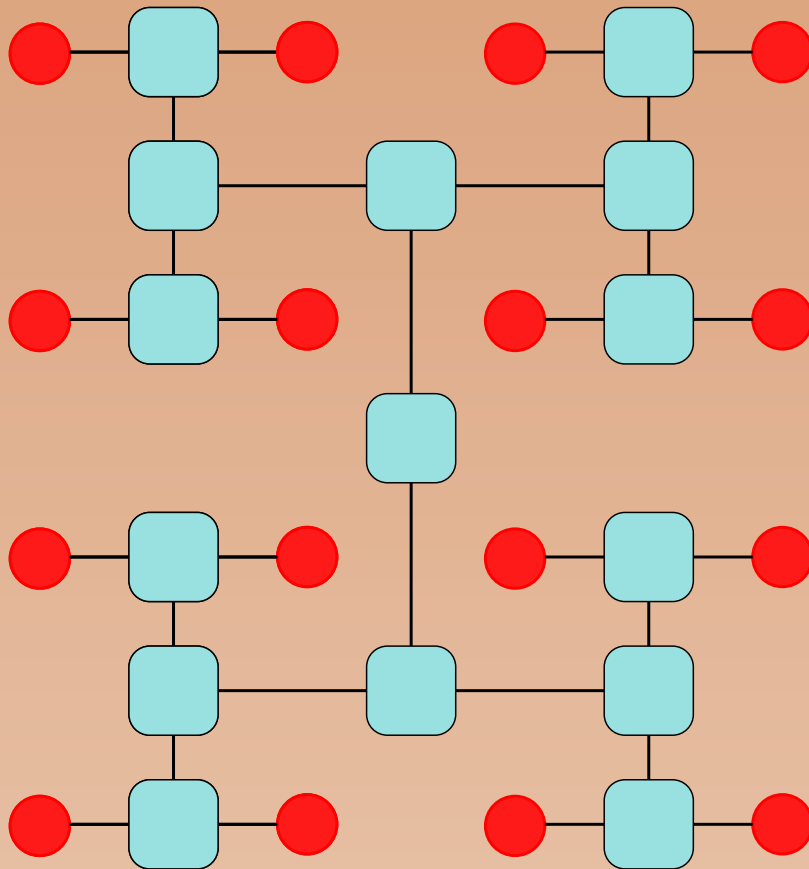
4	number of nodes $N$	$=$	$2^d$
3	number of switches	$=$	$2^d - 1$
2	switch degree	$=$	3
1	diameter	$=$	$2d$
0	distance	$\sim$	$d + 2$
	network cost	$=$	$O(N)$
	total bandwidth	$=$	$2 \cdot 2(N - 1)b$
	bisection bandwidth	$=$	$2b$

# $k$ -ary Trees



4	number of nodes $N$	$=$	$k^d$
	number of switches	$\sim$	$k^d$
3	switch degree	$=$	$k + 1$
	diameter	$=$	$2d$
2	distance	$\sim$	$d + 2$
	network cost	$=$	$O(N)$
1	total bandwidth	$=$	$2 \cdot 2(N - 1)b$
0	bisection bandwidth	$=$	$kb$

# Binary Tree Projection



- Efficient and regular 2-layout;
- Longest wires in resource width:

$$lW = 2^{\lfloor \frac{d-1}{2} \rfloor} - 1$$

$d$	2	3	4	5	6	7	8	9	10
$N$	4	8	16	32	64	128	256	512	1024
$lW$	0	1	1	2	2	4	4	8	8



## $k$ -ary $n$ -Cubes versus $k$ -ary Trees

$k$ -ary  $n$ -cubes:

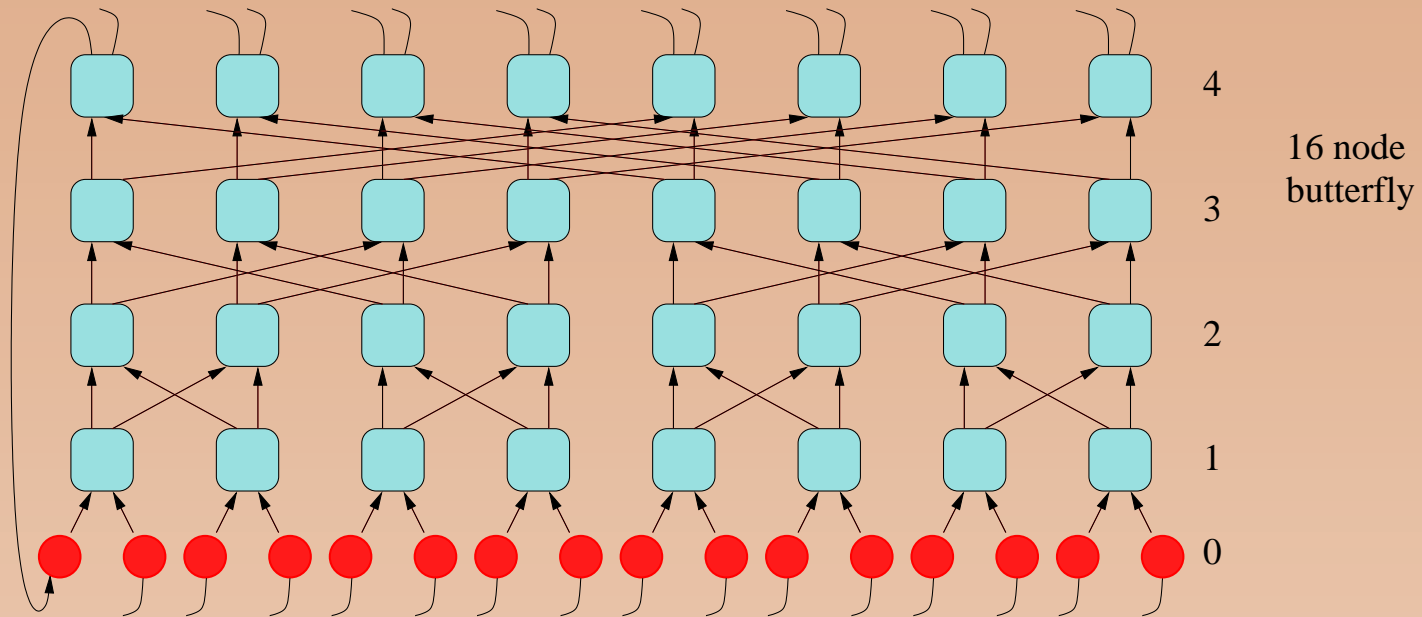
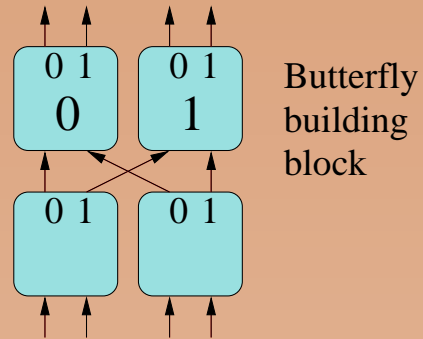
number of nodes $N$	$=$	$k^d$
switch degree	$=$	$d + 2$
diameter	$=$	$d(k - 1)$
distance	$\sim$	$d\frac{1}{2}(k - 1)$
network cost	$=$	$O(N)$
total bandwidth	$=$	$2Nb$
bisection bandwidth	$=$	$2k^{(d-1)}b$

$k$ -ary trees:

number of nodes $N$	$=$	$k^d$
number of switches	$\sim$	$k^d$
switch degree	$=$	$k + 1$
diameter	$=$	$2d$
distance	$\sim$	$d + 2$
network cost	$=$	$O(N)$
total bandwidth	$=$	$2 \cdot 2(N - 1)b$
bisection bandwidth	$=$	$kb$

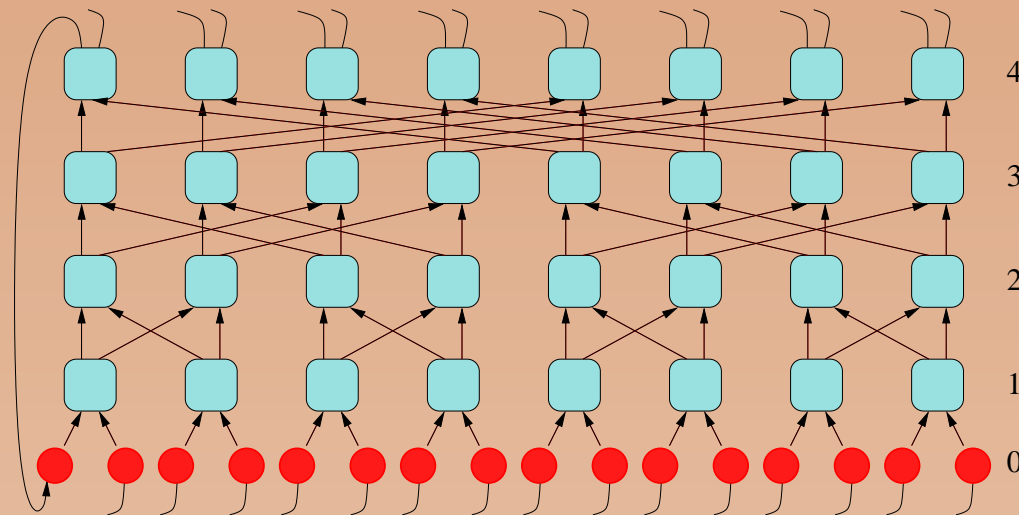


# Butterflies





# Butterfly Characteristics



number of nodes $N$	$= 2^d$
number of switches	$= 2^{d-1}d$
switch degree	$= 2$
diameter	$= d + 1$
distance	$= d + 1$
network cost	$= O(Nd)$
total bandwidth	$= 2^d db$
bisection bandwidth	$= \frac{N}{2}b$

## $k$ -ary $n$ -Cubes versus $k$ -ary Trees vs Butterflies

	$k$ -ary $n$ -cubes	binary tree	butterfly
cost	$O(N)$	$O(N)$	$O(N \log N)$
distance	$\frac{1}{2} \sqrt[d]{N} \log N$	$2 \log N$	$\log N$
links per node	2	2	$\log N$
bisection	$2N^{\frac{d-1}{d}}$	1	$\frac{1}{2}N$
frequency limit of random traffic	$1 / \left( \sqrt[d]{\frac{N}{2}} \right)$	$1/N$	$1/2$

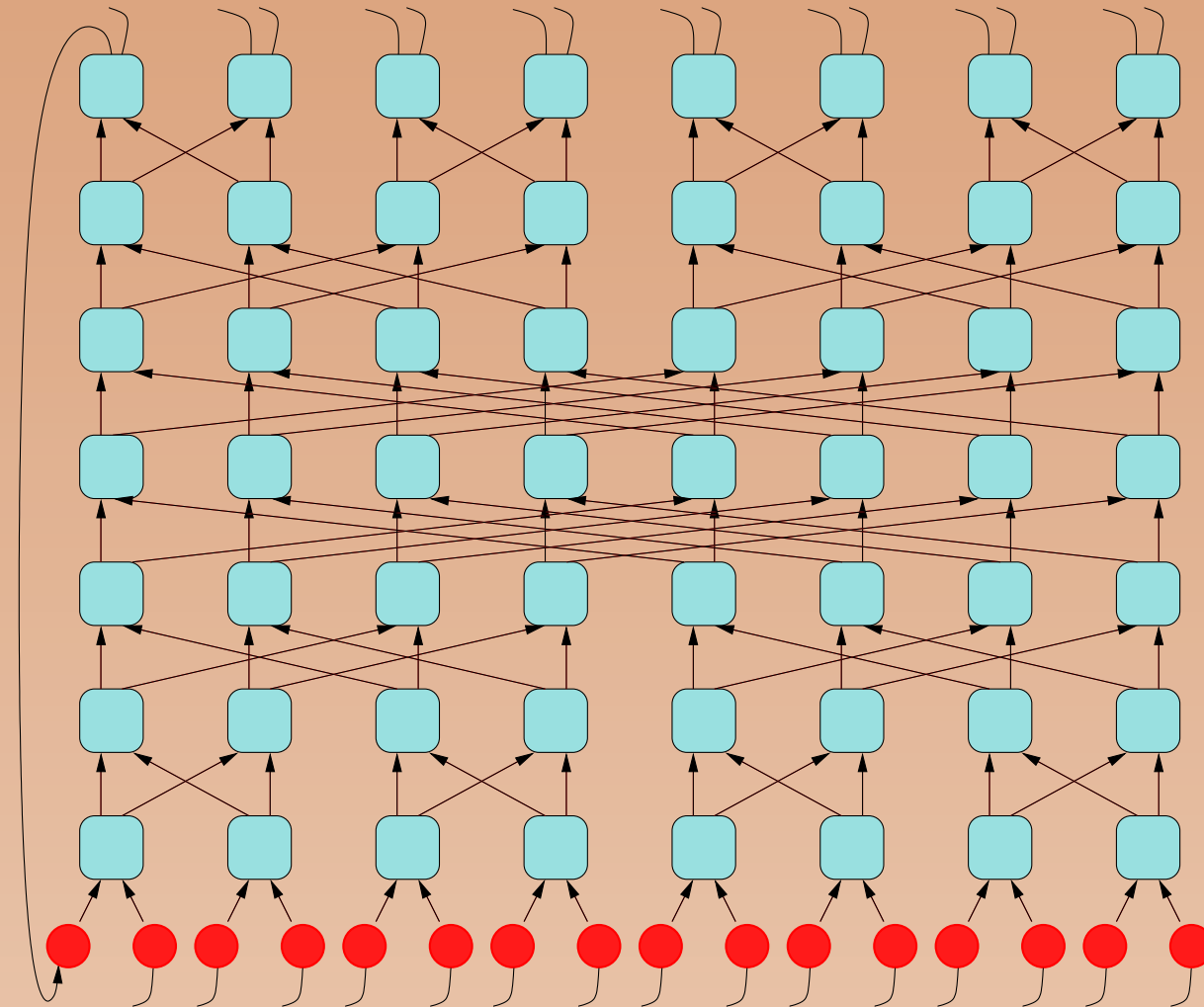


## Problems with Butterflies

- Cost of the network
  - ★  $O(N \log N)$
  - ★ 2-d layout is more difficult than for binary trees
  - ★ Number of long wires grows faster than for trees.
- For each source-destination pair there is only one route.
- Each route blocks many other routes.

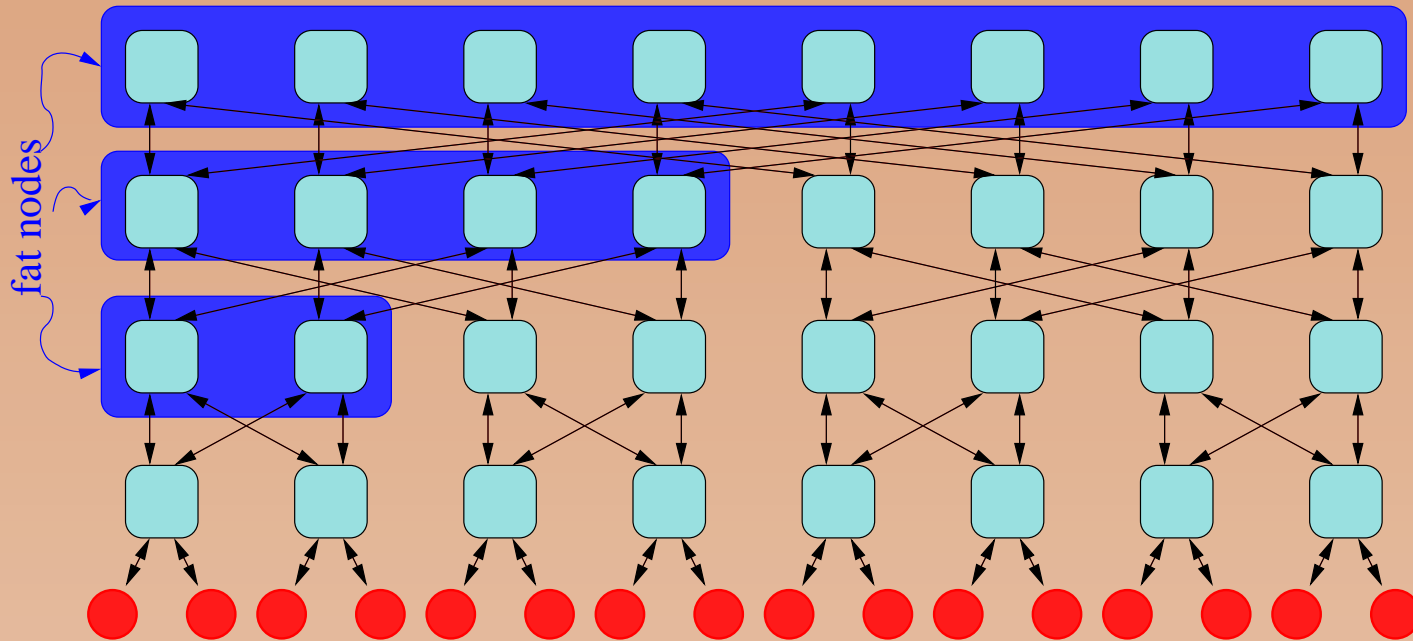


# Benes Networks



- Many routes;
- Costly to compute non-blocking routes;
- High probability for non-blocking route by randomly selecting an intermediate node [Leighton, 1992];

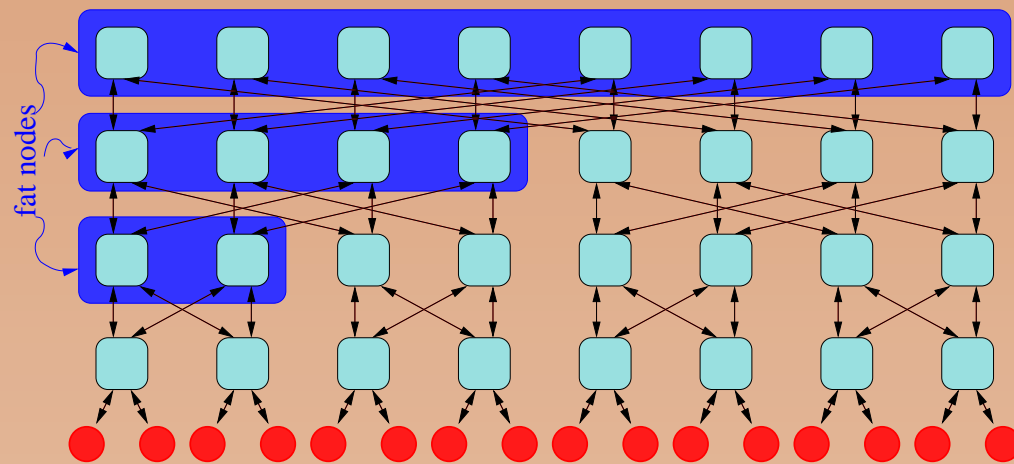
# Fat Trees



16-node 2-ary fat-tree



## $k$ -ary $n$ -dimensional Fat Tree Characteristics



16-node 2-ary fat-tree

number of nodes $N$	$=$	$k^d$
number of switches	$=$	$k^{d-1}d$
switch degree	$=$	$2k$
diameter	$=$	$2d$
distance	$\sim$	$d$
network cost	$=$	$O(Nd)$
total bandwidth	$=$	$2k^d db$
bisection bandwidth	$=$	$2k^{d-1}b$

## $k$ -ary $n$ -Cubes versus $k$ -ary $d$ -dimensional Fat Trees

$k$ -ary  $n$ -cubes:

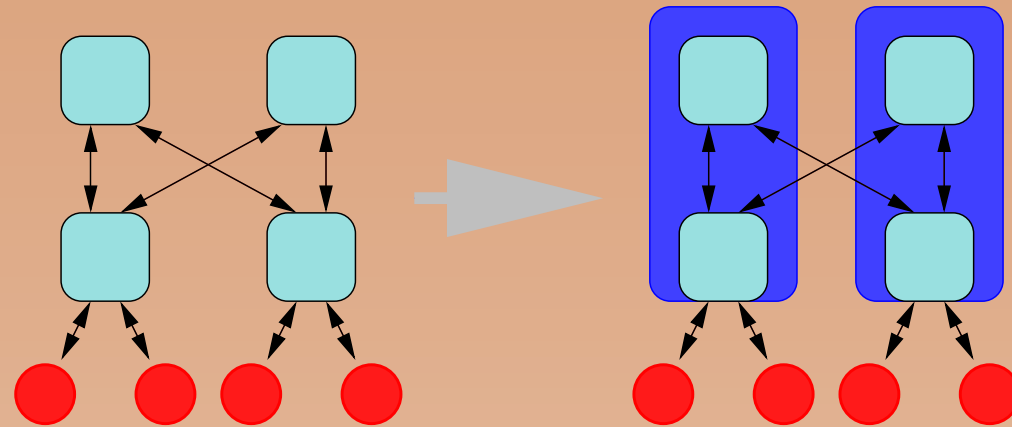
number of nodes $N$	=	$k^d$
switch degree	=	$d$
diameter	=	$d(k - 1)$
distance	$\sim$	$d\frac{1}{2}(k - 1)$
network cost	=	$O(N)$
total bandwidth	=	$2Nb$
bisection bandwidth	=	$2k^{(d-1)}b$

$k$ -ary  $n$ -dimensional fat trees:

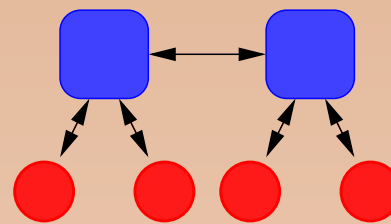
number of nodes $N$	=	$k^d$
number of switches	=	$k^{d-1}d$
switch degree	=	$2k$
diameter	=	$2d$
distance	$\sim$	$d$
network cost	=	$O(Nd)$
total bandwidth	=	$2k^d db$
bisection bandwidth	=	$2k^{d-1}b$



# Relation between Fat Tree and Hypercube



binary 2-dim fat tree

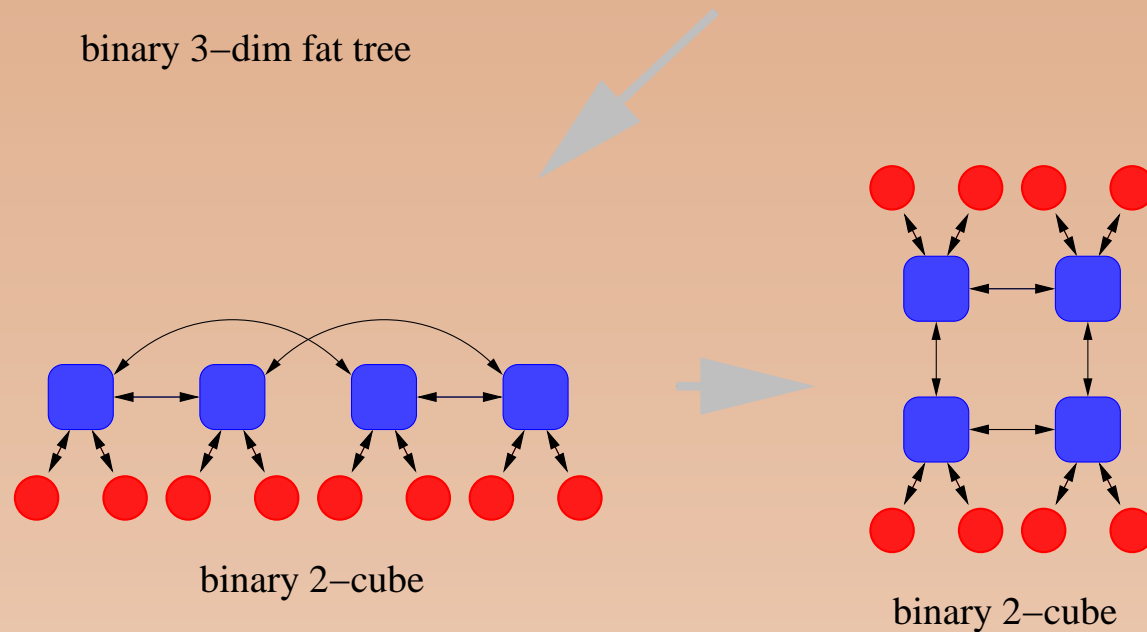
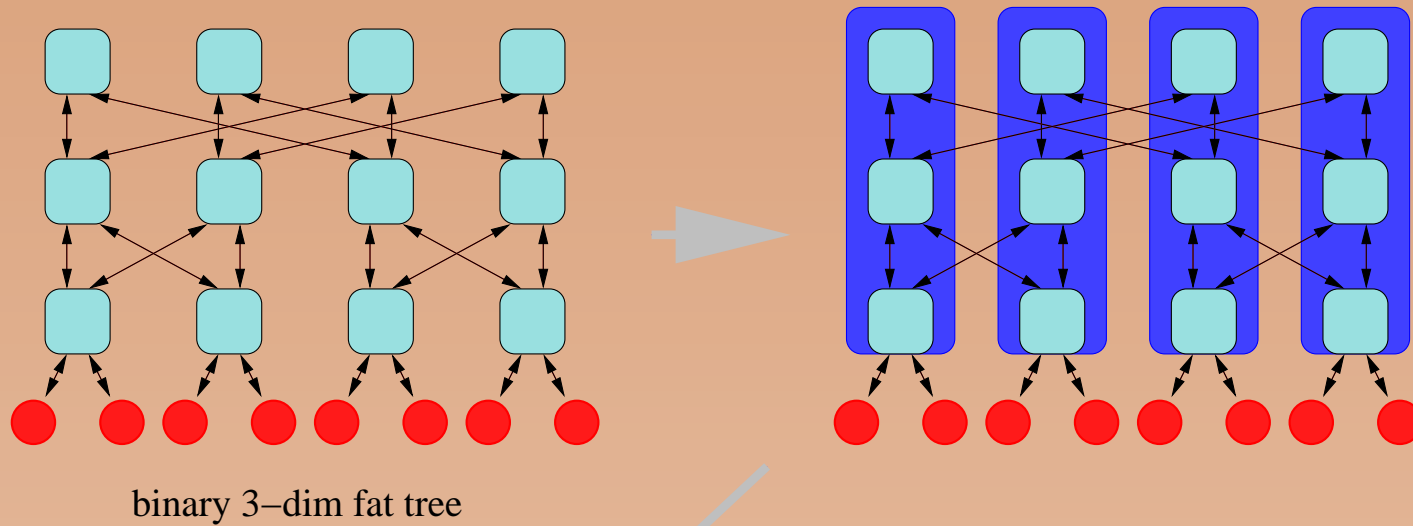


binary 1-cube

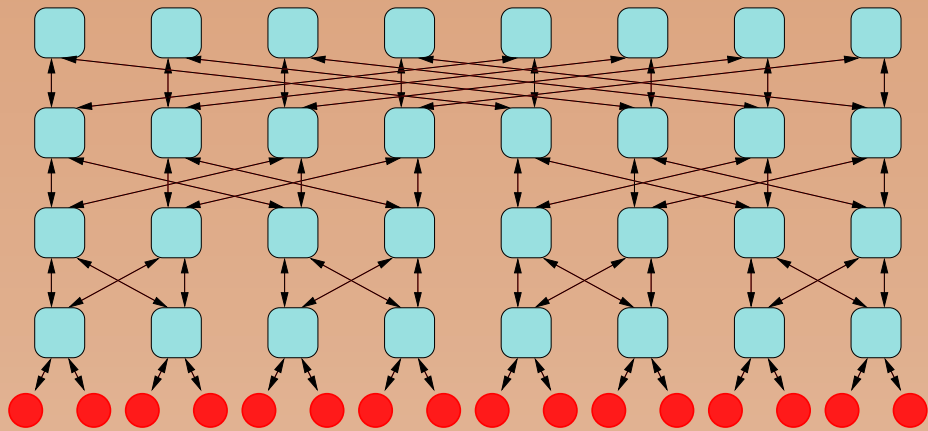




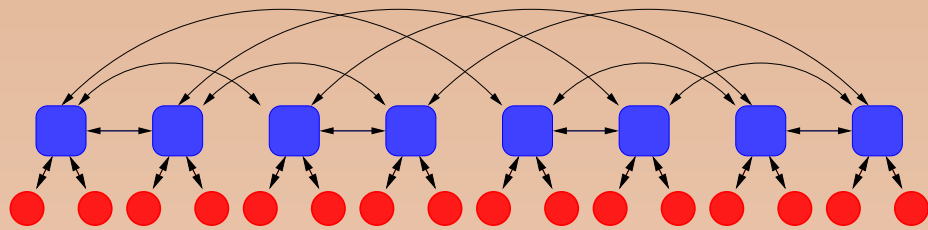
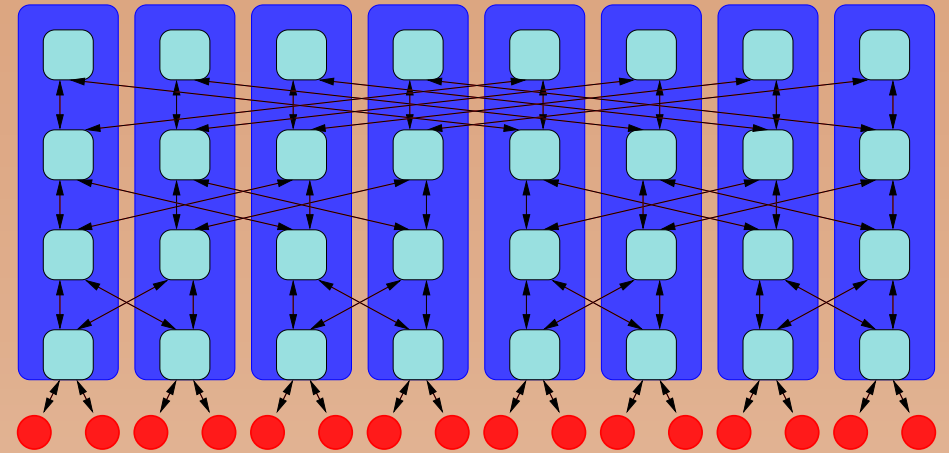
# Relation between Fat Tree and Hypercube - cont'd



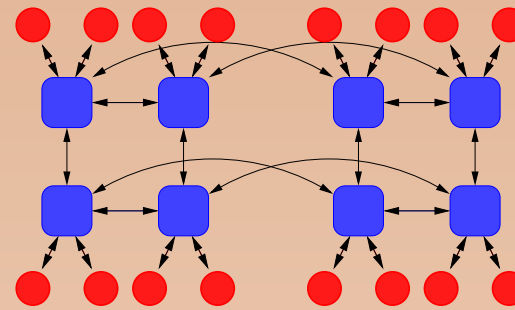
# Relation between Fat Tree and Hypercube - cont'd



binary 4-dim fat tree



binary 3-cube



binary 3-cube



# Trade-offs in Topology Design for the $k$ -ary $n$ -Cube

- Unloaded Latency
- Latency under Load

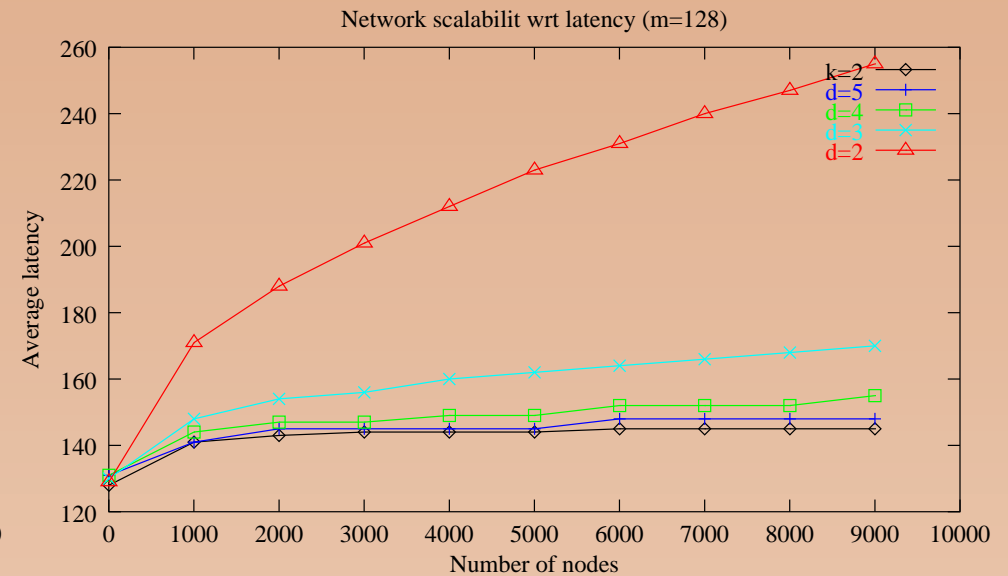
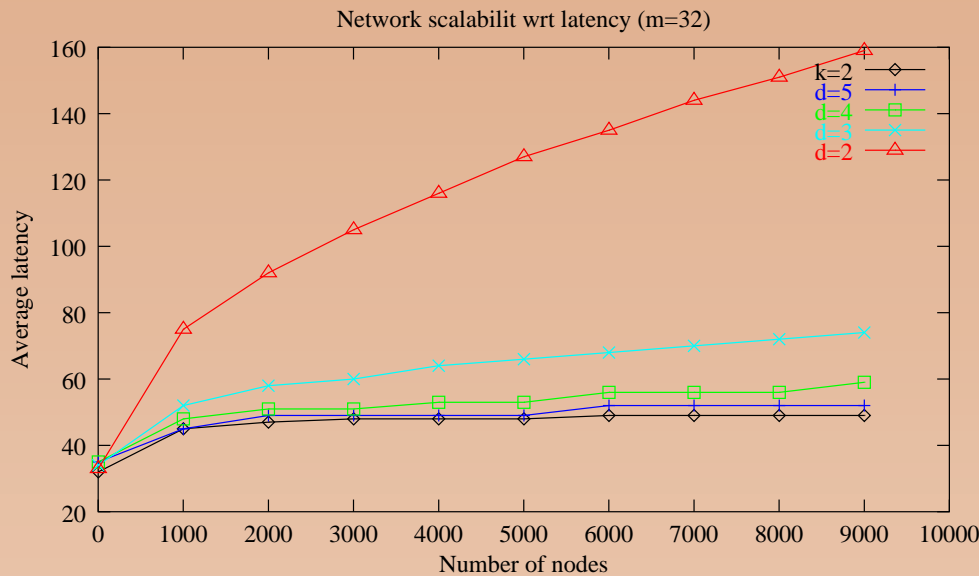


# Network Scaling for Unloaded Latency

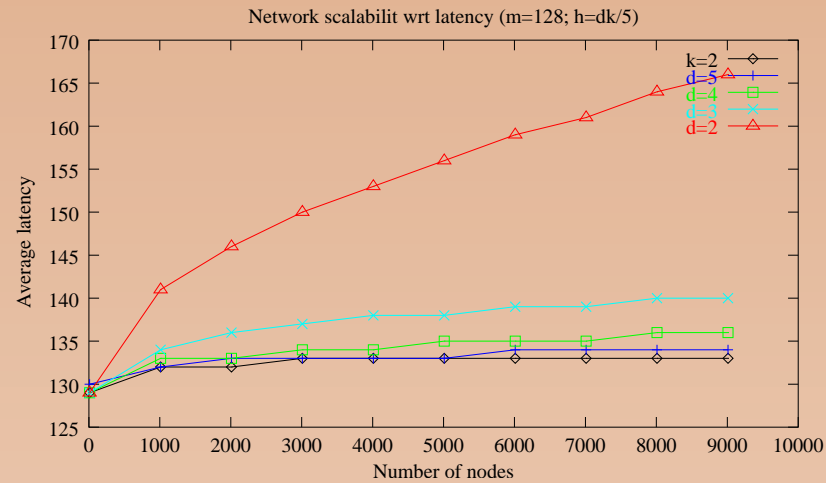
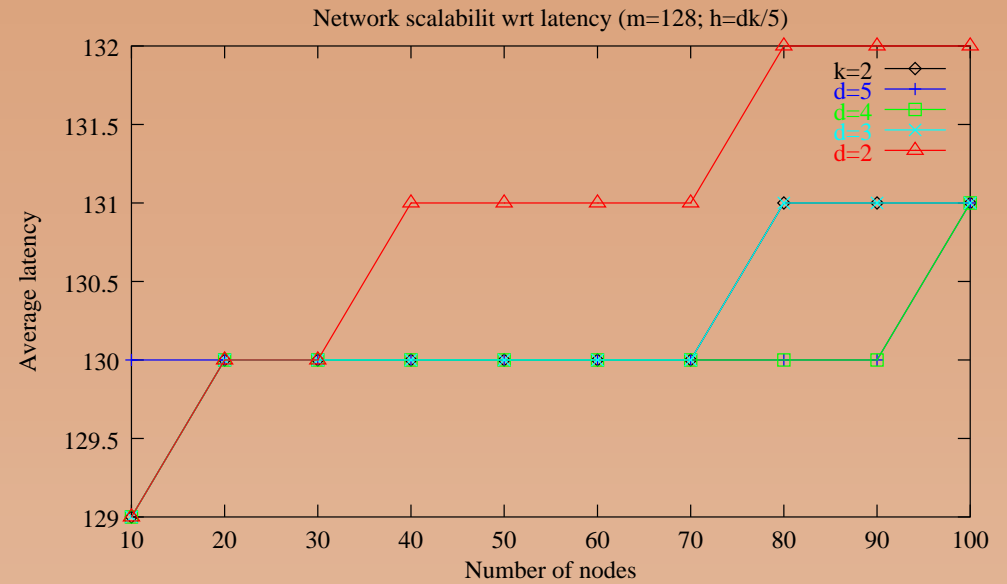
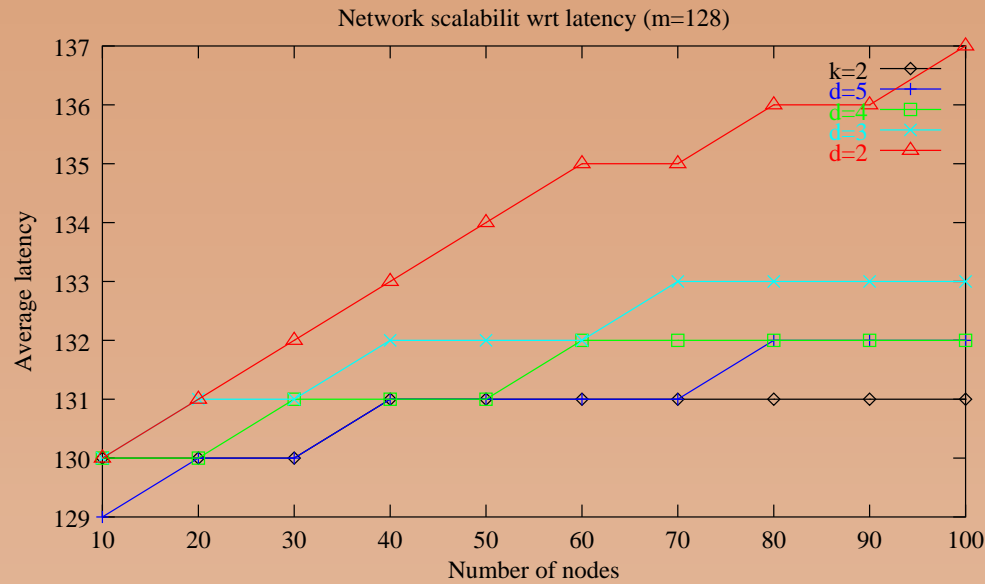
$$\text{Latency}(n) = \text{Admission} + \text{RoutingDelay} + \text{ContentionDelay}$$

$$\text{RoutingDelay } T_{ct}(n, h) = \frac{n}{b} + h\Delta$$

$$\text{RoutingDistance } h = \frac{1}{2}d(k - 1) = \frac{1}{2}(k - 1) \log_k N = \frac{1}{2}(d\sqrt[d]{N} - 1)$$

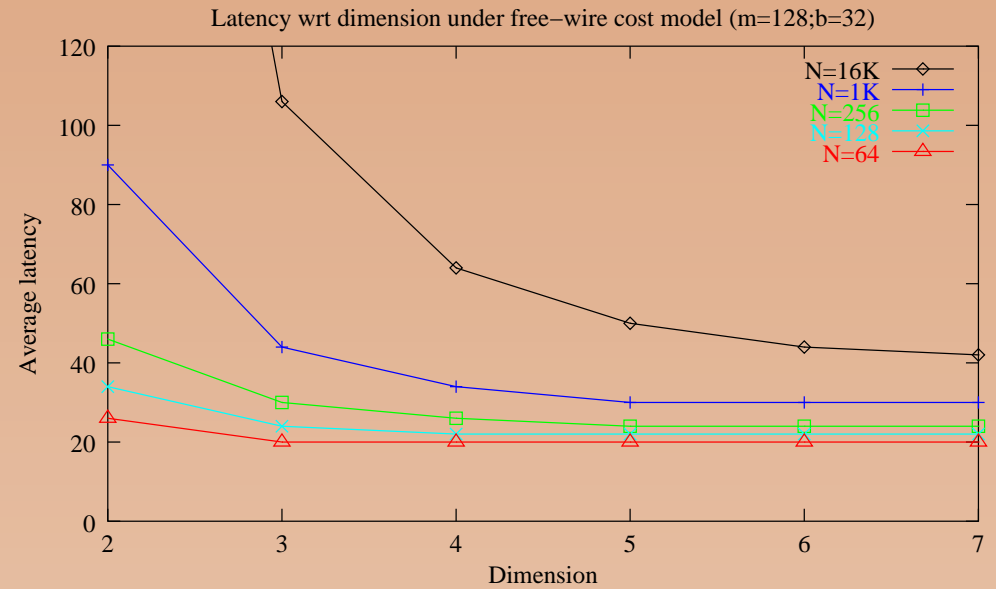
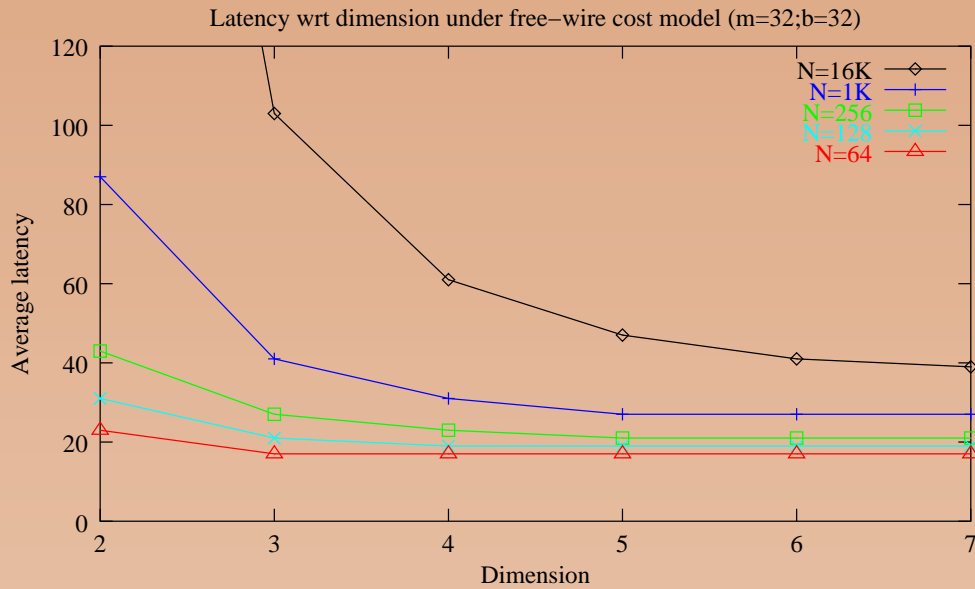


# Unloaded Latency for Small Networks and Local Traffic



# Unloaded Latency under a Free-Wire Cost Model

**Free-wire** cost model: Wires are free and can be added without penalty.

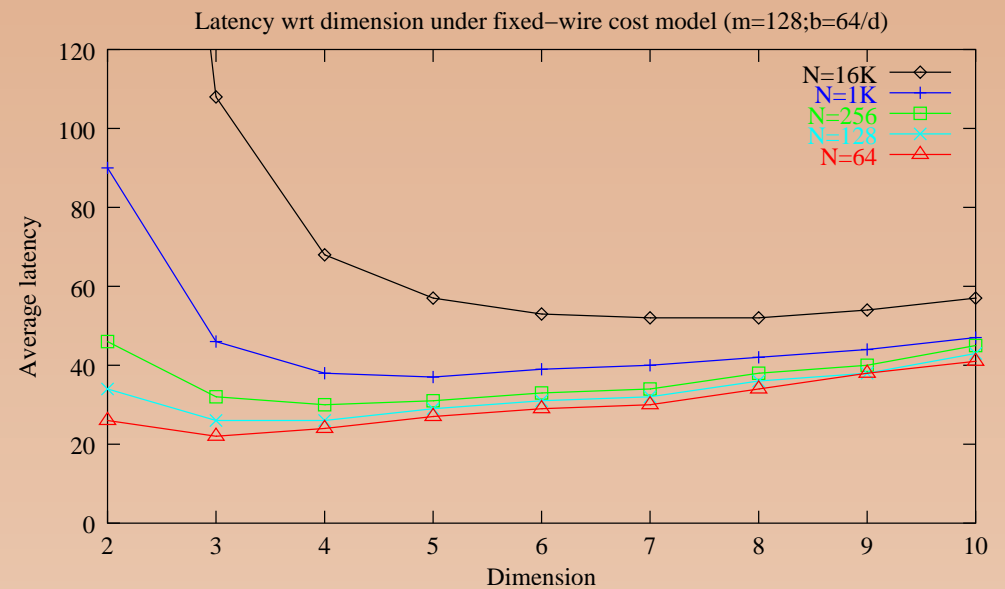
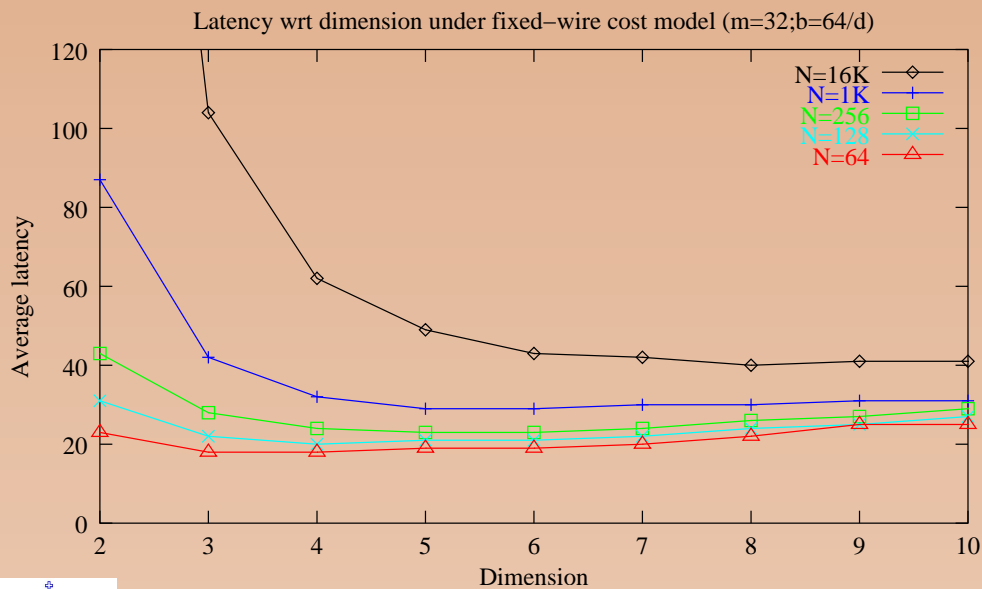


# Unloaded Latency under a Fixed-Wire Cost Models

**Fixed-wire** cost model: The number of wires is constant per node:

128 wires per node:  $w(d) = \lfloor \frac{64}{d} \rfloor$ .

$d$	2	3	4	5	6	7	8	9	10
$w(d)$	32	21	16	12	10	9	8	7	6



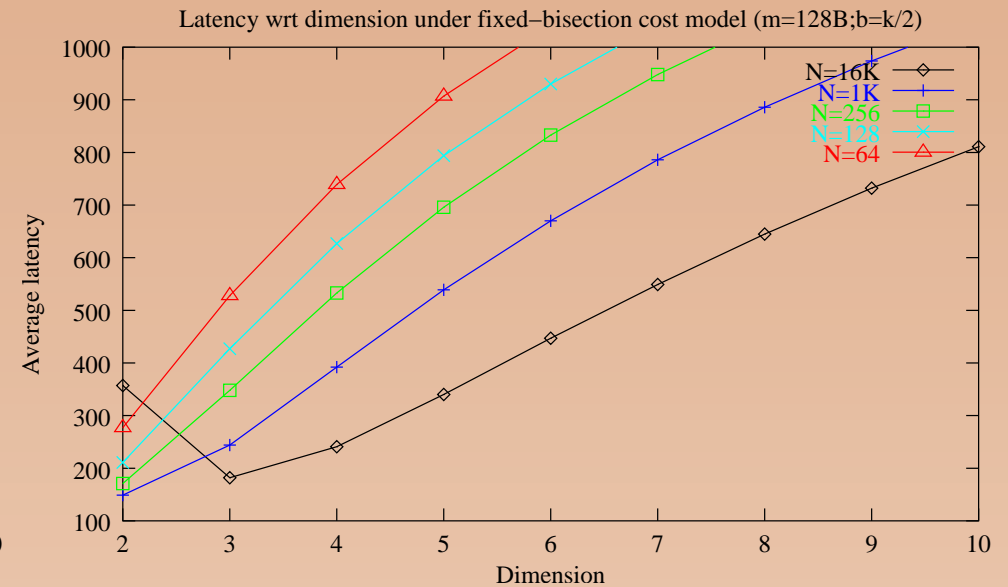
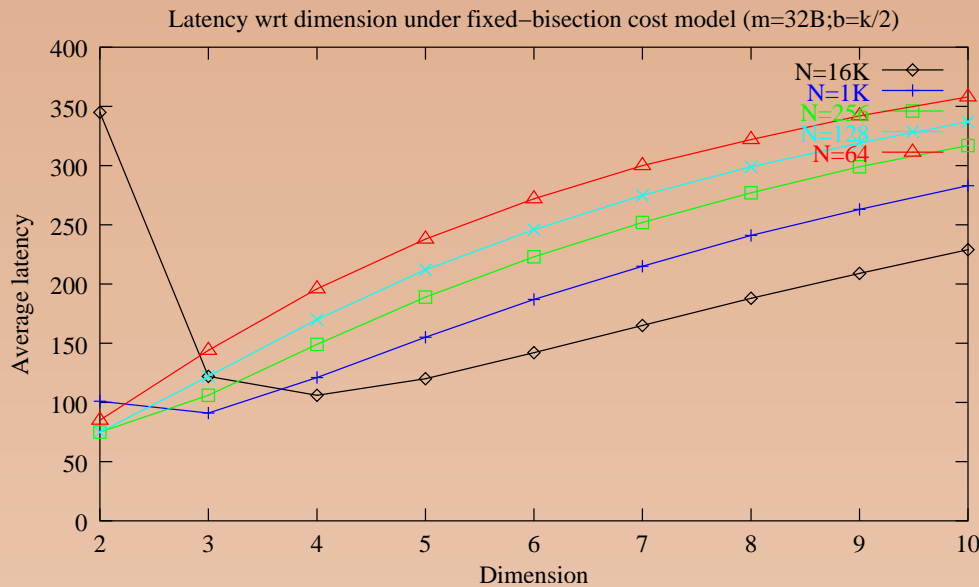
# Unloaded Latency under a Fixed-Bisection Cost Models

**Fixed-bisection** cost model: The number of wires across the bisection is constant:

bisection = 1024 wires:  $w(d) = \frac{k}{2} = \frac{d\sqrt{N}}{2}$ .

Example: N=1024:

$d$	2	3	4	5	6	7	8	9	10
$w(d)$	512	16	5	3	2	2	1	1	1

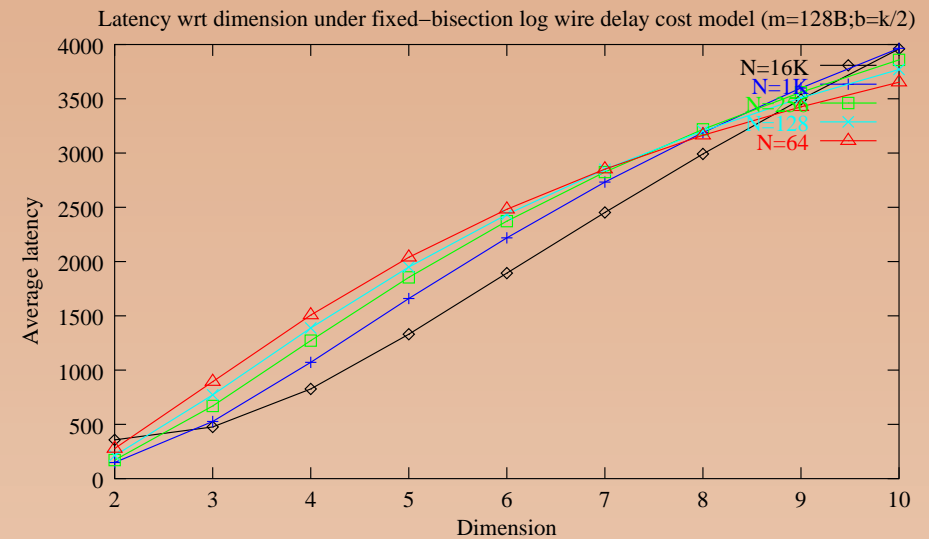
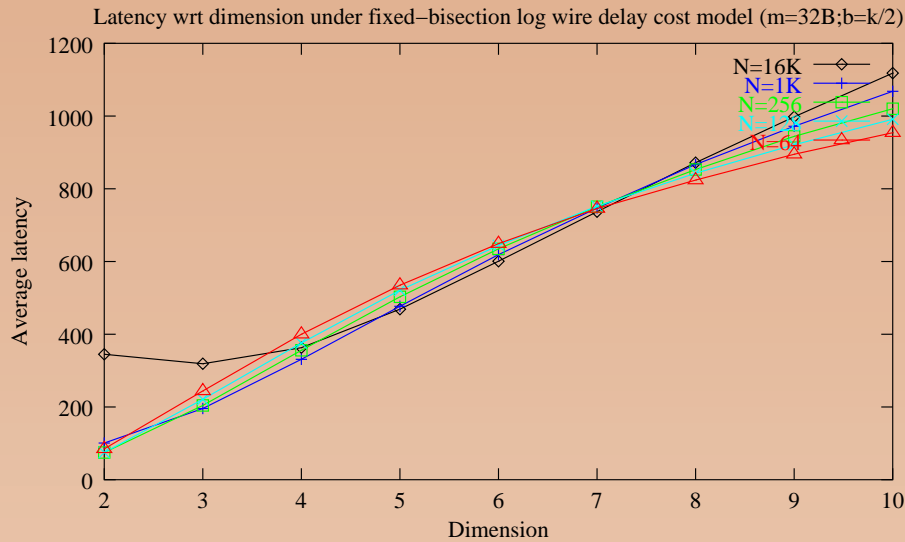




# Unloaded Latency under a Logarithmic Wire Delay Cost Models

**Fixed-bisection Logarithmic Wire Delay** cost model: The number of wires across the bisection is constant and the delay on wires increases logarithmically with the length [Dally, 1990]: Length of long wires:  $l = k^{\frac{n}{2}-1}$

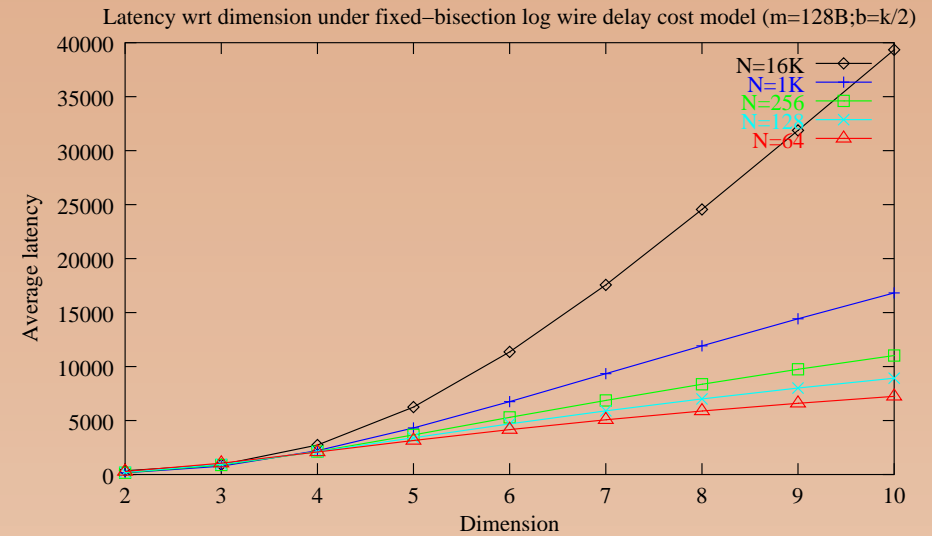
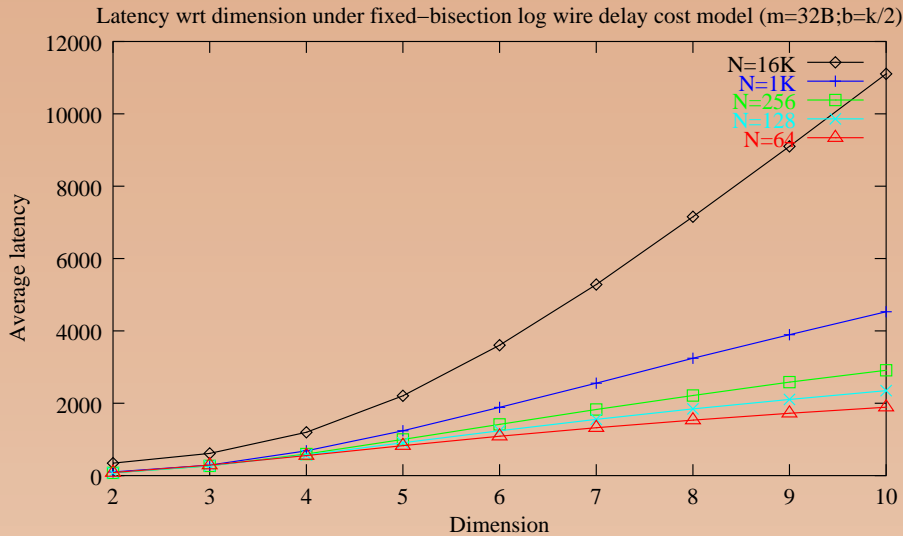
$$T_c \propto 1 + \log l = 1 + \left(\frac{d}{2} - 1\right) \log k$$



# Unloaded Latency under a Linear Wire Delay Cost Models

**Fixed-bisection Linear Wire Delay** cost model: The number of wires across the bisection is constant and the delay on wires increases linearly with the length [Dally, 1990]:  
 Length of long wires:  $l = k^{\frac{n}{2}-1}$

$$T_c \propto l = k^{\frac{d}{2}-1}$$



# Latency under Load

Assumptions [Agarwal, 1991]:

- $k$ -ary  $n$ -cubes
- random traffic
- dimension-order cut-through routing
- unbounded internal buffers (to ignore flow control and deadlock issues)



## Latency under Load - cont'd

$$\text{Latency}(n) = \text{Admission} + \text{RoutingDelay} + \text{ContentionDelay}$$

$$T(m, k, d, w, \rho) = \text{RoutingDelay} + \text{ContentionDelay}$$

$$T(m, k, d, w, \rho) = \frac{m}{w} + dh_k(\Delta + W(m, k, d, w, \rho))$$

$$W(m, k, d, w, \rho) = \frac{m}{w} \cdot \frac{\rho}{(1 - \rho)} \cdot \frac{h_k - 1}{h_k^2} \cdot \left(1 + \frac{1}{d}\right)$$

$$h = \frac{1}{2}d(k - 1)$$

$m$  ... message size

$w$  ... bitwidth of link

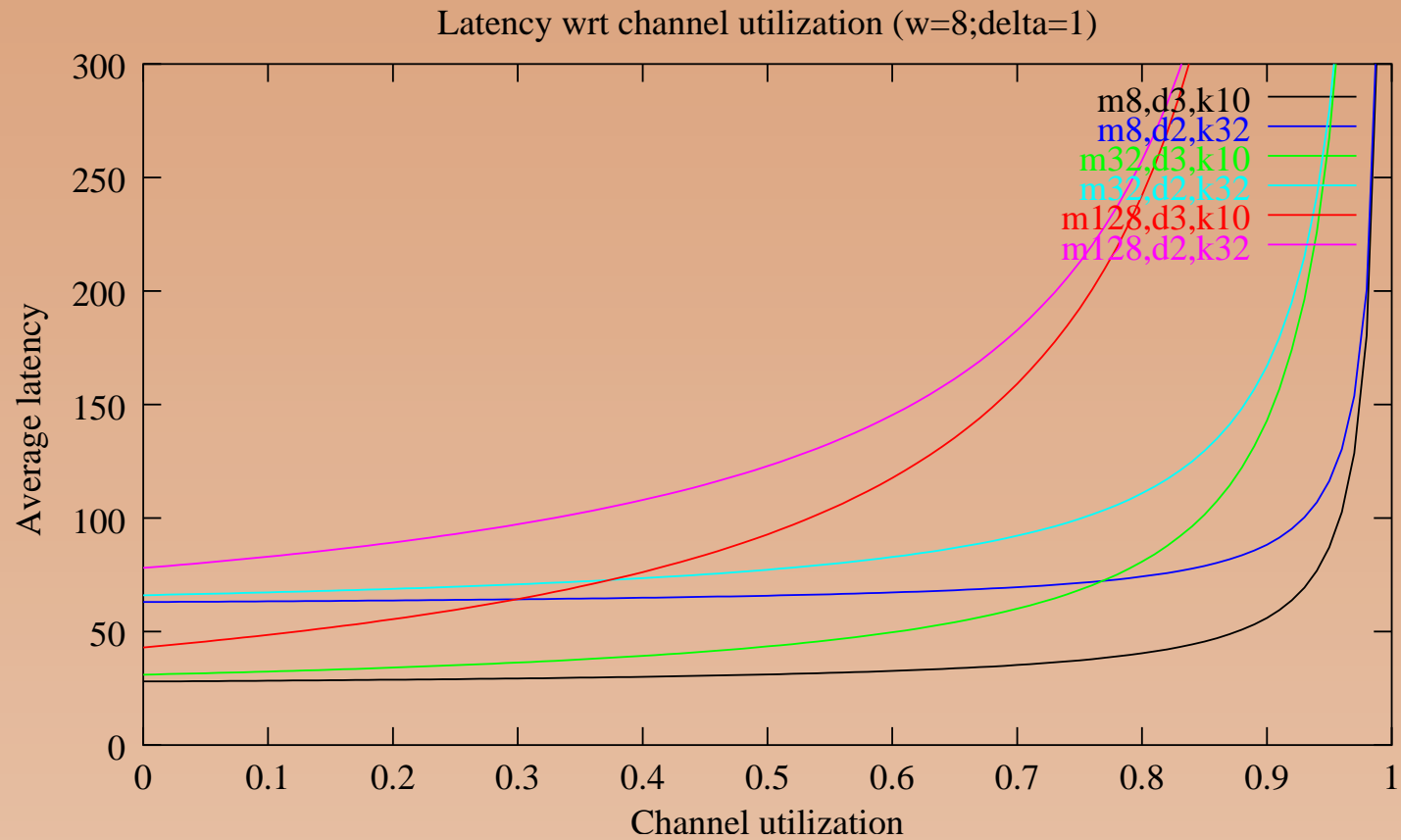
$\rho$  ... aggregate channel utilization

$h_k$  ... average distance in each dimension

$\Delta$  ... switching time in cycles



# Latency vs Channel Load



# Routing

**Deterministic routing** The route is determined solely by source and destination locations.

**Arithmetic routing** The destination address of the incoming packet is compared with the address of the switch and the packet is routed accordingly. (relative or absolute addresses)

**Source based routing** The source determines the route and builds a header with one directive for each switch. The switches strip off the top directive.

**Table-driven routing** Switches have routing tables, which can be configured.

**Adaptive routing** The route can be adapted by the switches to balance the load.

**Minimal routing** allows only shortest paths while non-minimal routing allows even longer paths.



# Quality of Service

- Best Effort (BE)
  - ★ Optimization of the average case
  - ★ Loose or non-existent worst case bounds
  - ★ Cost effective use of resources
- Guaranteed Service (GS)
  - ★ Maximum delay
  - ★ Minimum bandwidth
  - ★ Maximum Jitter
  - ★ Requires additional resources



## Regulated Flows

A Flow  $F$  is  $(\sigma, \rho)$  regulated if

$$F(b) - F(a) \leq \sigma + \rho(b - a)$$

for all time intervals  $[a, b]$ ,  $0 \leq a \leq b$  and where

$F(t) \dots$  the cumulative amount of traffic between 0 and  $t \geq 0$ .

$\sigma \geq 0$  is the burstiness constraint;

$\rho \geq 0$  is the maximum average rate;





## Regulated Flows

A Flow  $F$  is  $(\sigma, \rho)$  regulated if

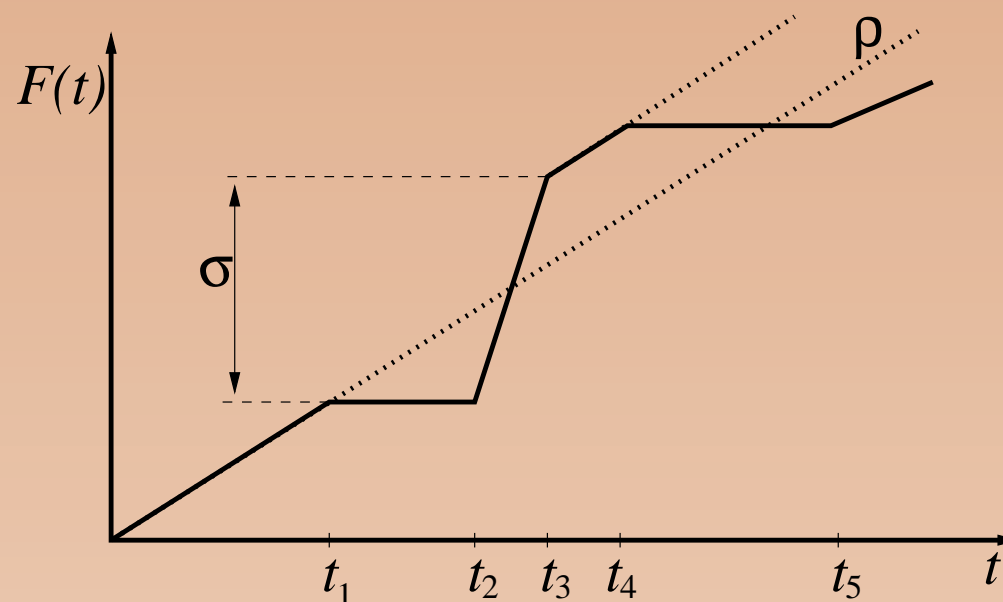
$$F(b) - F(a) \leq \sigma + \rho(b - a)$$

for all time intervals  $[a, b]$ ,  $0 \leq a \leq b$  and where

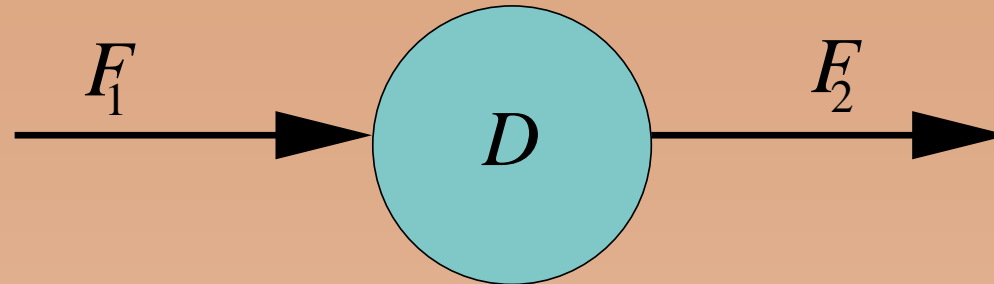
$F(t) \dots$  the cumulative amount of traffic between 0 and  $t \geq 0$ .

$\sigma \geq 0$  is the burstiness constraint;

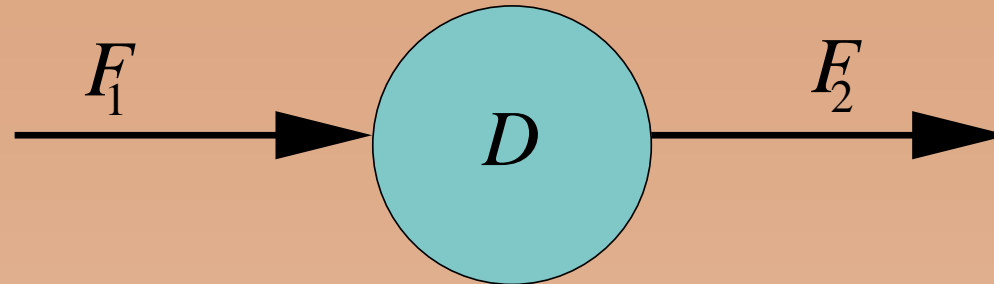
$\rho \geq 0$  is the maximum average rate;



## Regulated Flows - Delay Element



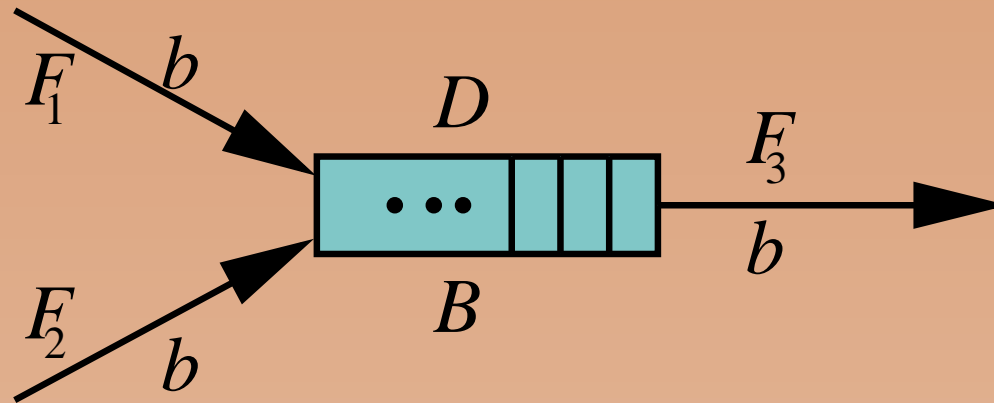
## Regulated Flows - Delay Element



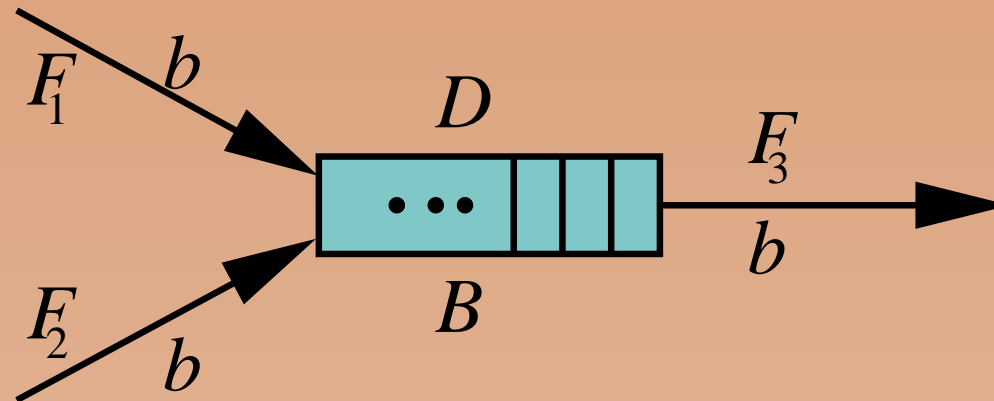
$$F_1 \sim (\sigma, \rho)$$

$$F_2 \sim (\sigma + \rho D, \rho)$$

## Regulated Flows - Work Conserving Multiplexer



## Regulated Flows - Work Conserving Multiplexer



$$F_1 \sim (\sigma_1, \rho_1)$$

$$F_2 \sim (\sigma_2, \rho_2)$$

$$\text{link bandwidth } b < \rho_1 + \rho_2$$

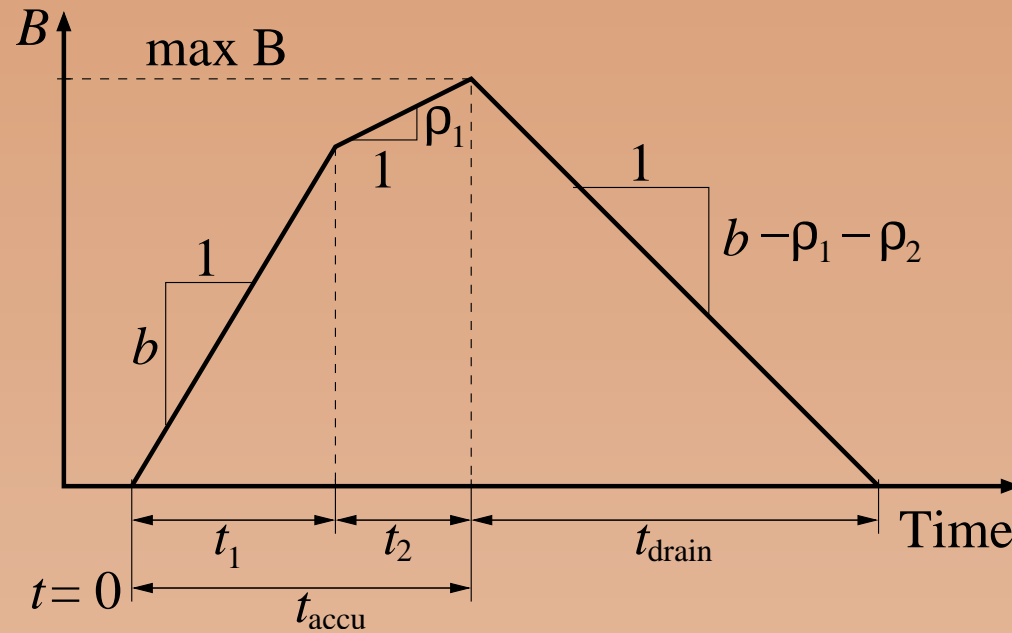
$$F_3 \sim ?$$

$$\text{maximum delay } D = ?$$

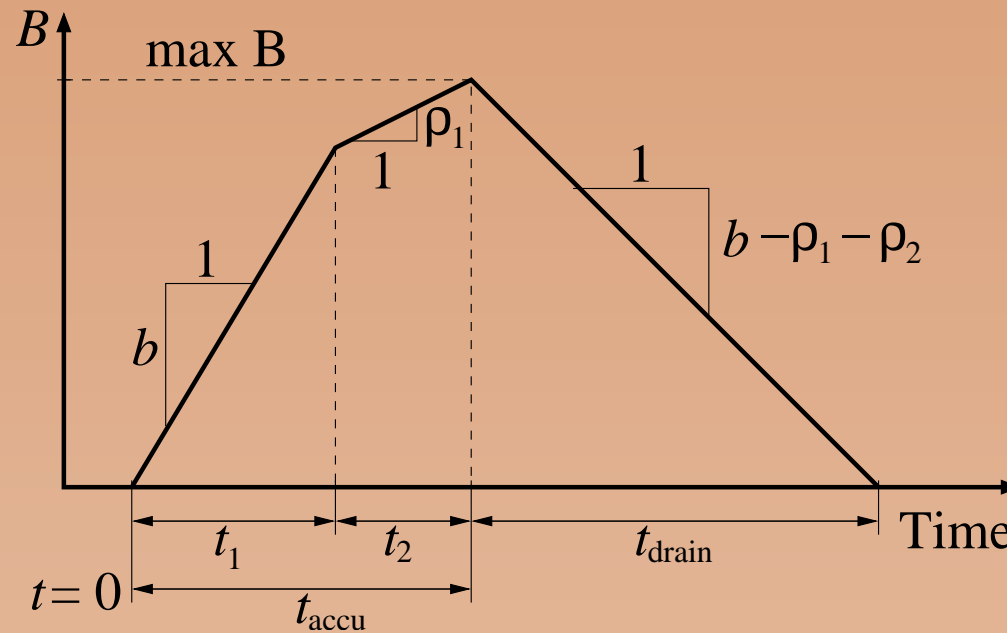
$$\text{maximum backlog } B = ?$$



# Work Conserving Multiplexer - 1

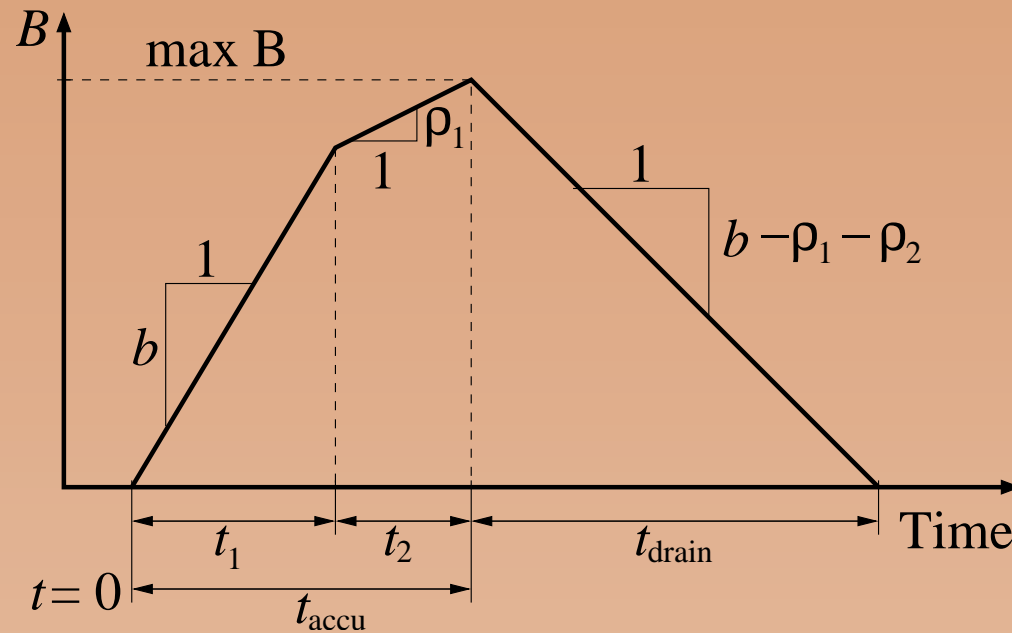


## Work Conserving Multiplexer - 1



**Phase 1** ( $t_1$ ):  $F_1$  and  $F_2$  transmit at full speed;

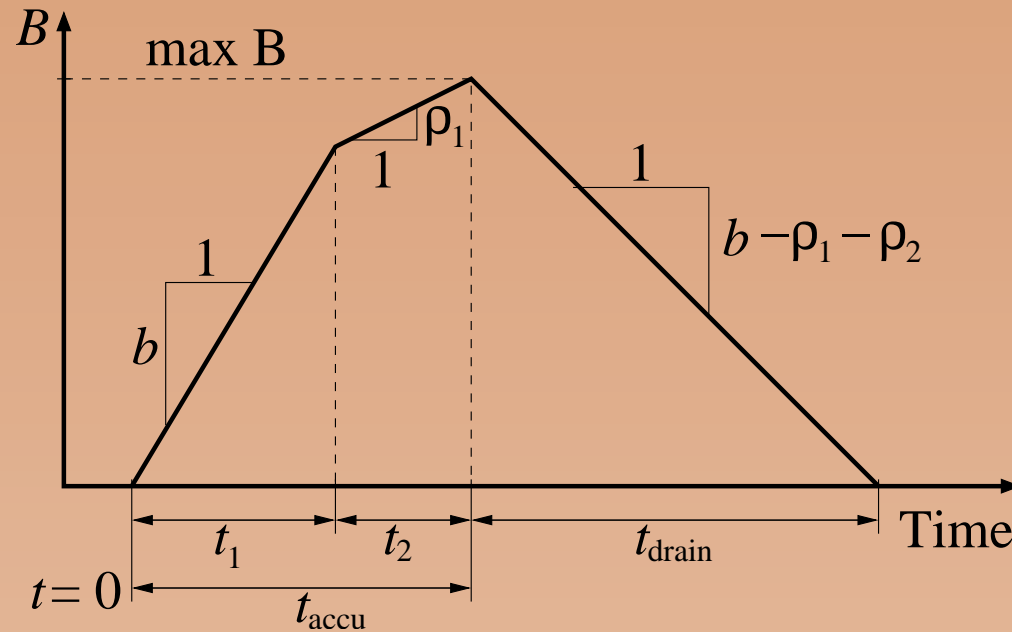
## Work Conserving Multiplexer - 1



**Phase 1** ( $t_1$ ):  $F_1$  and  $F_2$  transmit at full speed;  
 Assume: At  $t = 0$  the queue is empty;  $\sigma_1 \leq \sigma_2$

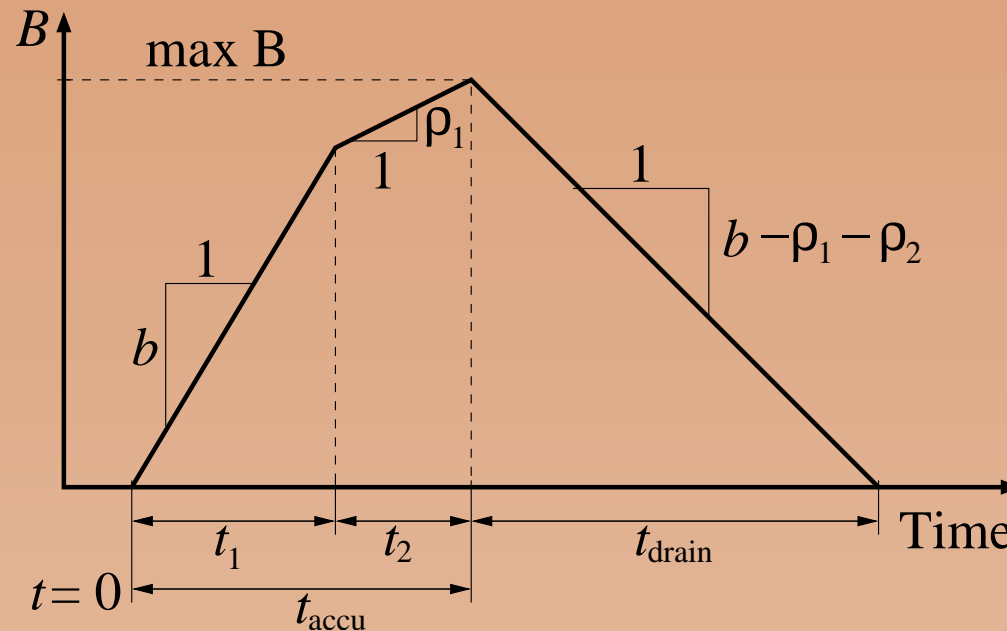


## Work Conserving Multiplexer - 1



**Phase 1** ( $t_1$ ):  $F_1$  and  $F_2$  transmit at full speed;  
 Assume: At  $t = 0$  the queue is empty;  $\sigma_1 \leq \sigma_2$   
 Injection rate:  $2b$ ; Drain rate:  $b$

## Work Conserving Multiplexer - 1



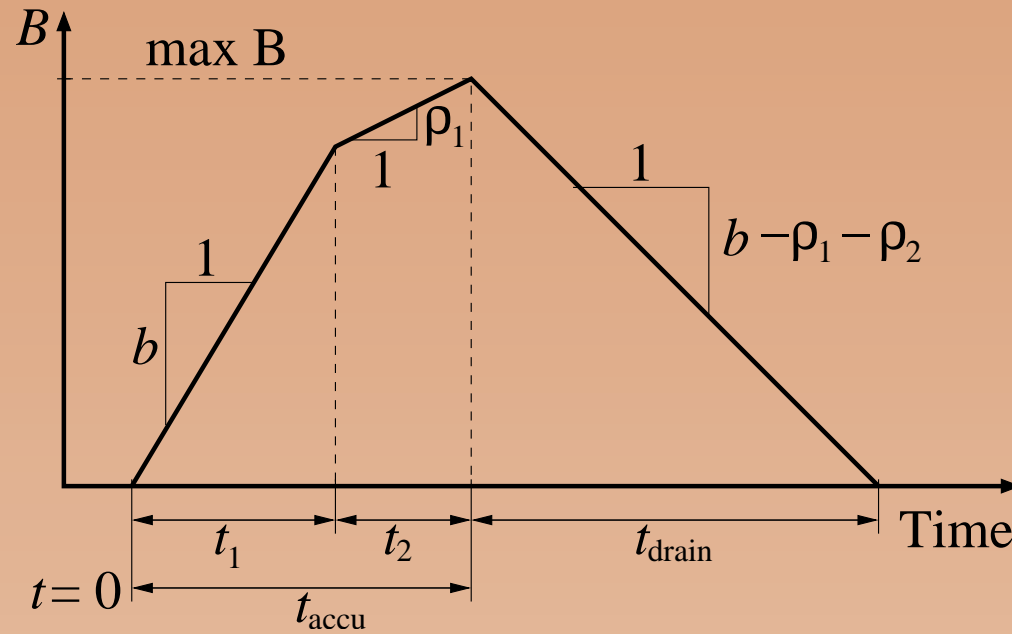
**Phase 1** ( $t_1$ ):  $F_1$  and  $F_2$  transmit at full speed;  
 Assume: At  $t = 0$  the queue is empty;  $\sigma_1 \leq \sigma_2$   
 Injection rate:  $2b$ ; Drain rate:  $b$

$$bt_1 = \sigma_1 + \rho_1 t_1$$

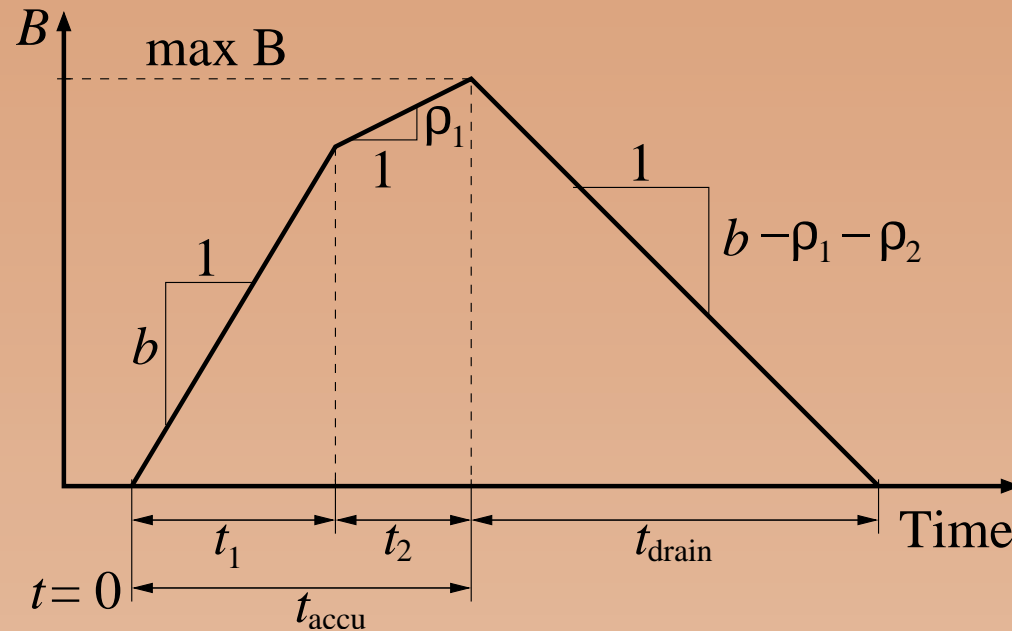
$$t_1 = \frac{\sigma_1}{b - \rho_1}$$



## Work Conserving Multiplexer - 2

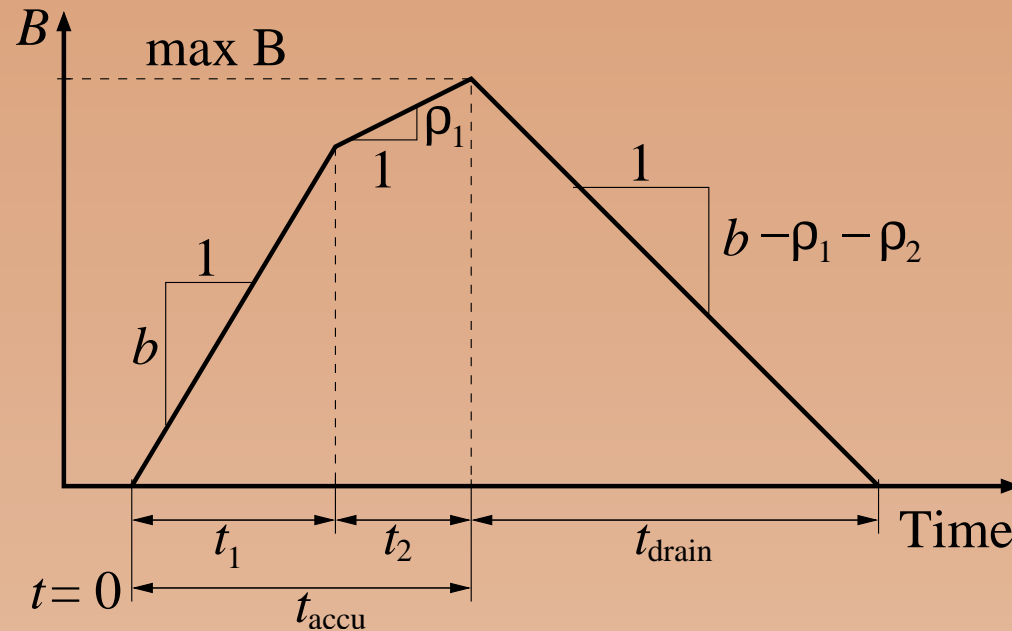


## Work Conserving Multiplexer - 2



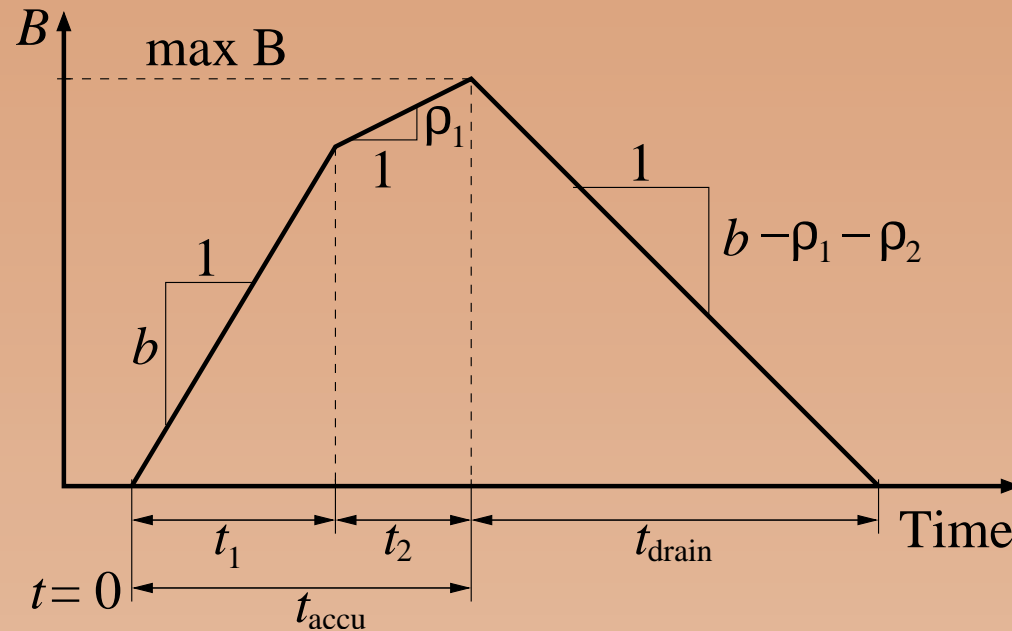
**Phase 2** ( $t_2$ ):  $F_1$  transmits at rate  $\rho_1$ ,  $F_2$  transmits at full speed;

## Work Conserving Multiplexer - 2



**Phase 2** ( $t_2$ ):  $F_1$  transmits at rate  $\rho_1$ ,  $F_2$  transmits at full speed;  
 Injection rate:  $b + \rho_1$ ; Drain rate:  $b$

## Work Conserving Multiplexer - 2



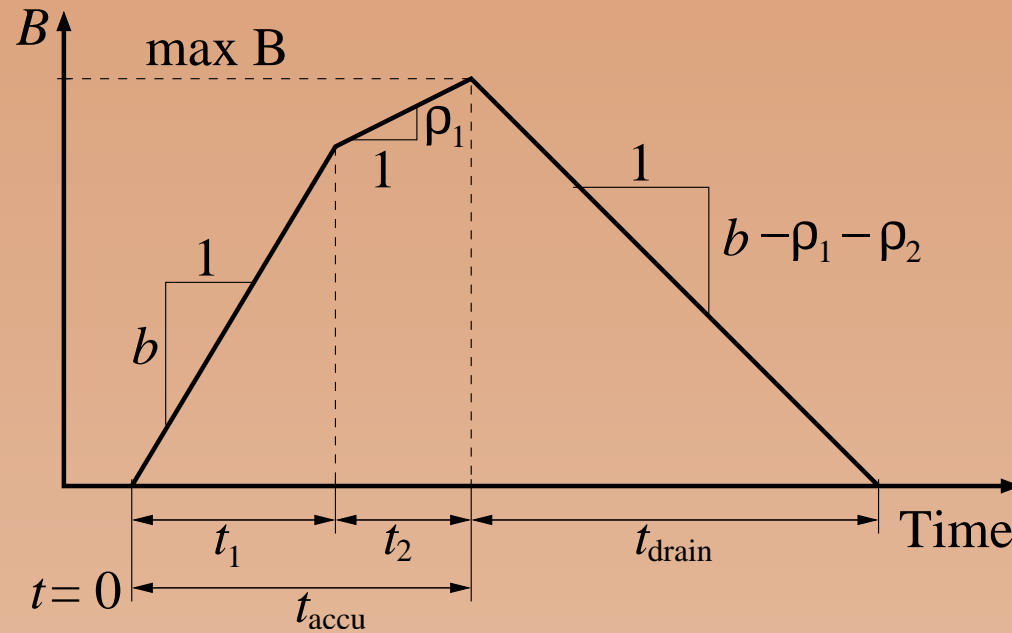
**Phase 2** ( $t_2$ ):  $F_1$  transmits at rate  $\rho_1$ ,  $F_2$  transmits at full speed;  
 Injection rate:  $b + \rho_1$ ; Drain rate:  $b$

$$bt_{\text{accu}} = \sigma_2 + \rho_2 t_{\text{accu}}$$

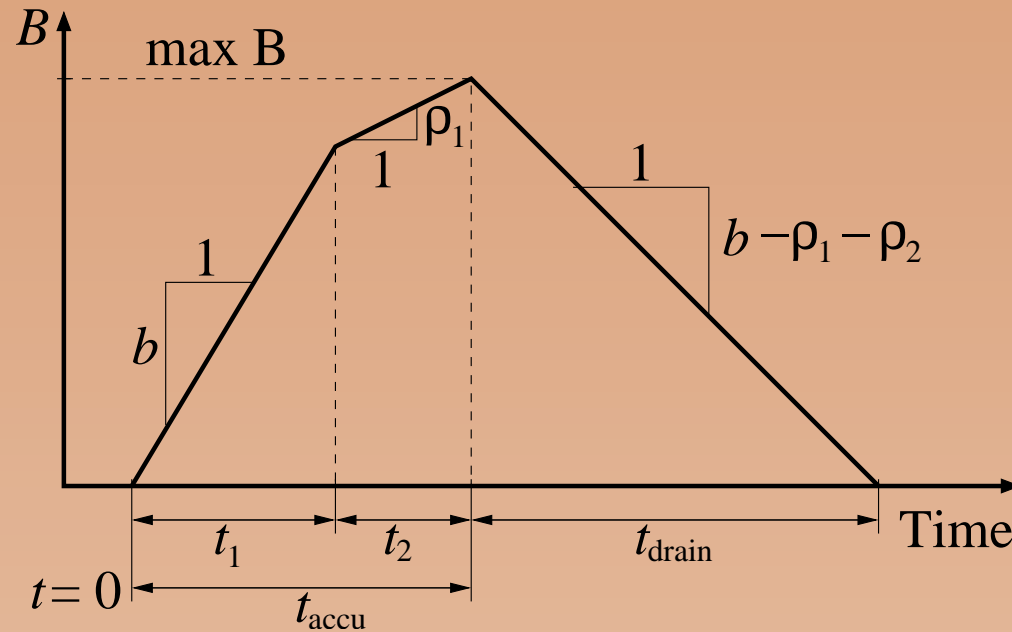
$$t_{\text{accu}} = \frac{\sigma_2}{b - \rho_2}$$



## Work Conserving Multiplexer - 3



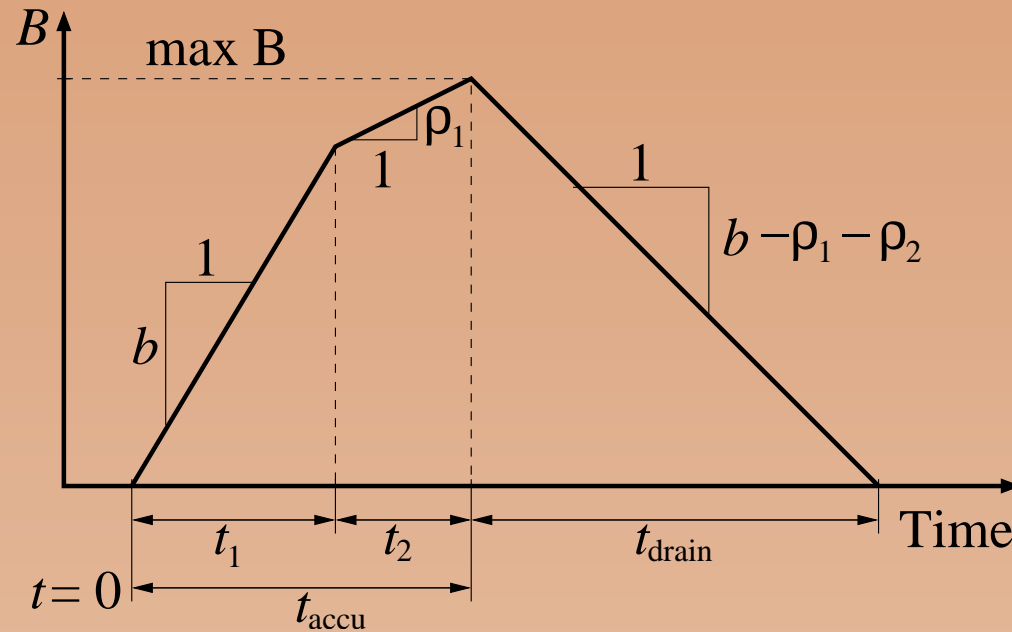
## Work Conserving Multiplexer - 3



**Phase 3** ( $t_{\text{drain}}$ ):  $F_1$  transmits at rate  $\rho_1$ ,  $F_2$  transmits at rate  $\rho_2$ ;

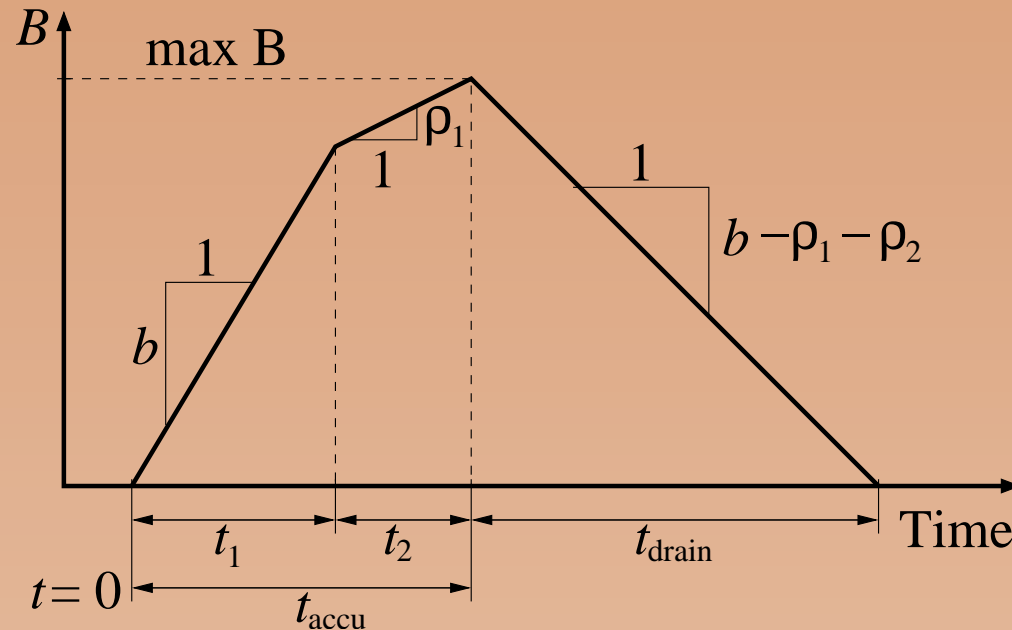


## Work Conserving Multiplexer - 3



**Phase 3** ( $t_{\text{drain}}$ ):  $F_1$  transmits at rate  $\rho_1$ ,  $F_2$  transmits at rate  $\rho_2$ ;  
 Injection rate:  $\rho_1 + \rho_2$ ; Drain rate:  $b$

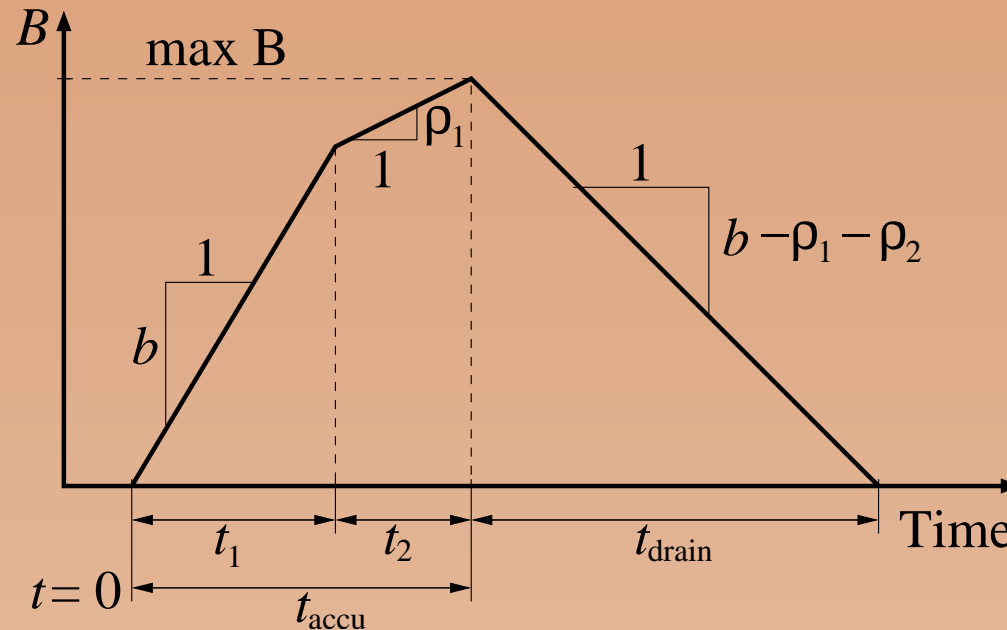
## Work Conserving Multiplexer - 3



**Phase 3** ( $t_{\text{drain}}$ ):  $F_1$  transmits at rate  $\rho_1$ ,  $F_2$  transmits at rate  $\rho_2$ ;  
 Injection rate:  $\rho_1 + \rho_2$ ; Drain rate:  $b$

$$t_{\text{drain}} = \frac{B_{\text{max}}}{b - \rho_1 - \rho_2}$$

## Work Conserving Multiplexer - 3

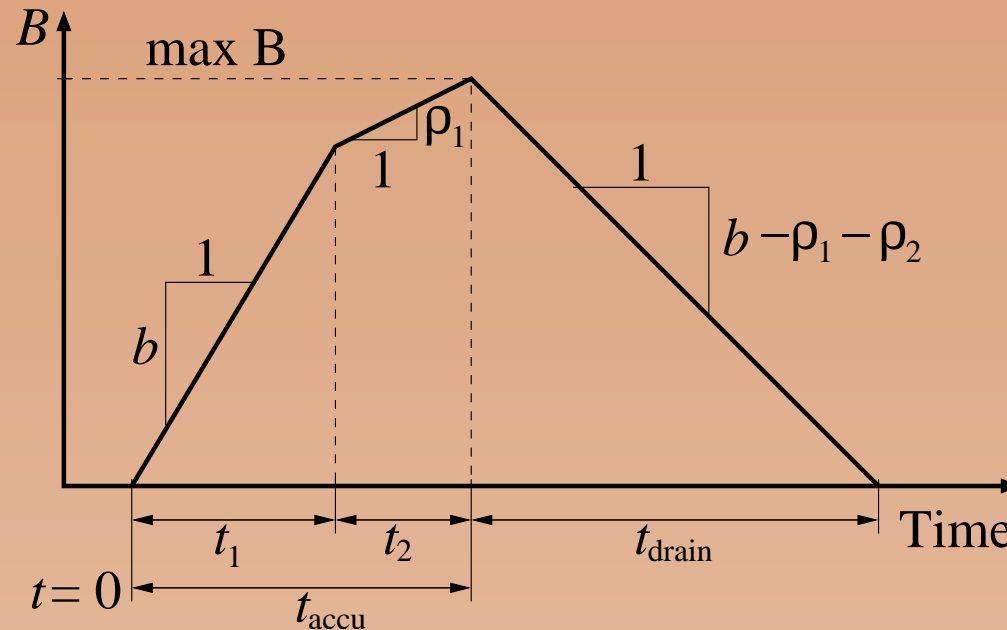


**Phase 3** ( $t_{\text{drain}}$ ):  $F_1$  transmits at rate  $\rho_1$ ,  $F_2$  transmits at rate  $\rho_2$ ;  
 Injection rate:  $\rho_1 + \rho_2$ ; Drain rate:  $b$

$$t_{\text{drain}} = \frac{B_{\text{max}}}{b - \rho_1 - \rho_2}$$

$$B_{\text{max}} = bt_1 + \rho_1 t_2$$

## Work Conserving Multiplexer - 3



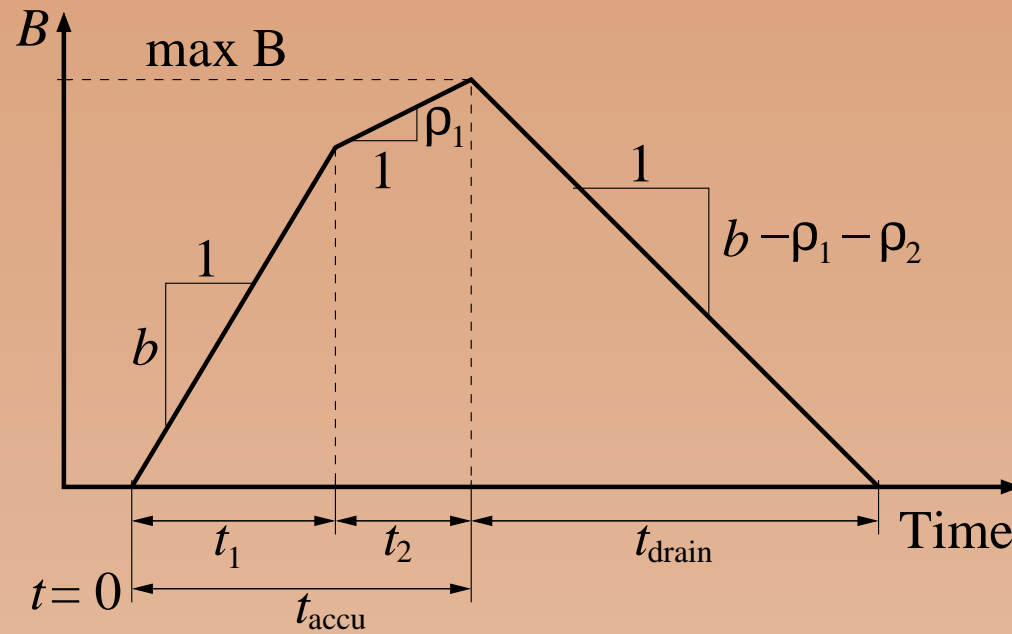
**Phase 3** ( $t_{\text{drain}}$ ):  $F_1$  transmits at rate  $\rho_1$ ,  $F_2$  transmits at rate  $\rho_2$ ;  
Injection rate:  $\rho_1 + \rho_2$ ; Drain rate:  $b$

$$t_{\text{drain}} = \frac{B_{\text{max}}}{b - \rho_1 - \rho_2}$$

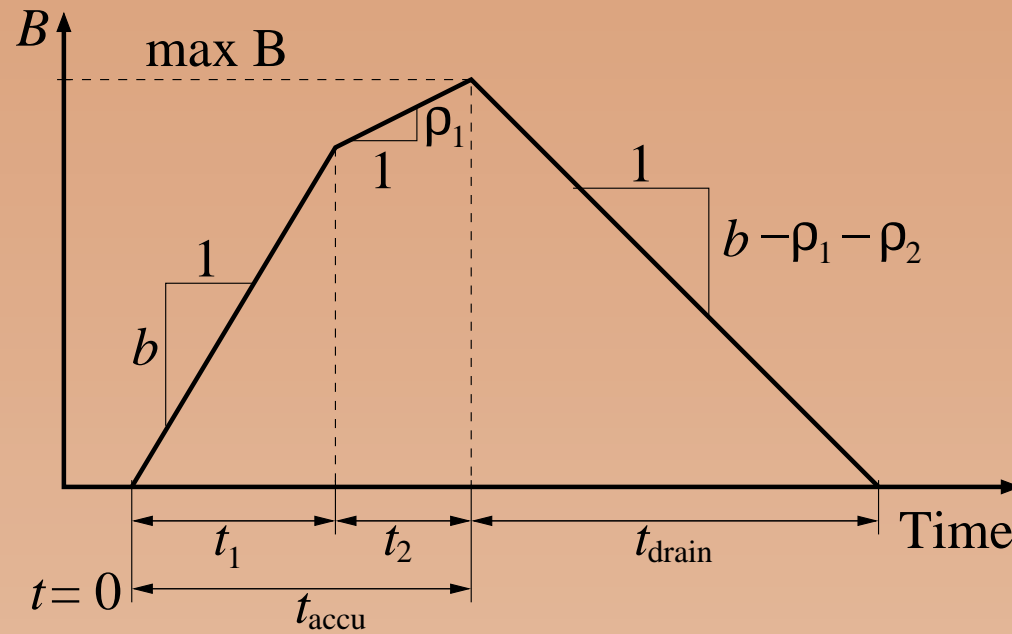
$$B_{\text{max}} = bt_1 + \rho_1 t_2 = \sigma_1 + \frac{\rho_1 \sigma_2}{b - \rho_2}$$



## Work Conserving Multiplexer - Summary

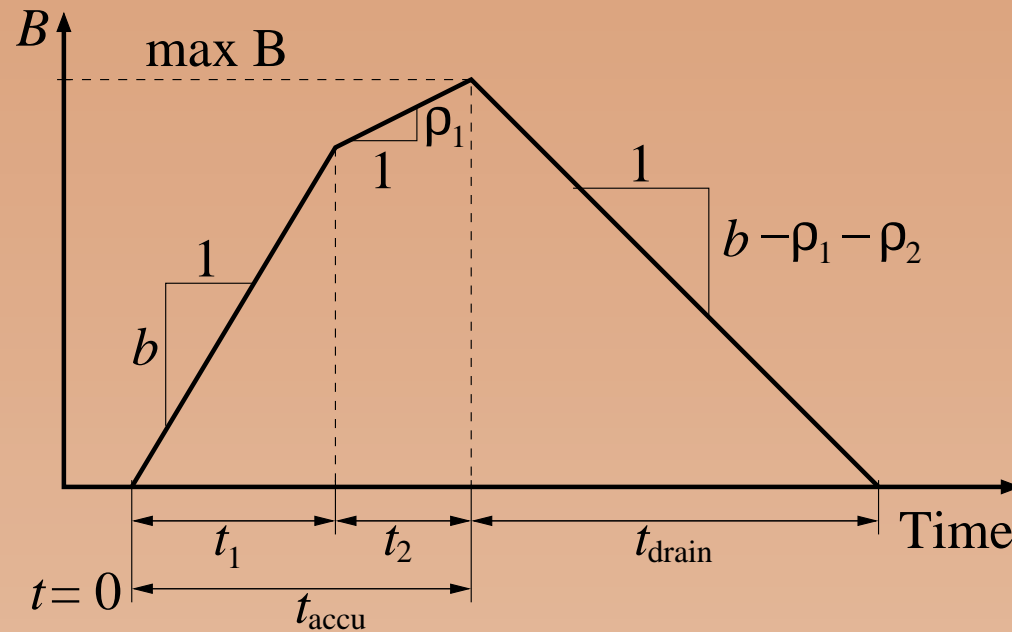


## Work Conserving Multiplexer - Summary



$$B_{\max} = \sigma_1 + \frac{\rho_1 \sigma_2}{b - \rho_2}$$

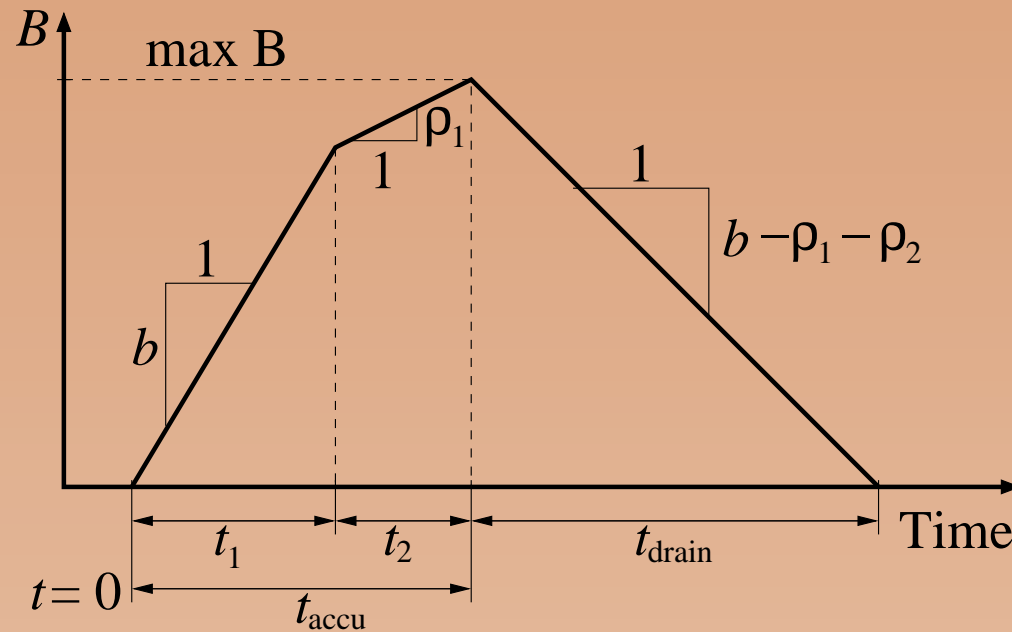
## Work Conserving Multiplexer - Summary



$$B_{\max} = \sigma_1 + \frac{\rho_1 \sigma_2}{b - \rho_2}$$

$$D_{\max} = t_{\text{accu}} + t_{\text{drain}} = \frac{\sigma_1 + \sigma_2}{b - \rho_1 - \rho_2}$$

## Work Conserving Multiplexer - Summary



$$B_{\max} = \sigma_1 + \frac{\rho_1 \sigma_2}{b - \rho_2}$$

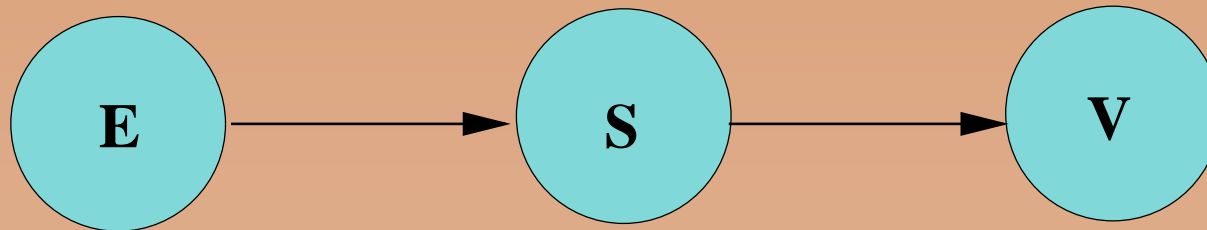
$$D_{\max} = t_{\text{accu}} + t_{\text{drain}} = \frac{\sigma_1 + \sigma_2}{b - \rho_1 - \rho_2}$$

$$F_3 \sim (\sigma_1 + \sigma_2, \rho_1 + \rho_2)$$

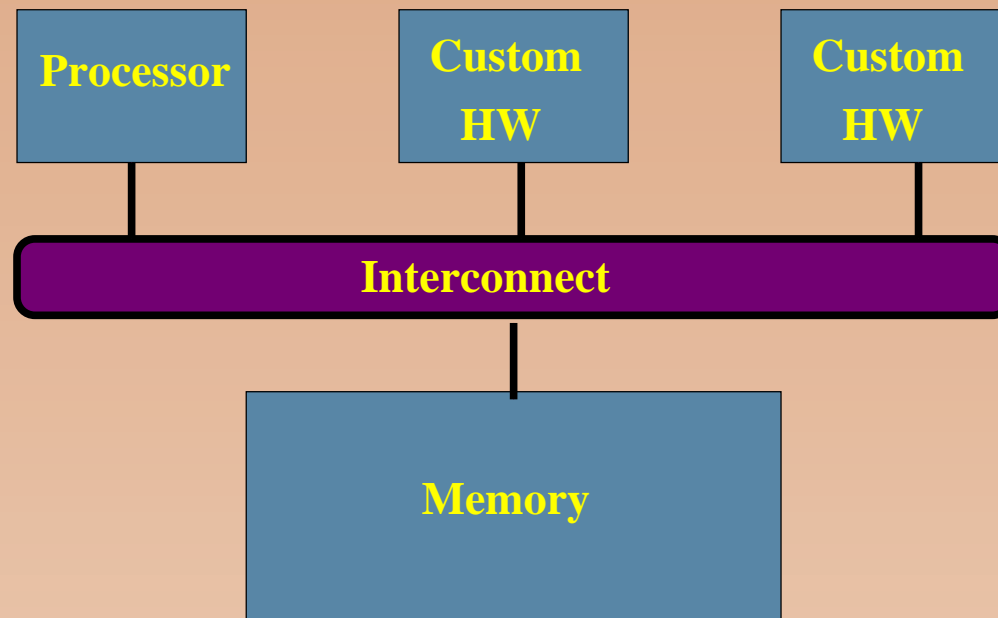
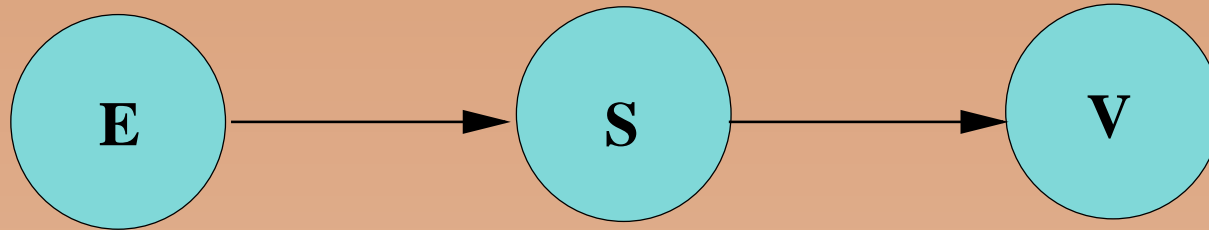




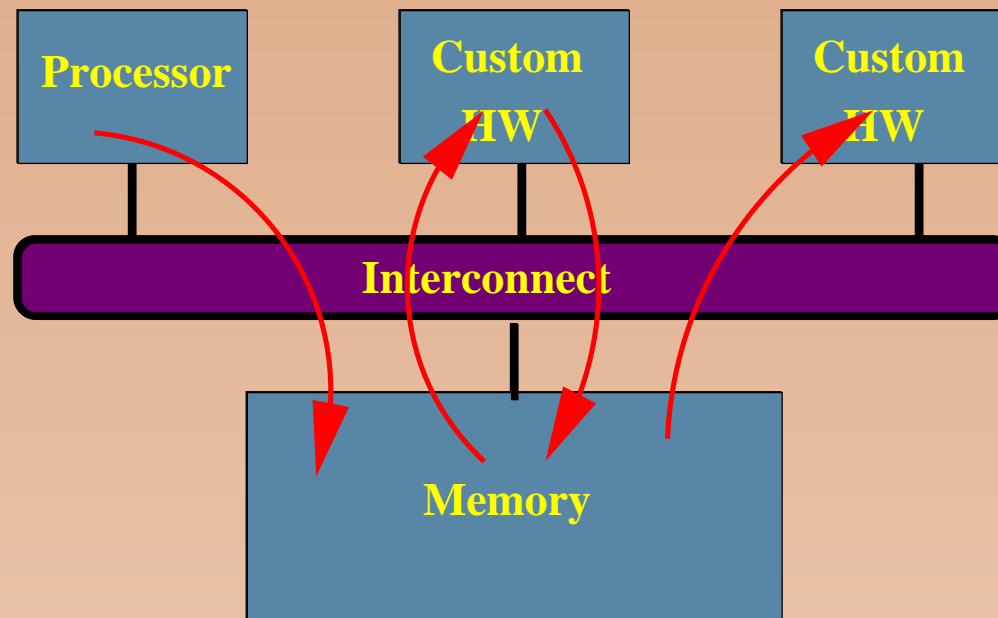
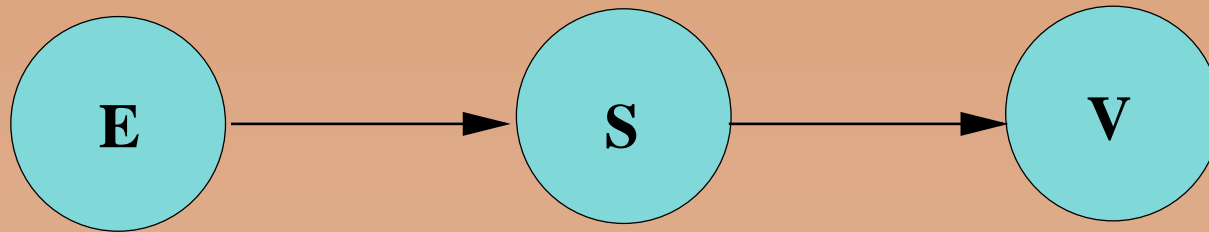
# MPEG Encoding Case Study



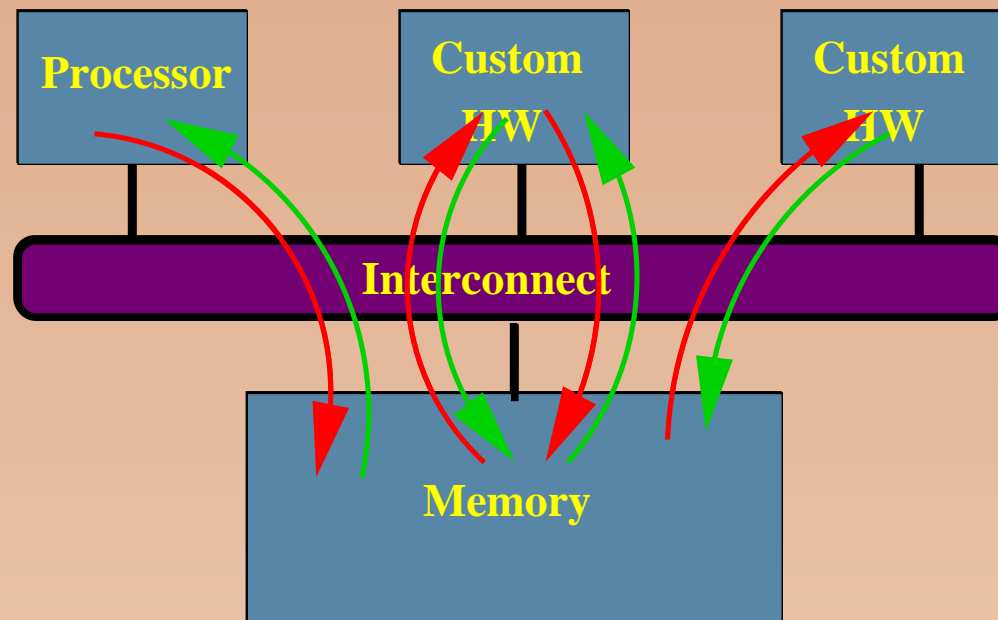
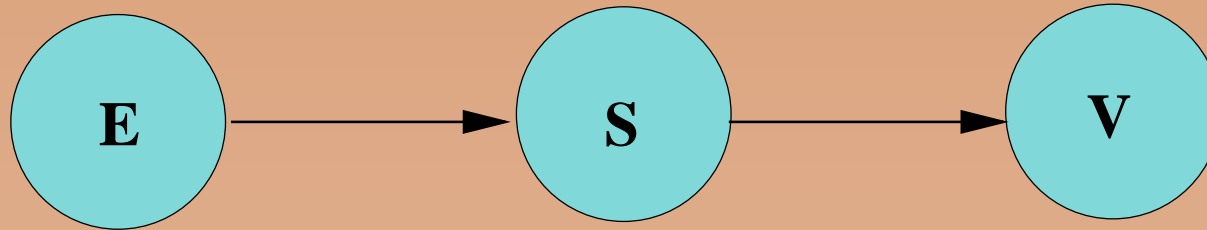
# MPEG Encoding Case Study



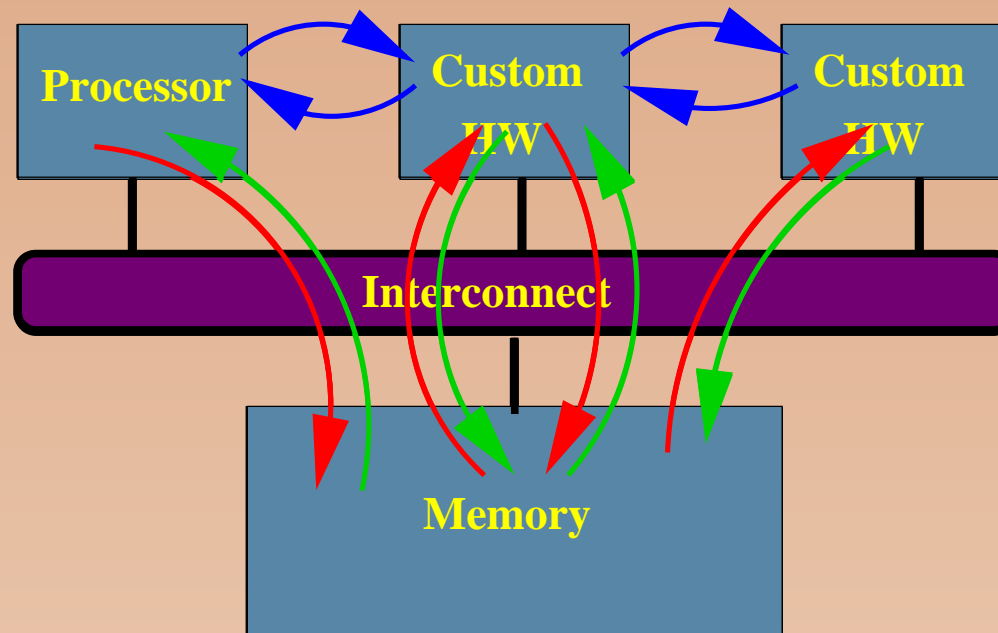
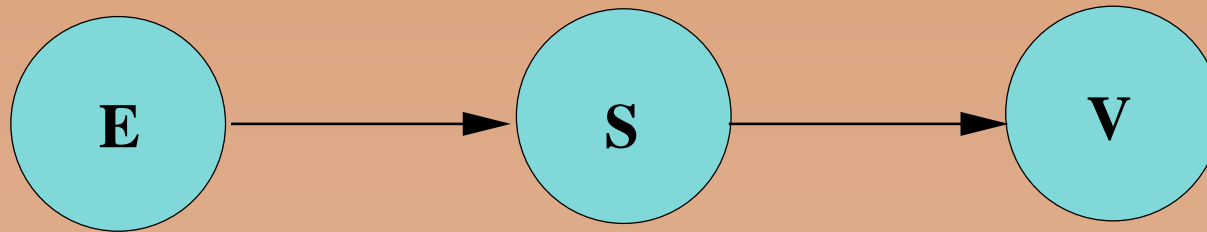
# MPEG Encoding Case Study



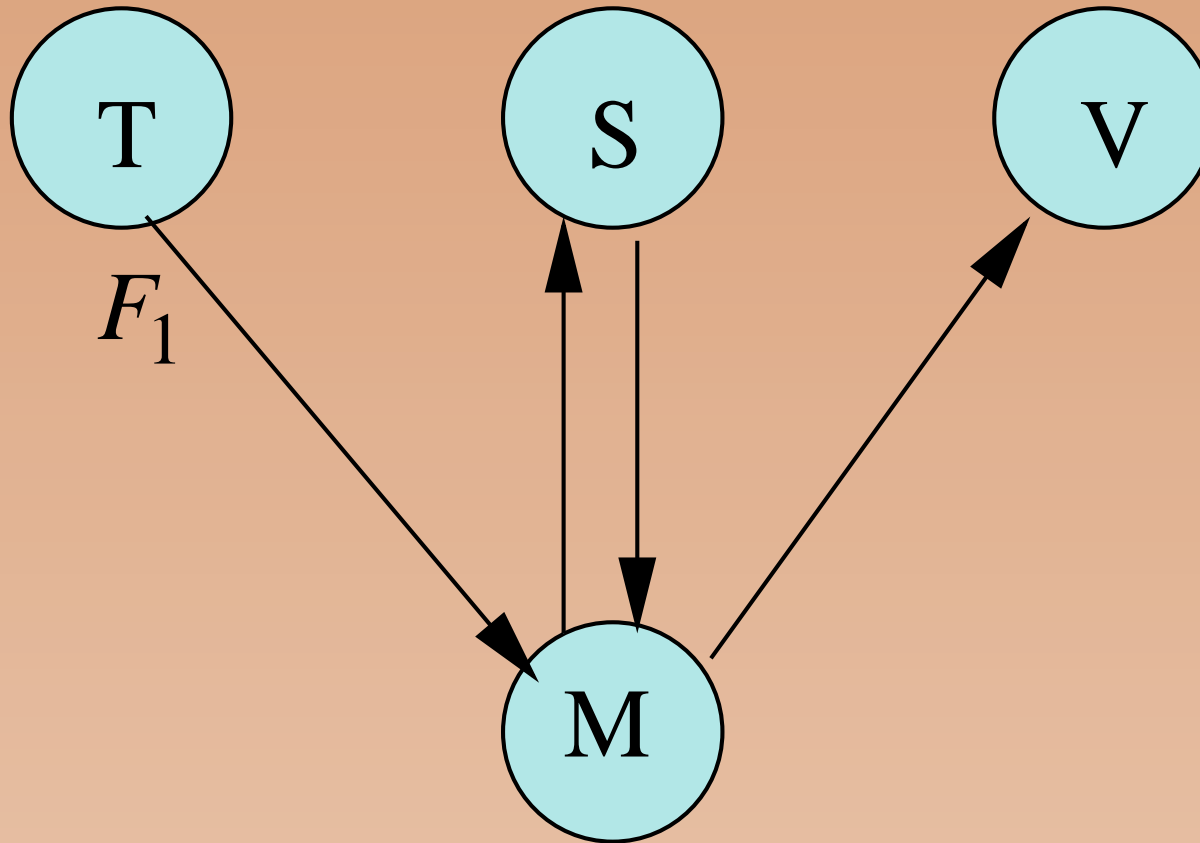
# MPEG Encoding Case Study



# MPEG Encoding Case Study

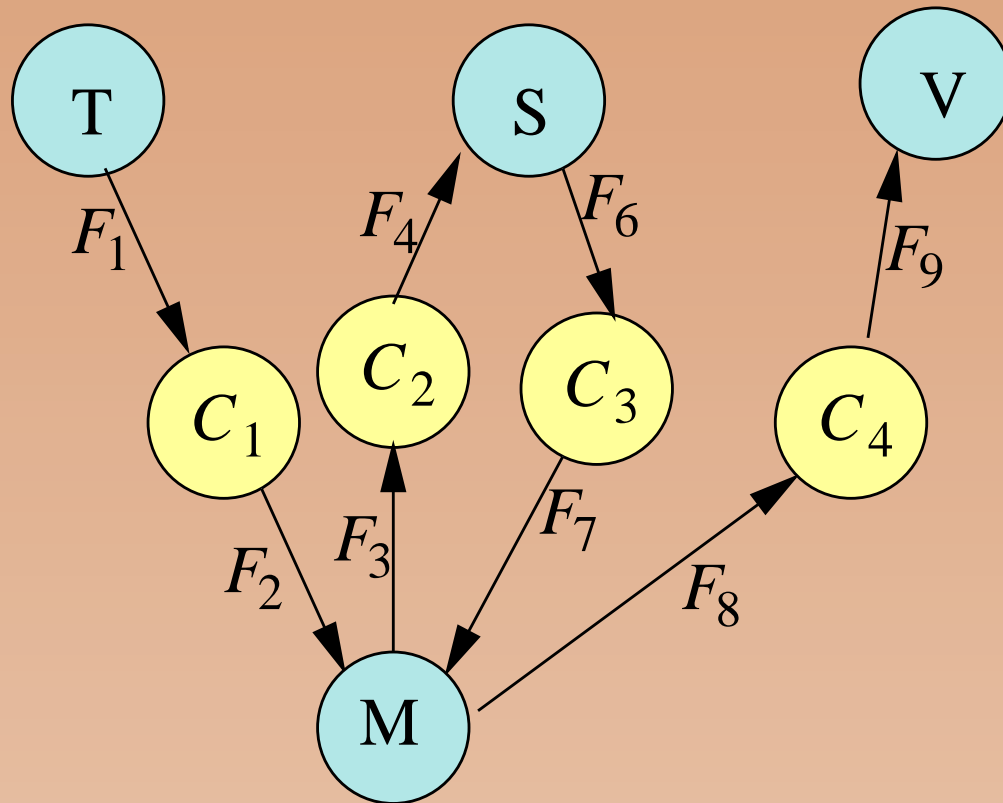


## MPEG Encoding Case Study - cont'd



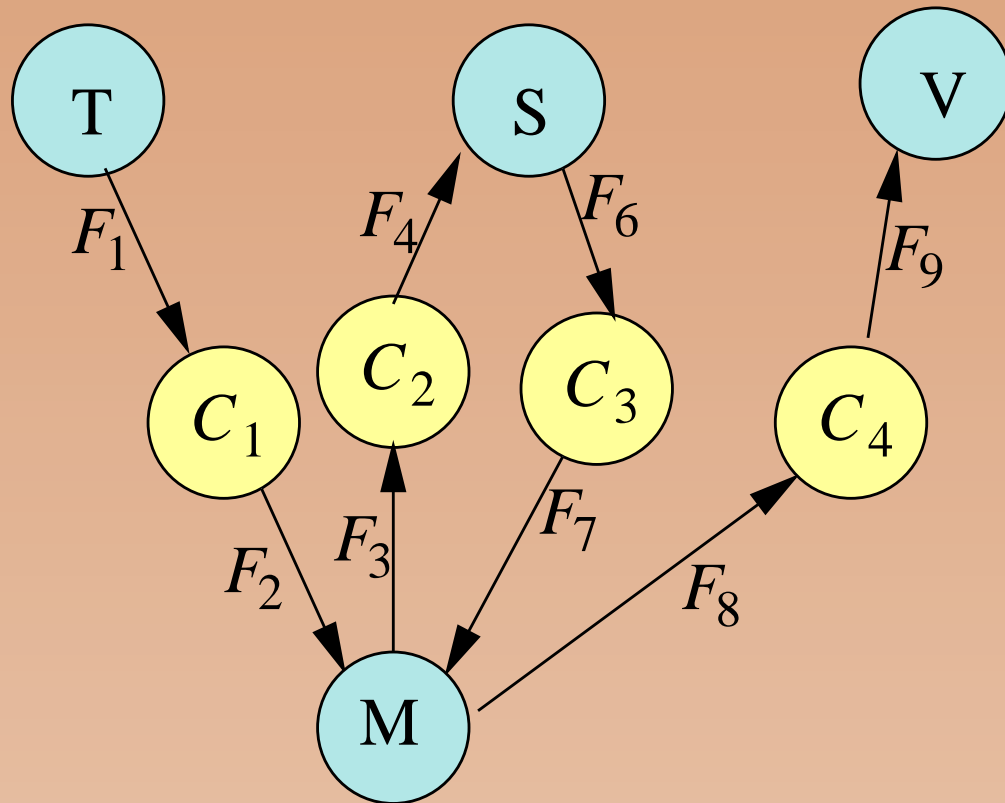
$$F_1 \sim (0, \rho_t)$$

## MPEG Encoding Case Study - cont'd



$$F_1 \sim (0, \rho_t)$$

## MPEG Encoding Case Study - cont'd



$$F_1 \sim (0, \rho_t)$$

$$C_1 : (\rho_t, D_1)$$

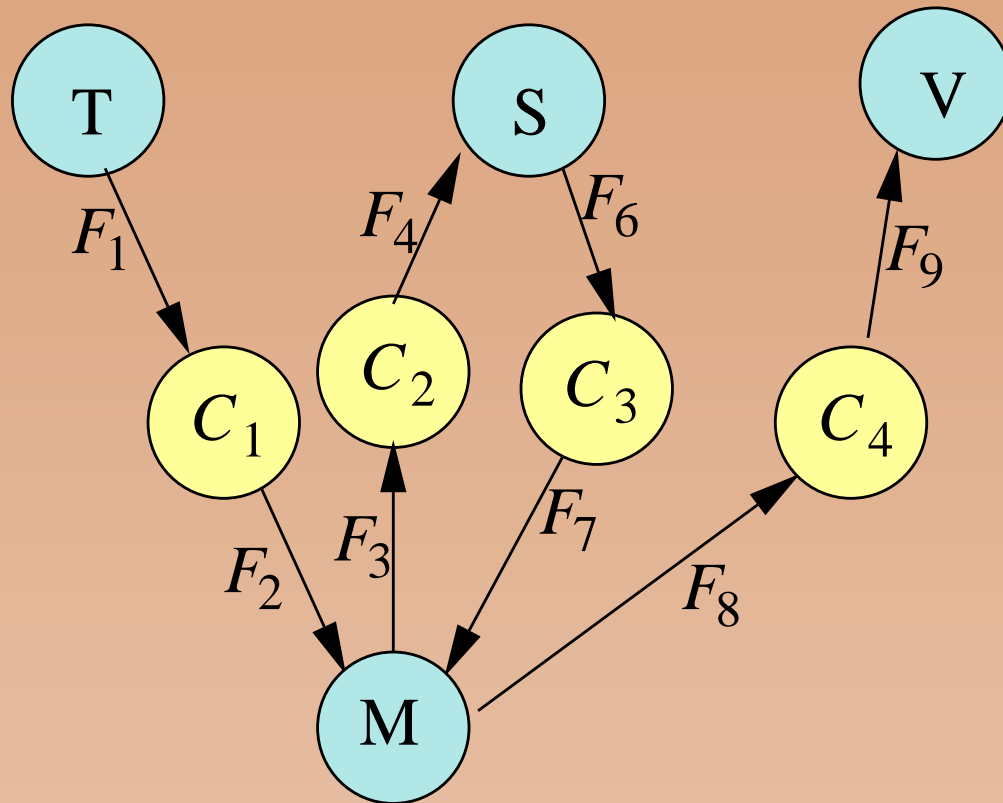
$$C_2 : (\rho_t, D_2)$$

$$C_3 : (\rho_t, D_3)$$

$$C_4 : (\rho_t, D_4)$$



## MPEG Encoding Case Study - cont'd



$$F_1 \sim (0, \rho_t)$$

$$C_1 : (\rho_t, D_1)$$

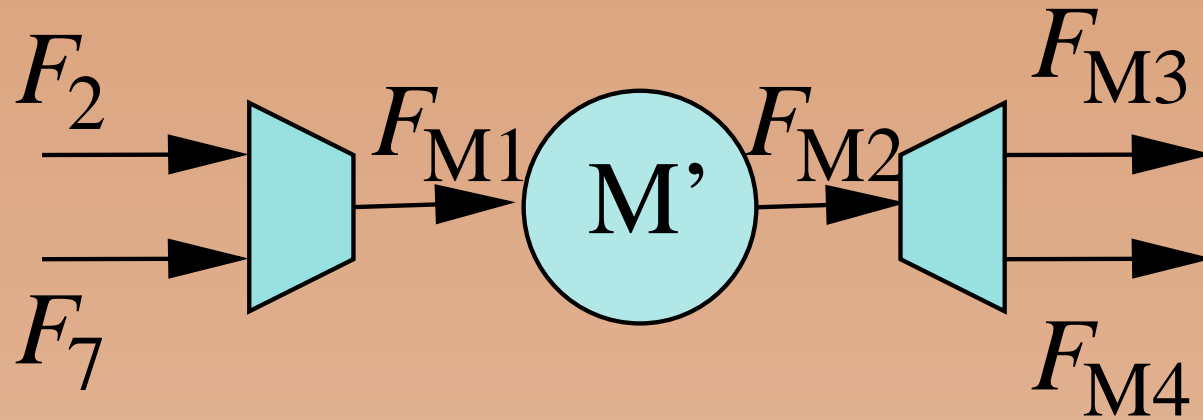
$$C_2 : (\rho_t, D_2)$$

$$C_3 : (\rho_t, D_3)$$

$$C_4 : (\rho_t, D_4)$$

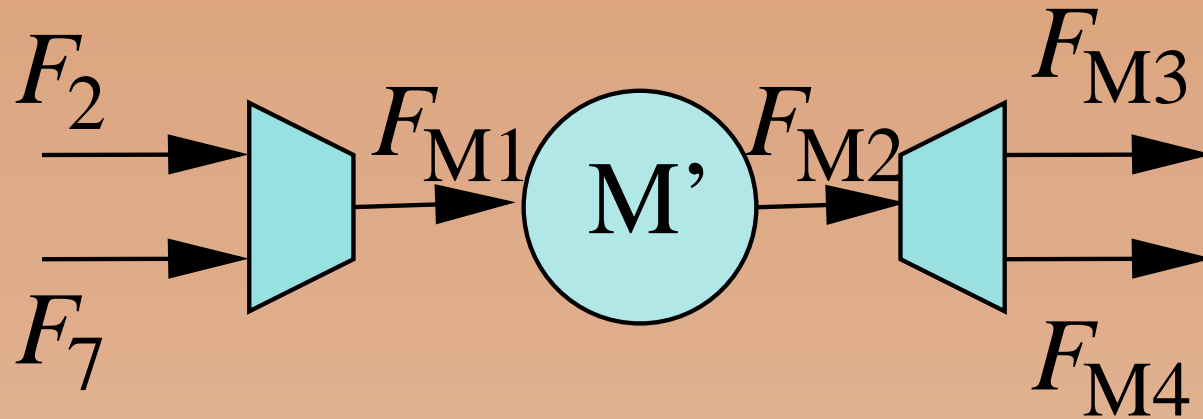
$$F_2 \sim (\rho_t D_1, \rho_t)$$

## MPEG Encoding Case Study - Memory



$$M' : (2\rho_t, D_{M'})$$

## MPEG Encoding Case Study - Memory



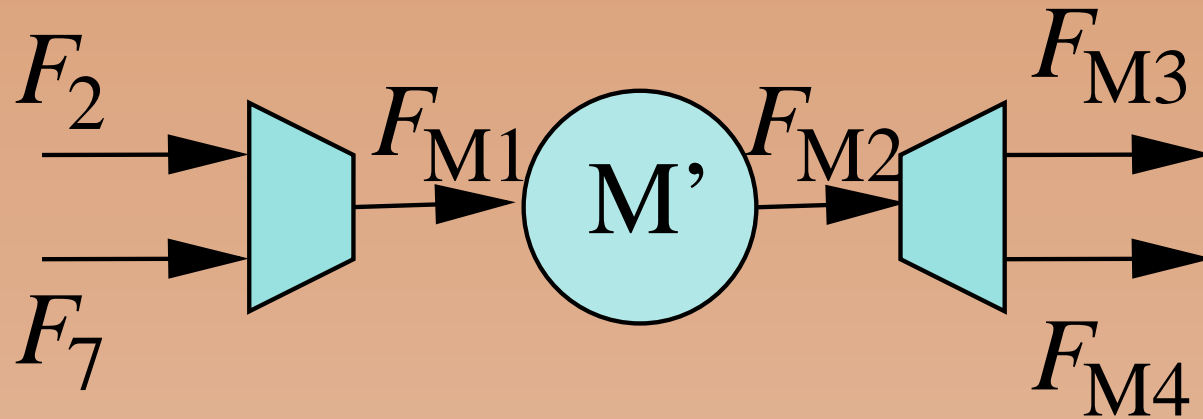
$$M' : (2\rho_t, D_{M'})$$

For a general multiplexer we have:

$$D_{\text{mux}} = \frac{\sigma_1 + \sigma_2}{C_{\text{out}} - \rho_1 - \rho_2}$$

$$F_{\text{muxout}} \sim (\sigma_1 + \sigma_2, \rho_1 + \rho_2)$$

## MPEG Encoding Case Study - Memory



$$M' : (2\rho_t, D_{M'})$$

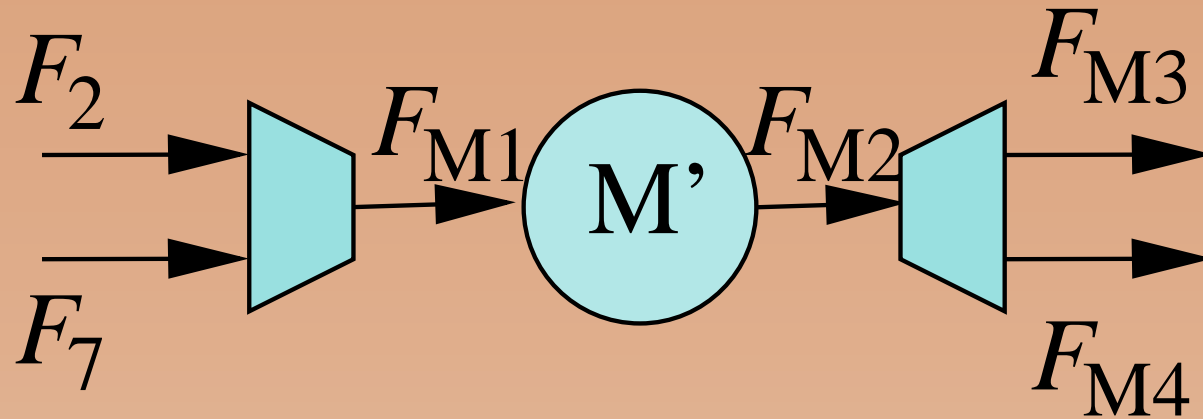
For a general multiplexer we have:

$$D_{\text{mux}} = \frac{\sigma_1 + \sigma_2}{C_{\text{out}} - \rho_1 - \rho_2}$$

$$F_{\text{muxout}} \sim (\sigma_1 + \sigma_2, \rho_1 + \rho_2)$$

$$F_{M3} \sim (\rho_t(D_1 + D_{\text{mux}} + D_{M'}), \rho_t)$$

## MPEG Encoding Case Study - Memory



$$M' : (2\rho_t, D_{M'})$$

For a general multiplexer we have:

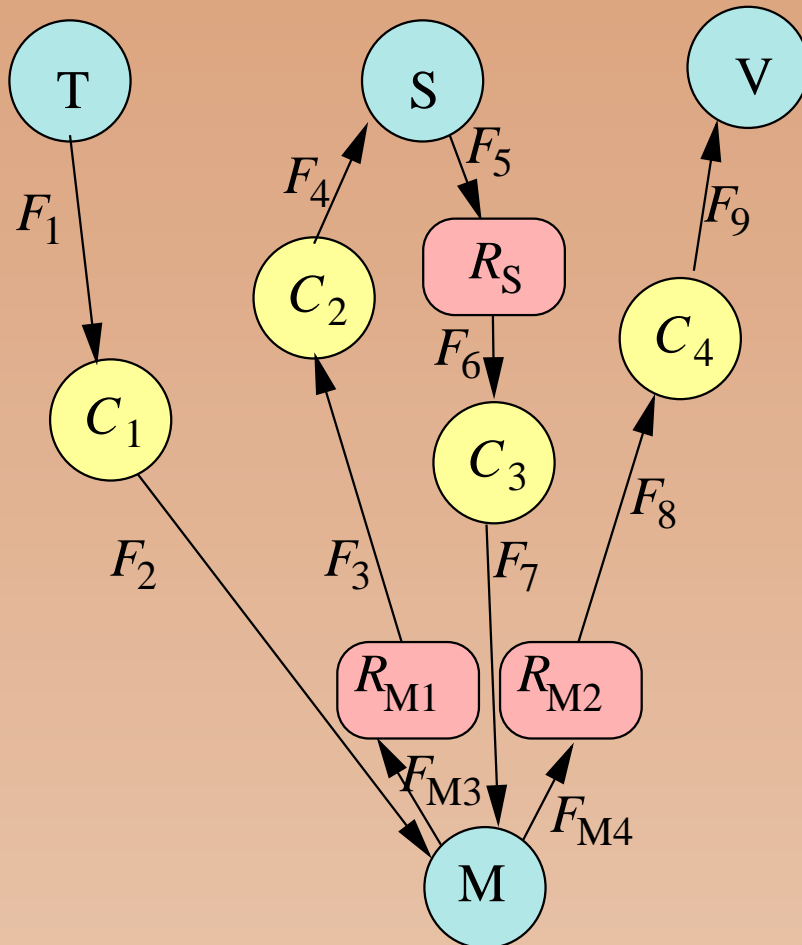
$$D_{\text{mux}} = \frac{\sigma_1 + \sigma_2}{C_{\text{out}} - \rho_1 - \rho_2}$$

$$F_{\text{muxout}} \sim (\sigma_1 + \sigma_2, \rho_1 + \rho_2)$$

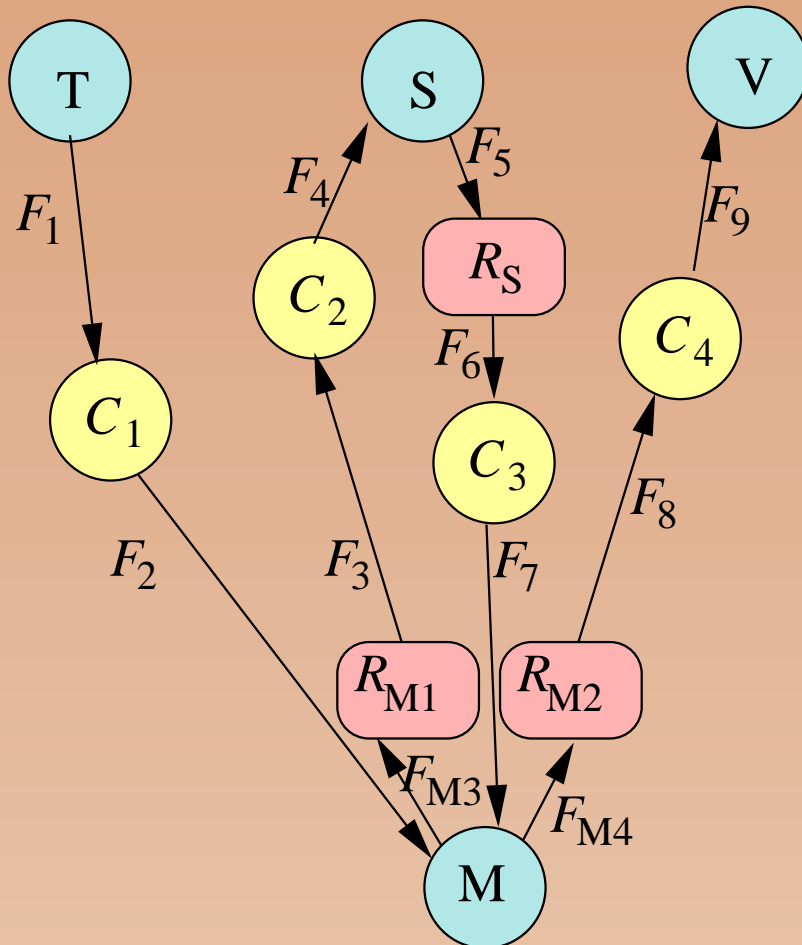
$$F_{M3} \sim (\rho_t(D_1 + D_{\text{mux}} + D_{M'}), \rho_t)$$

$$F_{M4} \sim ?$$

## MPEG Encoding Case Study - cont'd



## MPEG Encoding Case Study - cont'd



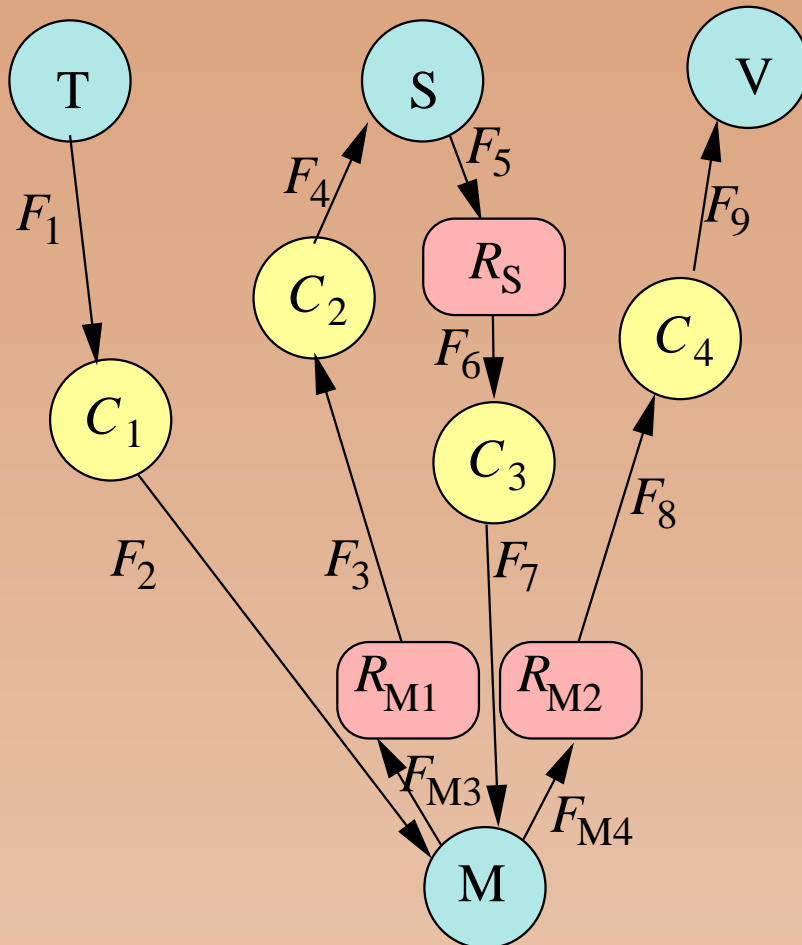
$$R_{M1} \sim (S_{\text{buffer}}, \rho_t);$$

$$R_{M2} \sim (S_{\text{buffer}}, \rho_t);$$

$$R_S \sim (S_{\text{buffer}}, \rho_t);$$

$S_{\text{buffer}}$  is the size of the input buffer in S.

## MPEG Encoding Case Study - cont'd



$$R_{M1} \sim (S_{\text{buffer}}, \rho t);$$

$$R_{M2} \sim (S_{\text{buffer}}, \rho t);$$

$$R_S \sim (S_{\text{buffer}}, \rho t);$$

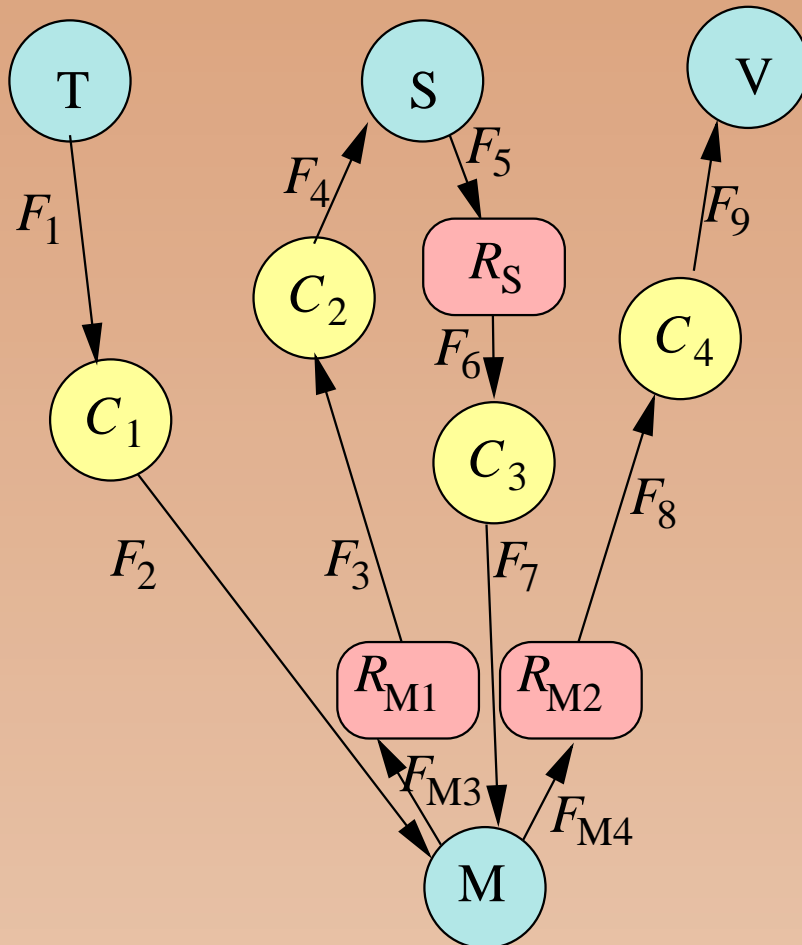
$S_{\text{buffer}}$  is the size of the input buffer in S.

$$D_{(\sigma, \rho)\text{-regulator}} = \frac{\max(0, \sigma' - \sigma)}{\rho}$$

$$B_{(\sigma, \rho)\text{-regulator}} = \max(0, \sigma' - \sigma)$$



## MPEG Encoding Case Study - cont'd



$$R_{M1} \sim (S_{\text{buffer}}, \rho_t);$$

$$R_{M2} \sim (S_{\text{buffer}}, \rho_t);$$

$$R_S \sim (S_{\text{buffer}}, \rho_t);$$

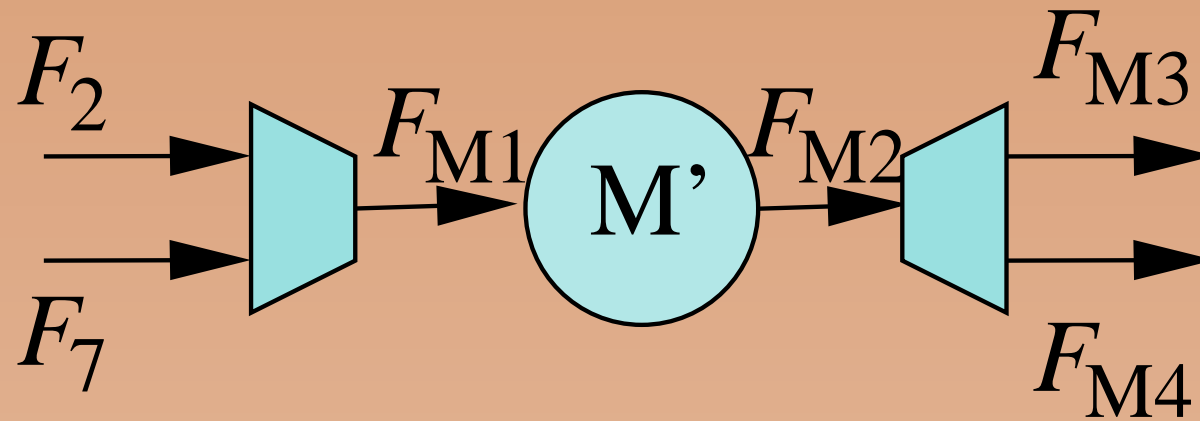
$S_{\text{buffer}}$  is the size of the input buffer in S.

$$F_6 \sim (S_{\text{buffer}}, \rho_t)$$

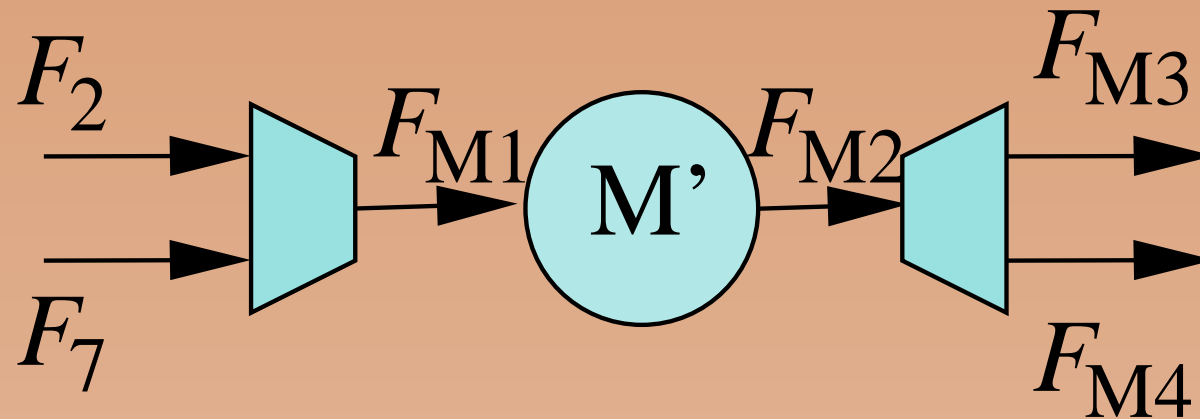
$$C_3 : (\rho_t, D_3)$$

$$F_7 \sim (S_{\text{buffer}} + \rho_t D_3, \rho_t)$$

## MPEG Encoding Case Study - Memory



## MPEG Encoding Case Study - Memory



$$M' : (2\rho_t, D_{M'})$$

$$D_{\text{mux}} = \frac{\sigma_1 + \sigma_2}{C_{\text{out}} - \rho_1 - \rho_2} = \frac{S_{\text{buffer}} + \rho_t(D_1 + D_3)}{C_{\text{out}} - 2\rho_t}$$

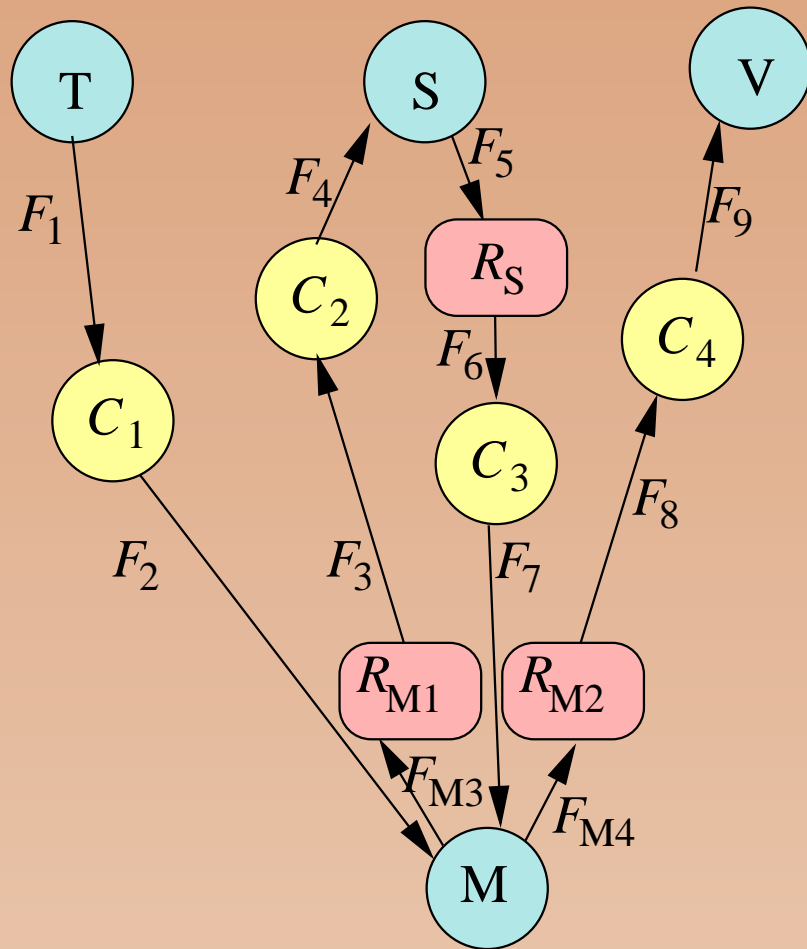
$$F_{M1} \sim (S_{\text{buffer}} + \rho_t(D_1 + D_3), 2\rho_t)$$

$$F_{M2} \sim (S_{\text{buffer}} + \rho_t(D_1 + D_3 + 2D_{M'}), 2\rho_t)$$

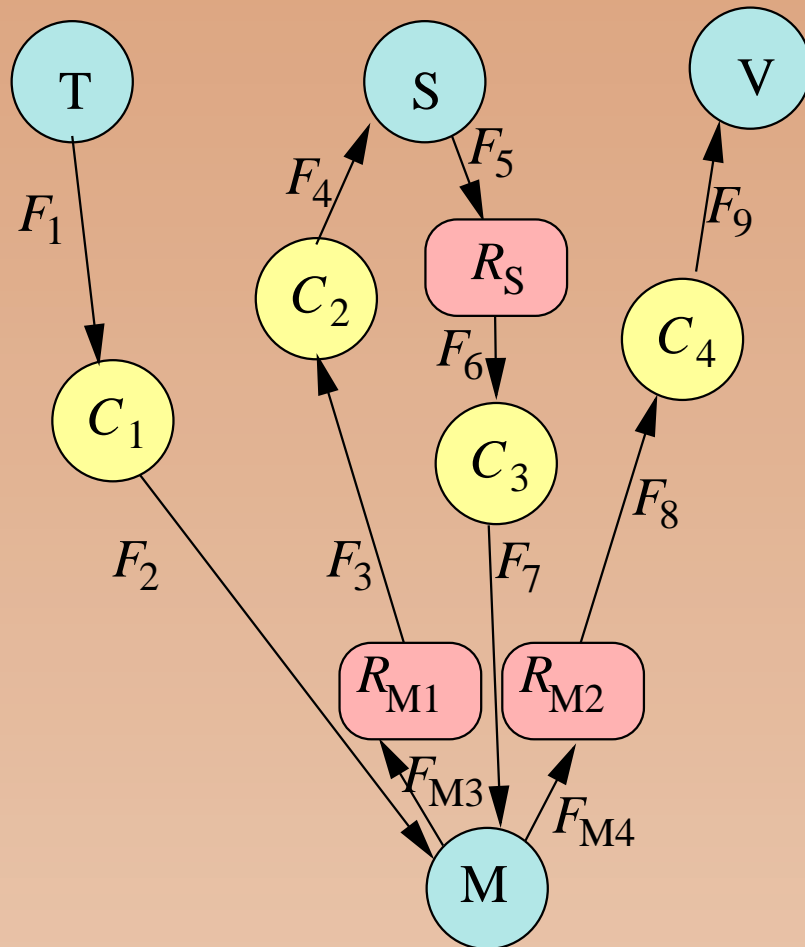
$$F_{M3} \sim (\rho_t(D_1 + D_{\text{mux}} + D_{M'}), \rho_t)$$

$$F_{M4} \sim (S_{\text{buffer}} + \rho_t(D_3 + D_{\text{mux}} + D_{M'}), \rho_t)$$

# MPEG Encoding Case Study - cont'd



## MPEG Encoding Case Study - cont'd



Backlog of the regulators:

$$B_{RM1} = \max(0, \rho_t(D_1 + D_{\text{mux}} + D_{M'}) - S_{\text{buffer}})$$

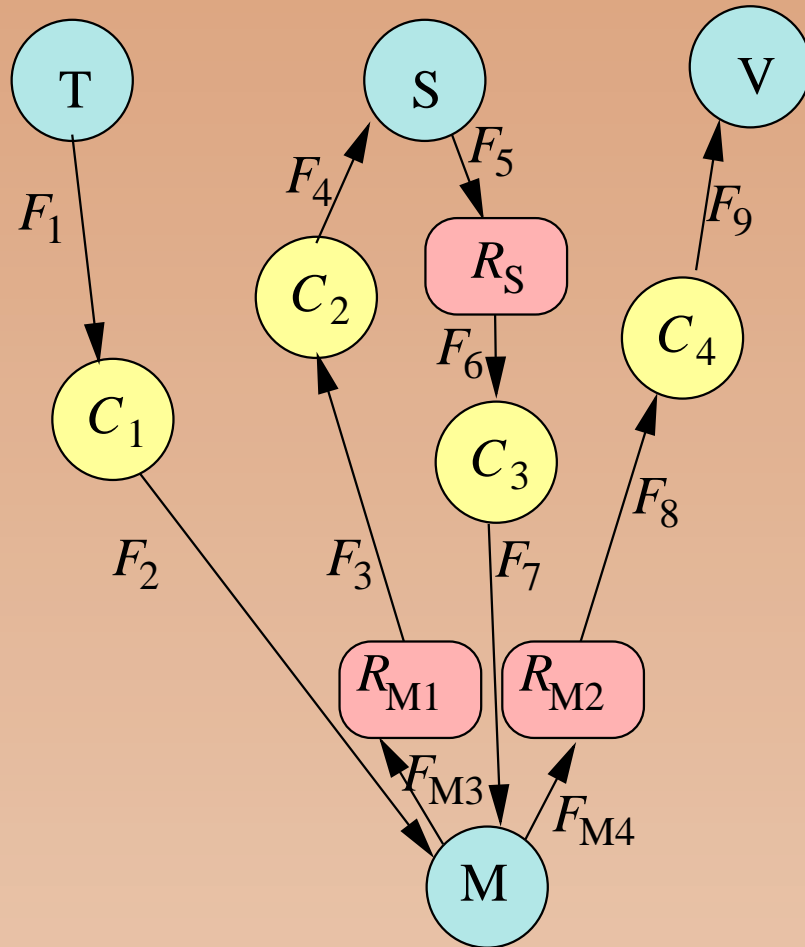
$$B_{RM2} = \max(0, 128B + \rho_t(D_3 + D_{\text{mux}} + D_{M'}) - S_{\text{buffer}})$$

Delay of the regulators:

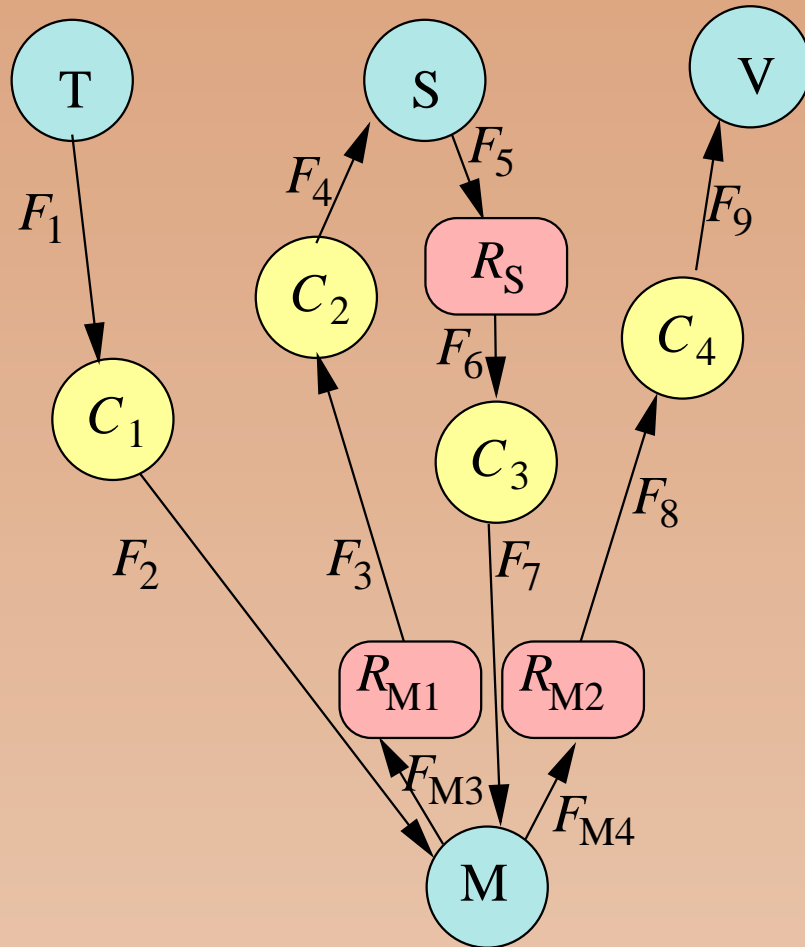
$$D_{RM1} = \frac{B_{RM1}}{\rho_t}$$

$$D_{RM2} = \frac{B_{RM2}}{\rho_t}$$

# MPEG Encoding Case Study - cont'd



## MPEG Encoding Case Study - cont'd



The flow from the memory to S:

$$F_3 \sim (S_{\text{buffer}}, \rho_t)$$

$$C_2 : (\rho_t, D_2)$$

$$F_4 \sim (S_{\text{buffer}} + \rho D_2, \rho_t),$$

A characterization of S and its output:

$$S : \left( \rho_t, \frac{S_{\text{buffer}}}{\rho_t} \right)$$

$$F_5 \sim (2S_{\text{buffer}} + \rho_t D_2, \rho_t)$$

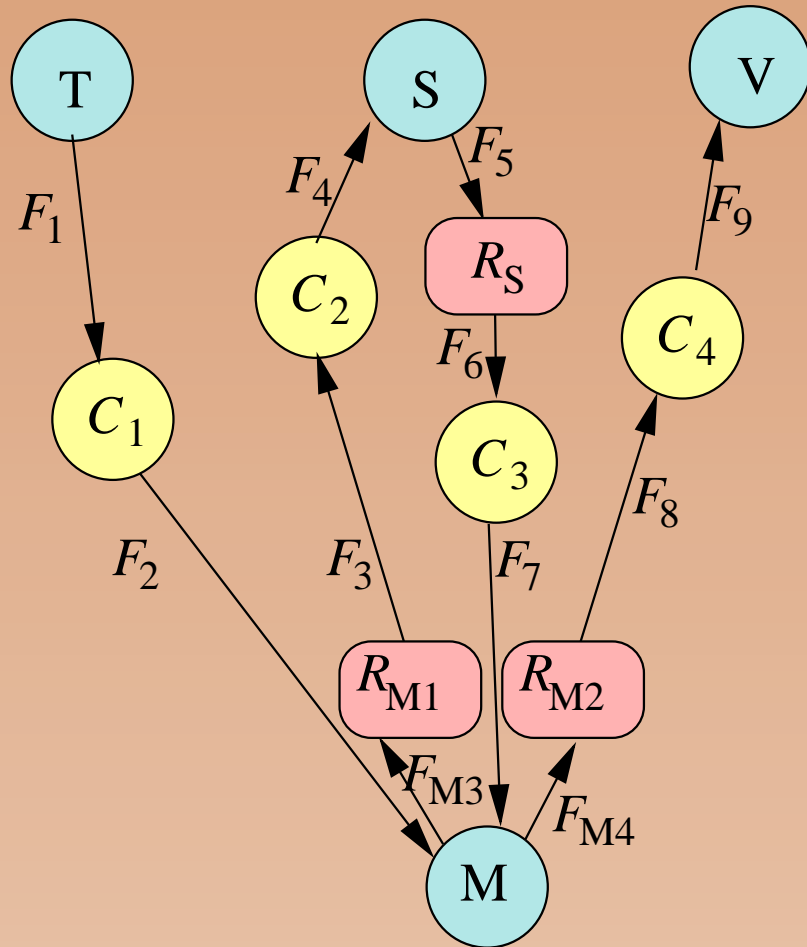
The flows between memory and V:

$$F_8 \sim (S_{\text{buffer}}, \rho_t)$$

$$C_4 : (\rho, D_4)$$

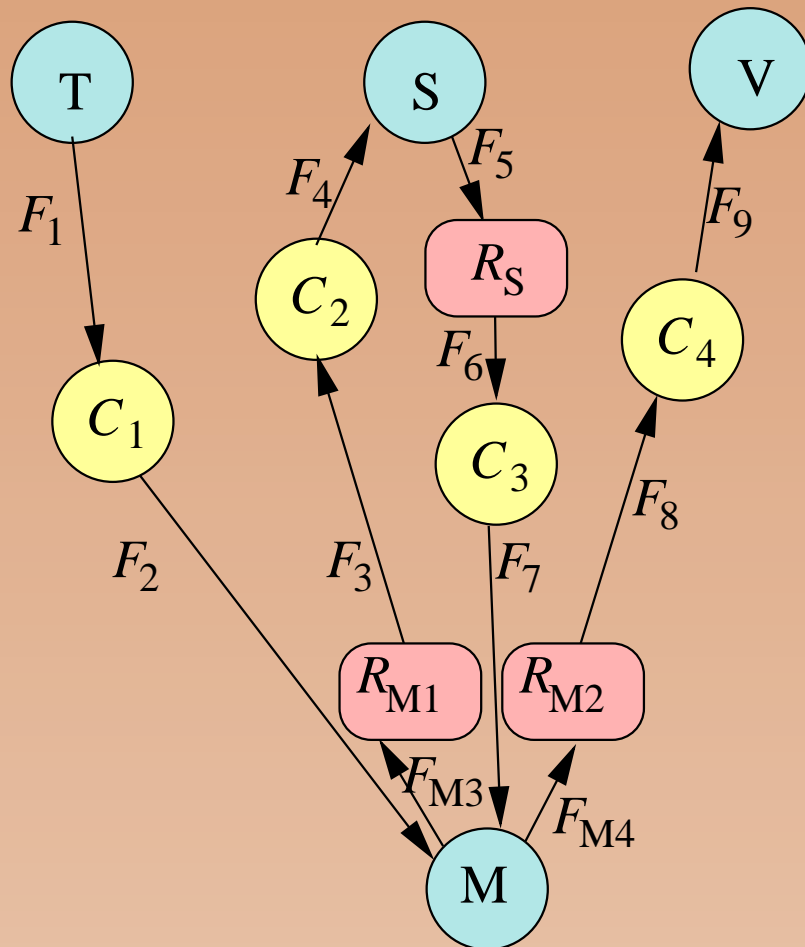
$$F_9 \sim (S_{\text{buffer}} + \rho_t D_4, \rho_t)$$

# MPEG Encoding Case Study - cont'd





## MPEG Encoding Case Study - cont'd



End to end delay:

$$\begin{aligned}
 D_{\text{total}} = & D_1 + D_{\text{mux}} + D_{M'} \\
 & + D_{RM1} + D_2 + D_S + D_{RS} + D_3 \\
 & + D_{\text{mux}} + D_{M'} + D_{RM2} + D_4
 \end{aligned}$$

The flow at V:

$$F_{T \rightarrow V} \sim (0 + \rho_t D_{\text{total}}, \rho_t)$$

## Modeling with Regulated Flows

- Interconnect:
  - ★ Model each channel by available bandwidth and maximum delay variation;
  - ★ Model each node in the interconnect as an arbiter;
- Model read request, write acknowledge as separate flows;
- Model synchronization as separate flows;
- A simple generalization of  $(\sigma, \rho)$  flows is

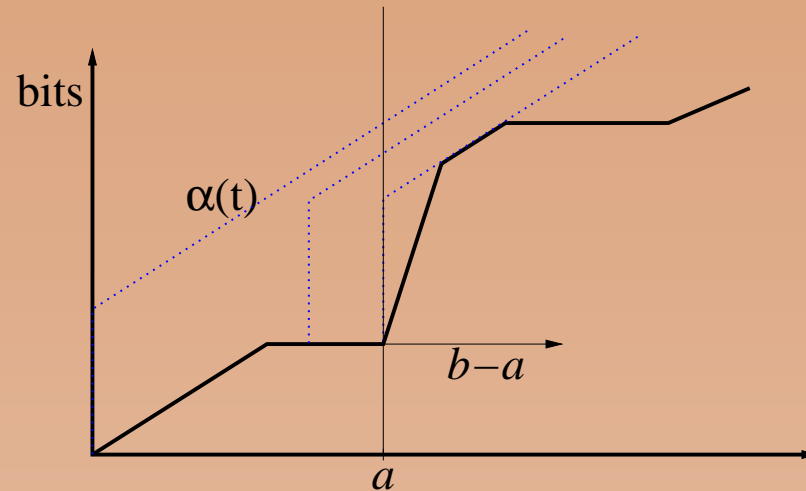
$$F \sim \min(\sigma_i, \rho_i), i > 0$$

$$F(b) - F(a) \leq \min_i(\sigma_i + \rho_i(b - a))$$

- Good analysis depends on good element models;



## Network Calculus - Arrival Curves



Given a monotonically increasing function  $\alpha$ , defined for  $t \geq 0$ ,  $\alpha$  is an arrival curve for flow  $F$  if for all  $0 \leq a \leq b$ :

$$F(b) - F(a) \leq \alpha(b - a)$$

## Network Calculus - Min-Plus Convolution

Given two monotonically increasing functions  $f$  and  $g$ . The min-plus convolution of  $f$  and  $g$  is the function

$$(f \otimes g)(t) = \inf_{0 \leq s \leq t} (f(t-s) + g(s))$$



## Network Calculus - Min-Plus Convolution

Given two monotonically increasing functions  $f$  and  $g$ . The min-plus convolution of  $f$  and  $g$  is the function

$$(f \otimes g)(t) = \inf_{0 \leq s \leq t} (f(t-s) + g(s))$$

If  $\alpha$  is an arrival curve for  $F$  we have:

$$F \leq F \otimes \alpha$$



## Network Calculus - Min-Plus Convolution

Given two monotonically increasing functions  $f$  and  $g$ . The min-plus convolution of  $f$  and  $g$  is the function

$$(f \otimes g)(t) = \inf_{0 \leq s \leq t} (f(t-s) + g(s))$$

If  $\alpha$  is an arrival curve for  $F$  we have:

$$F \leq F \otimes \alpha$$

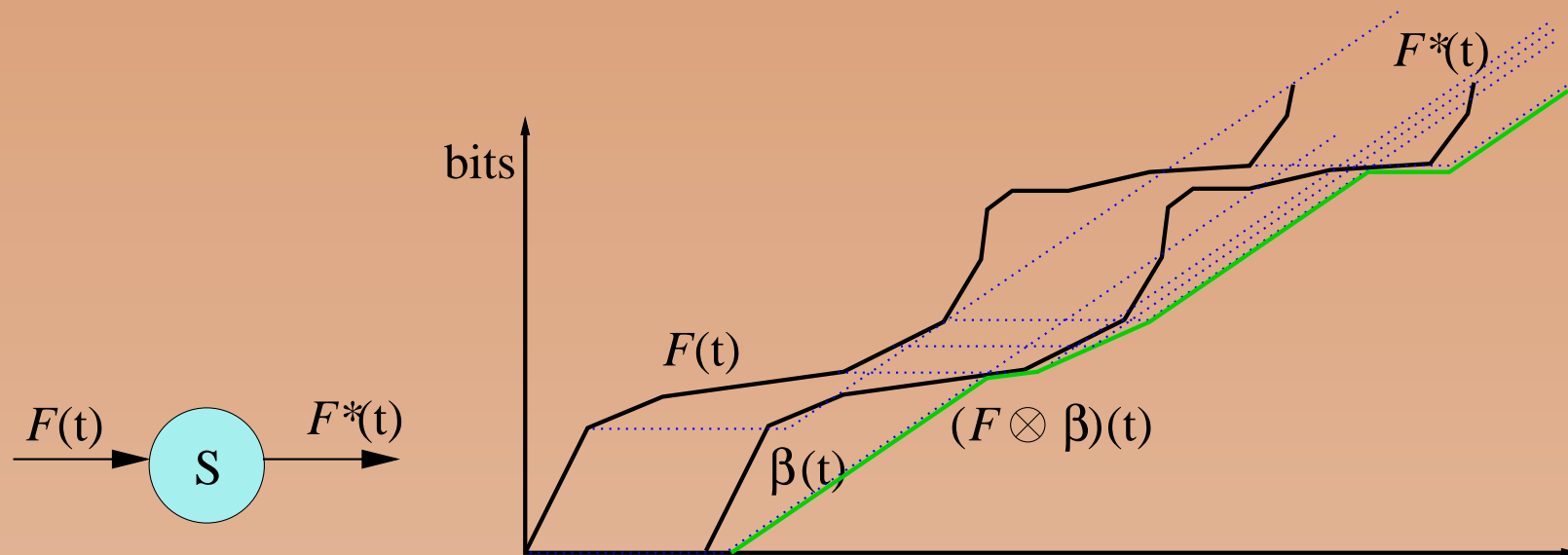
and

$$F \leq \alpha \otimes \alpha$$

with  $\alpha \otimes \alpha$  being the best bound that we can find based on information of  $\alpha$ .



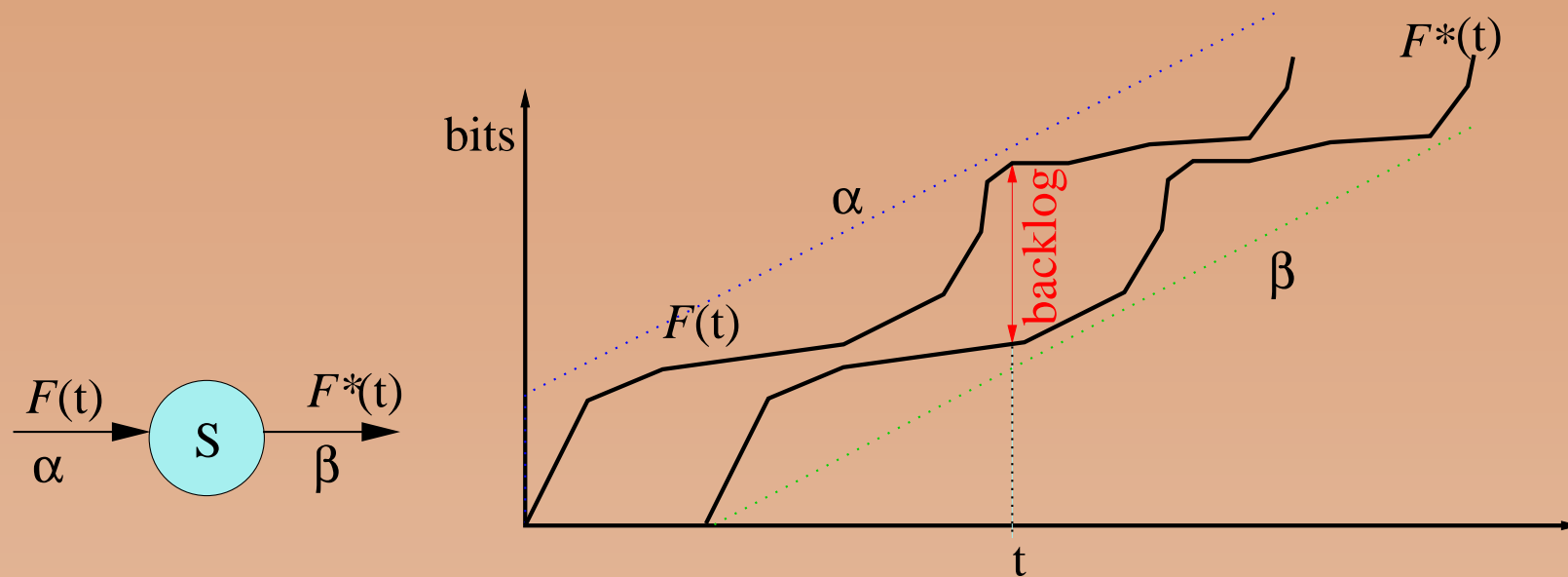
## Network Calculus - Service Curves



Given a system  $S$  with an input flow  $F$  and an output flow  $F^*$ .  $S$  offers the flow a service curve  $\beta$  if and only if  $\beta$  is a monotonically increasing function and  $F^* \geq F \otimes \beta$  which means that

$$F^*(t) \geq \inf_{s \leq t} (F(t) + \beta(t - s))$$

## Network Calculus - Backlog Bound

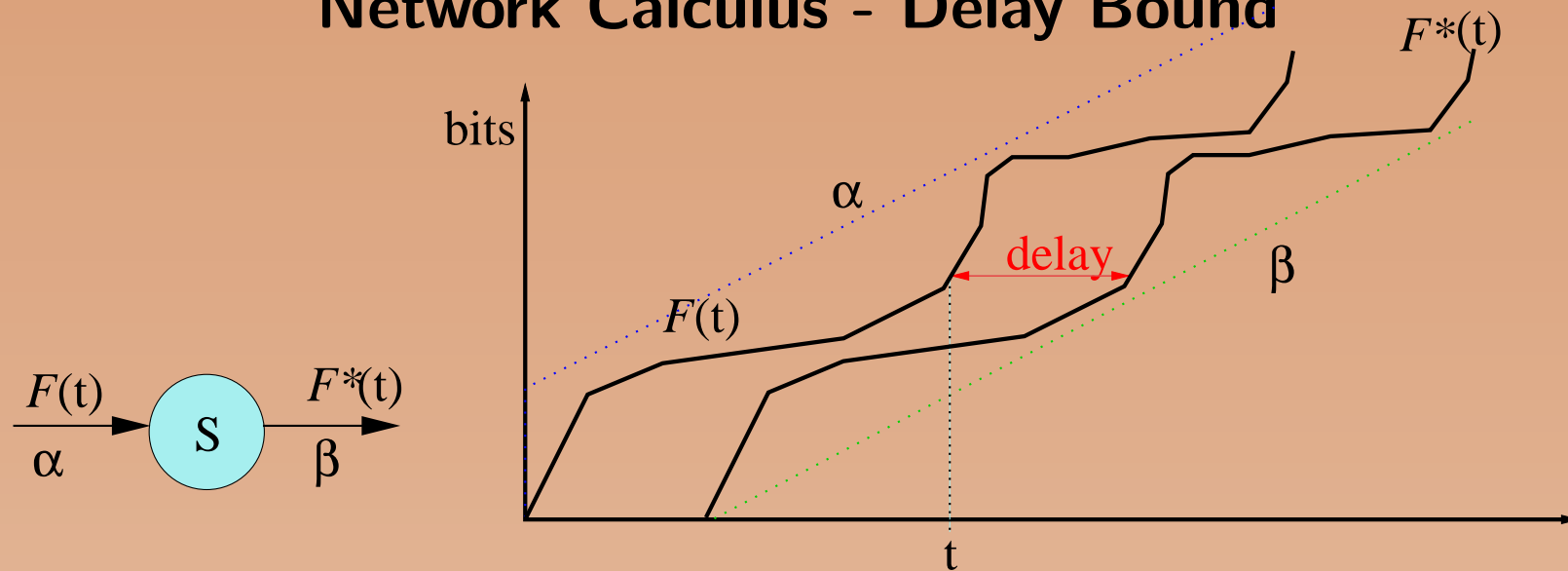


Given a flow  $F$  constrained by arrival curve  $\alpha$  and a system offering a service curve  $\beta$ , the backlog  $F(t) - F^*(t)$  for all  $t$  satisfies

$$F(t) - F^*(t) \leq \sup_{s \geq 0} (\alpha(s) - \beta(s))$$



## Network Calculus - Delay Bound



Given a flow  $F$  constrained by arrival curve  $\alpha$  and a system offering a service curve  $\beta$ , the delay  $d(t)$  at time  $t$  is

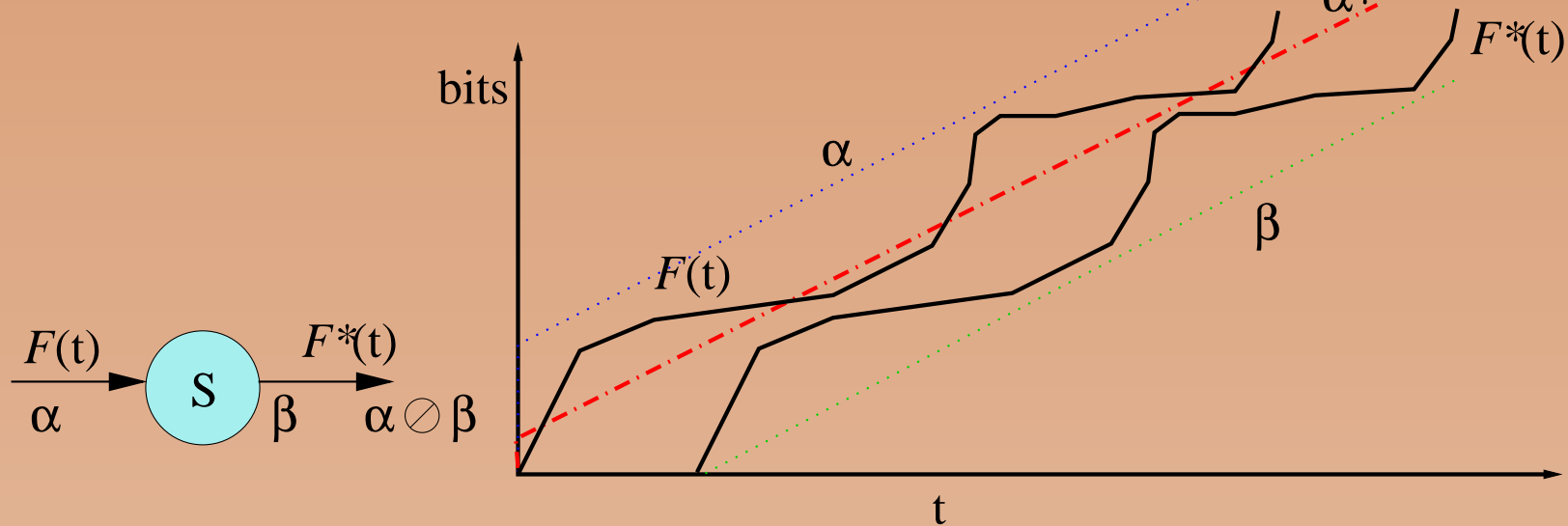
$$d(t) = \inf(\tau \geq 0 : F(t) \leq F^*(t + \tau)).$$

It satisfies

$$d(t) \leq h(\alpha, \beta) = \sup_{t \geq 0} (\inf_{\tau \geq 0} (\alpha(t) \leq \beta(t + \tau)))$$



# Network Calculus - Output Arrival Curve



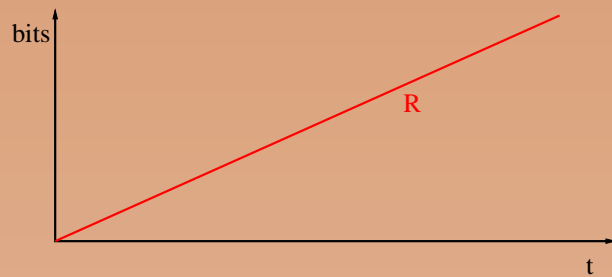
Given a flow  $F$  constrained by arrival curve  $\alpha$  and a system offering a service curve  $\beta$ , the output flow  $F^*$  is constrained by the arrival curve  $\alpha^*$

$$\alpha^* = \alpha \circledast \beta.$$

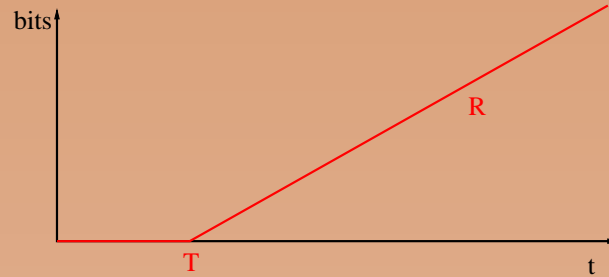
$$(\alpha \circledast \beta)(t) = \sup_{s \geq 0} (\alpha(t + s) - \beta(s))$$



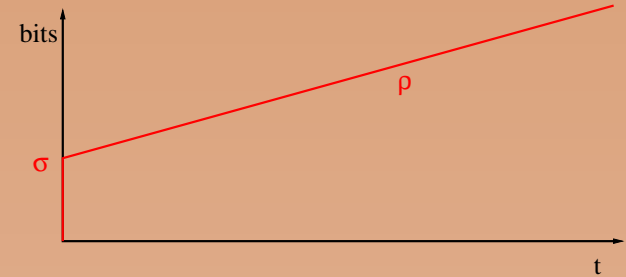
# Network Calculus - Useful Functions



Peak rate function:  
 $\lambda_R(t) = Rt$

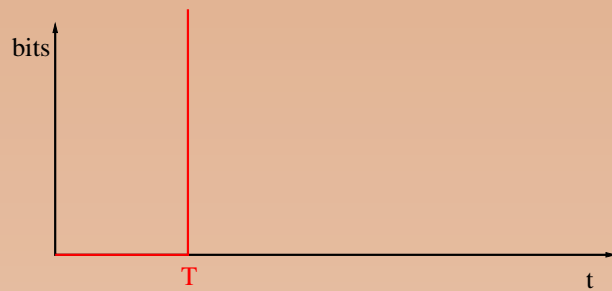


Rate latency function:  
 $\beta_{R,T}(t) = R[t - T]^+$



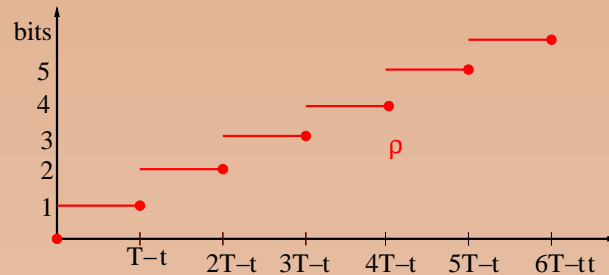
Affine function:  

$$\gamma_{\sigma,\rho}(t) = \begin{cases} 0 & \text{for } t = 0 \\ \sigma + \rho t & \text{for } t > 0 \end{cases}$$



Burst-delay function:  

$$\delta_T(t) = \begin{cases} 0 & \text{for } t \leq T \\ \infty & \text{for } t > T \end{cases}$$



Staircase function:  
 $v_{T,\tau}(t) = \lceil (t + \tau) / T \rceil$



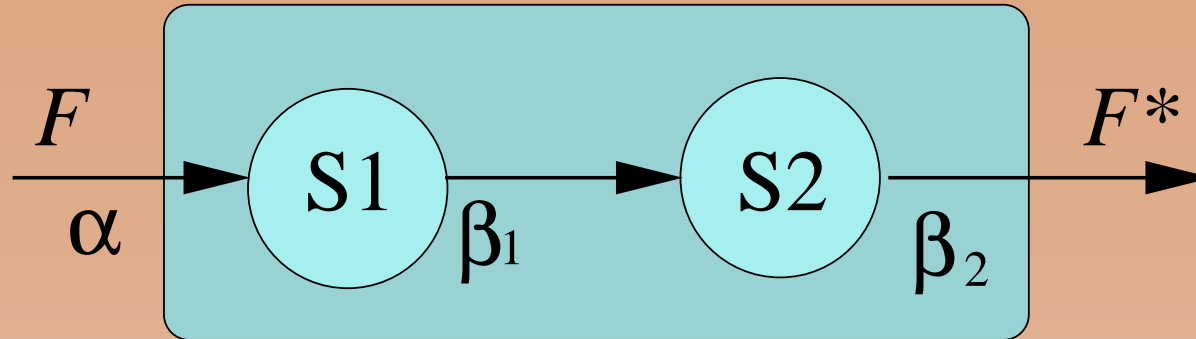
Step function:  

$$u_T(t) = \begin{cases} 0 & \text{for } t \leq T \\ 1 & \text{for } t > T \end{cases}$$



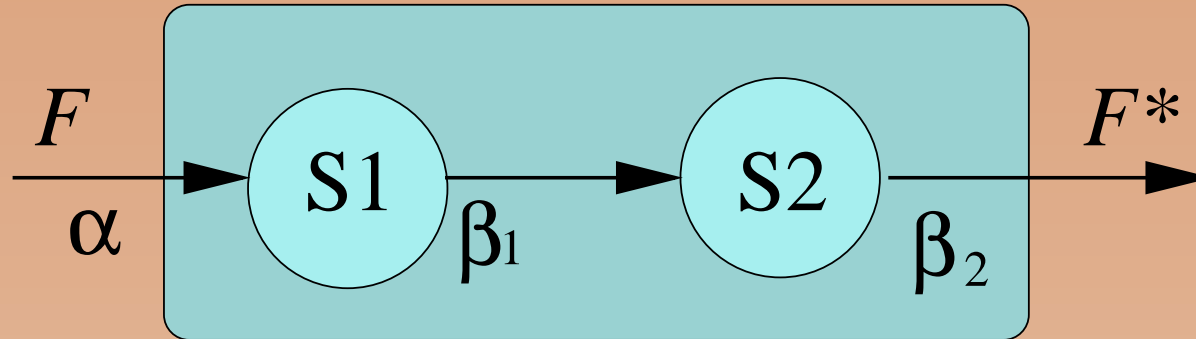
## Network Calculus - Concatenation of Nodes

$$S / \beta_1 \otimes \beta_2$$



## Network Calculus - Concatenation of Nodes

$$S / \beta_1 \otimes \beta_2$$



Example:

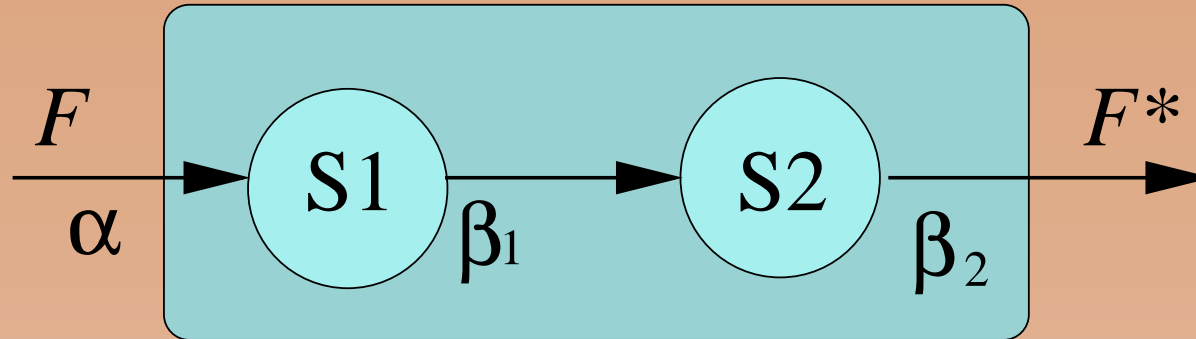
$$\beta_1 = \beta_{R_1, T_1}$$

$$\beta_2 = \beta_{R_2, T_2}$$

$$\beta_{R_1, T_1} \otimes \beta_{R_2, T_2} = \beta_{\min(R_1, R_2), T_1 + T_2}$$

## Network Calculus - Concatenation of Nodes

$$S / \beta_1 \otimes \beta_2$$



Example:

$$\beta_1 = \beta_{R_1, T_1}$$

$$\beta_2 = \beta_{R_2, T_2}$$

$$\beta_{R_1, T_1} \otimes \beta_{R_2, T_2} = \beta_{\min(R_1, R_2), T_1 + T_2}$$

Useful properties:

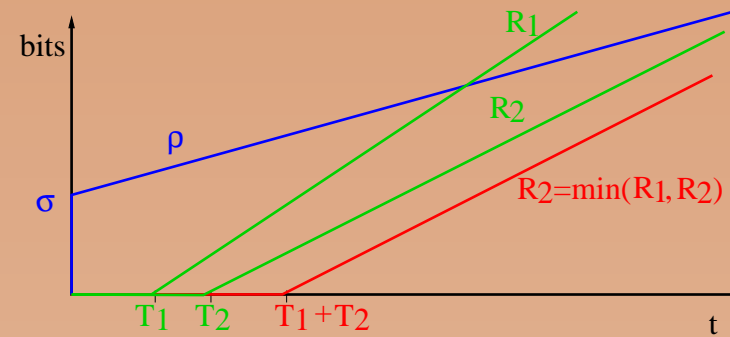
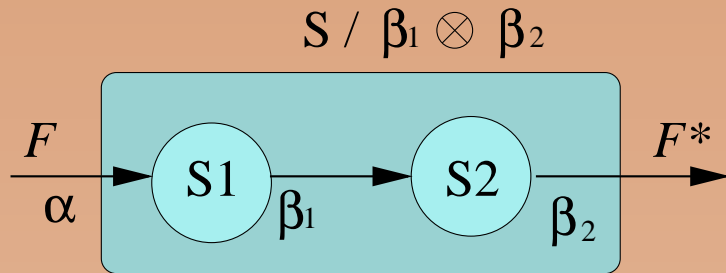
$$f \otimes g = g \otimes f$$

$$(f \otimes g) \otimes h = f \otimes (g \otimes h)$$

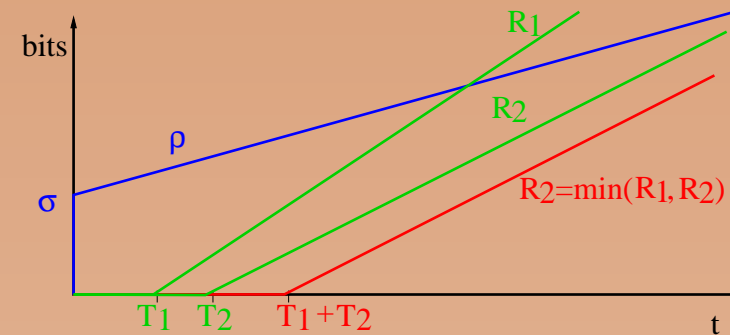
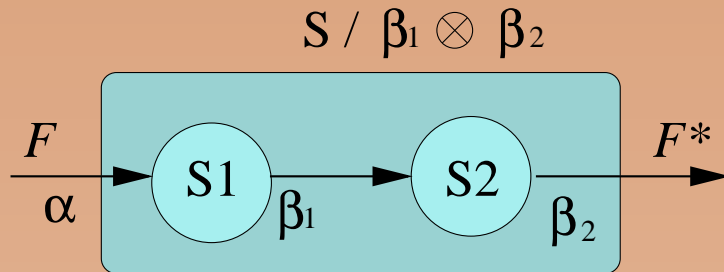
$$(f + c) \otimes g = (f \otimes g) + c \text{ for any constant } c \in \mathbb{R}$$



# Network Calculus - Pay Bursts Only Once



## Network Calculus - Pay Bursts Only Once



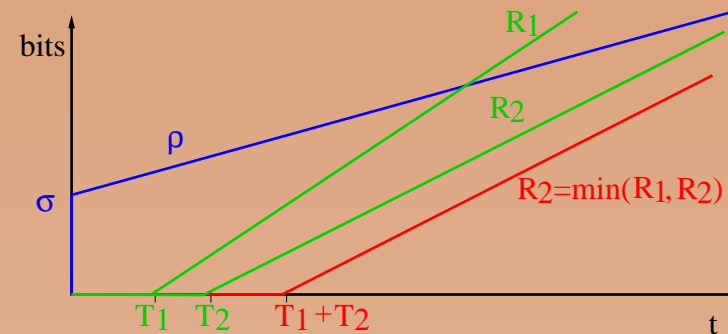
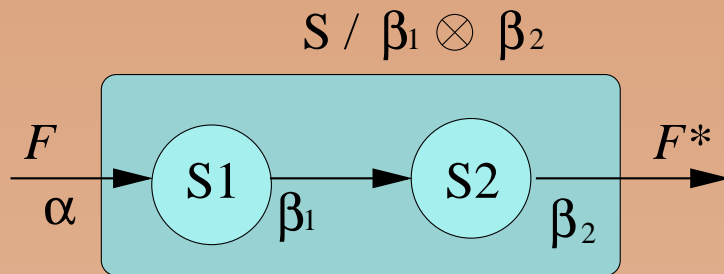
$$\alpha = \gamma_{\rho, \sigma}$$

$$\beta_1 = \beta_{R_1, T_1} = R_1 \max(0, t - T_1)$$

$$\beta_2 = \beta_{R_2, T_2} = R_2 \max(0, t - T_2)$$



## Network Calculus - Pay Bursts Only Once



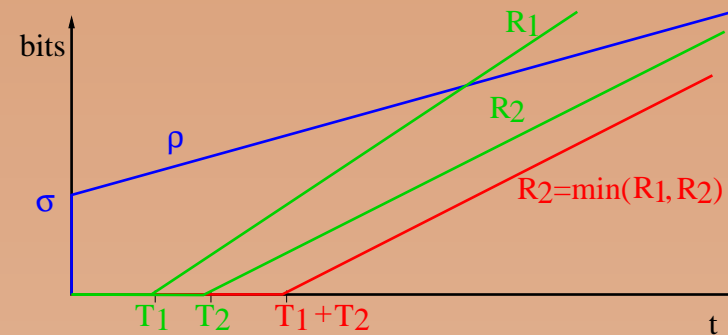
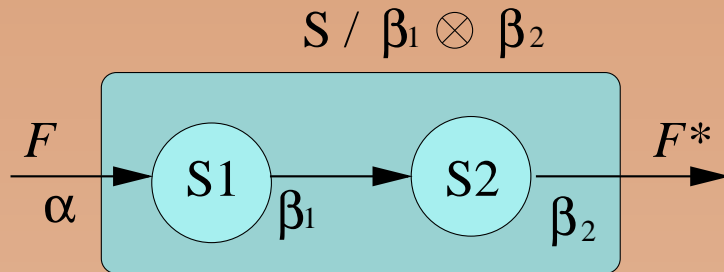
$$\alpha = \gamma_{\rho, \sigma}$$

$$\beta_1 = \beta_{R_1, T_1} = R_1 \max(0, t - T_1)$$

$$\beta_2 = \beta_{R_2, T_2} = R_2 \max(0, t - T_2)$$

$$\beta_{R_1, T_1} \otimes \beta_{R_2, T_2} = \beta_{\min(R_1, R_2), T_1 + T_2} = \min(R_1, R_2) \max(0, t - (T_1 + T_2))$$

## Network Calculus - Pay Bursts Only Once



$$\alpha = \gamma_{\rho, \sigma}$$

$$\beta_1 = \beta_{R_1, T_1} = R_1 \max(0, t - T_1)$$

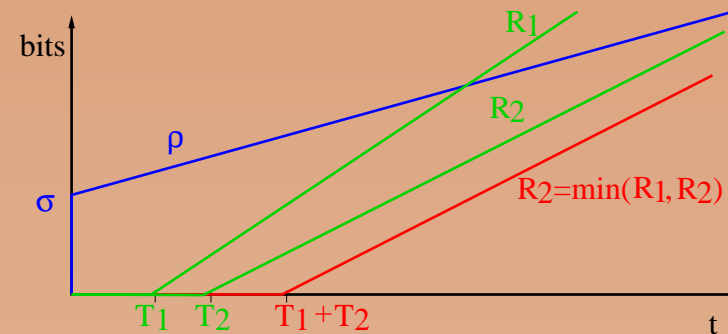
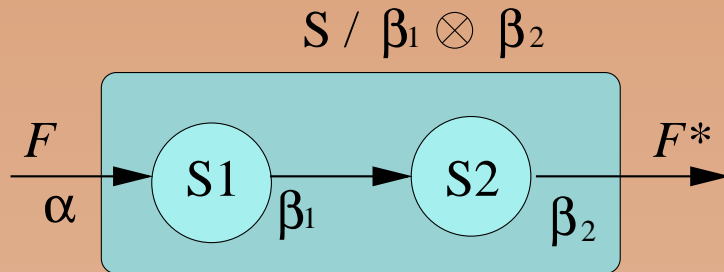
$$\beta_2 = \beta_{R_2, T_2} = R_2 \max(0, t - T_2)$$

$$\beta_{R_1, T_1} \otimes \beta_{R_2, T_2} = \beta_{\min(R_1, R_2), T_1 + T_2} = \min(R_1, R_2) \max(0, t - (T_1 + T_2))$$

$$D_1 + D_2 = \frac{\sigma}{R_1} + \frac{\sigma}{R_2} + \frac{\rho T_1}{R_2} + T_1 + T_2$$



## Network Calculus - Pay Bursts Only Once



$$\alpha = \gamma_{\rho, \sigma}$$

$$\beta_1 = \beta_{R_1, T_1} = R_1 \max(0, t - T_1)$$

$$\beta_2 = \beta_{R_2, T_2} = R_2 \max(0, t - T_2)$$

$$\beta_{R_1, T_1} \otimes \beta_{R_2, T_2} = \beta_{\min(R_1, R_2), T_1 + T_2} = \min(R_1, R_2) \max(0, t - (T_1 + T_2))$$

$$D_1 + D_2 = \frac{\sigma}{R_1} + \frac{\sigma}{R_2} + \frac{\rho T_1}{R_2} + T_1 + T_2$$

$$D_S = \frac{\sigma}{\min(R_1, R_2)} + T_1 + T_2$$



## Summary

- Communication Performance: bandwidth, unloaded latency, loaded latency
- Organizational Structure: NI, switch, link
- Topologies: wire space and delay domination favors low dimension topologies;
- Routing: deterministic vs source based vs adaptive routing; deadlock;
- Quality of Service and flow regulation



## To Probe Further

### Classic papers:

- [Agarwal, 1991] Agarwal, A. (1991). Limit on interconnection performance. *IEEE Transactions on Parallel and Distributed Systems*, 4(6):613–624.
- [Dally, 1990] Dally, W. J. (1990). Performance analysis of k-ary n-cube interconnection networks. *IEEE Transactions on Computers*, 39(6):775–785.

### Text books:

- [Duato et al., 1998] Duato, J., Yalamanchili, S., and Ni, L. (1998). *Interconnection Networks - An Engineering Approach*. Computer Society Press, Los Alamitos, California.
- [Culler et al., 1999] Culler, D. E., Singh, J. P., and Gupta, A. (1999). *Parallel Computer Architecture - A Hardware/Software Approach*. Morgan Kaufman Publishers.
- [Dally and Towels, 2004] Dally, W. J. and Towels, B. (2004). *Principles and Practices of Interconnection Networks*. Morgan Kaufman Publishers.
- [DeMicheli and Benini, 2006] DeMicheli, G. and Benini, L. (2006). *Networks on Chip*. Morgan Kaufmann.
- [Leighton, 1992] Leighton, F. T. (1992). *Introduction to Parallel Algorithms and Architectures*. Morgan Kaufmann, San Francisco.
- [LeBoudec, 200] Jean-Yves LeBoudec, J-Y. (2001). *Network Calculus*. Springer Verlag, LCNS

2050

