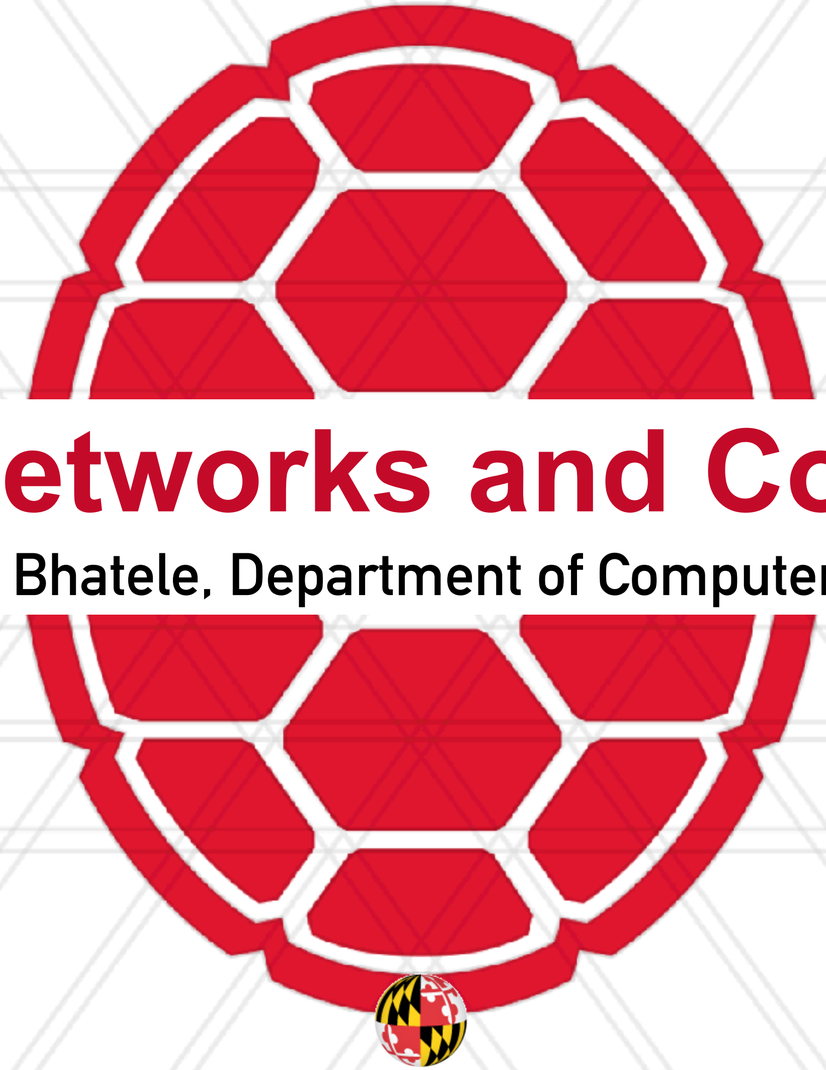
Introduction to Parallel Computing (CMSC498X / CMSC818X)



Lecture 20: Networks and Communication

Abhinav Bhatele, Department of Computer Science

UNIVERSITY OF MARYLAND

Announcements

- Assignment 3 posted online
  - Only for 818X students
  - Due on November 23
- Quiz 2: November 12

High-speed interconnection networks

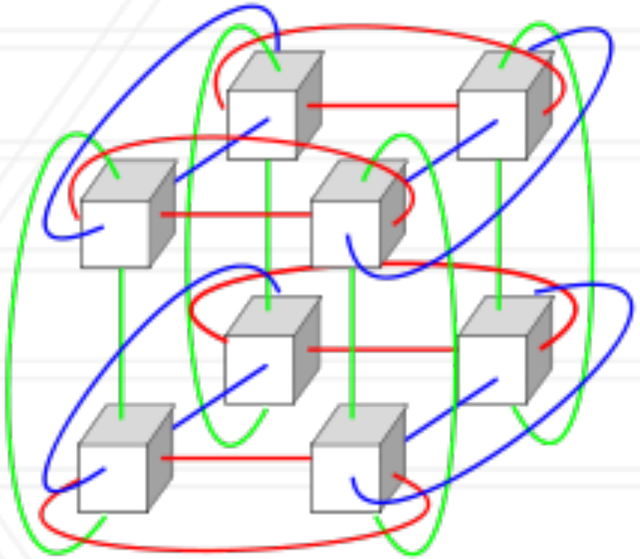
- Typically supercomputers and HPC clusters are connected by low latency and high bandwidth networks
- The connections between nodes form different topologies
- Popular topologies:
  - Fat-tree: Charles Leiserson in 1985
  - Mesh and torus networks
  - Dragonfly networks

Network components

- Network interface controller or card
- Router or switch
- Network cables: copper or optical

N-dimensional mesh / torus networks

- Each switch as a small number of nodes connected to it (typically 1)
- Each switch has direct links to 2n switches where n is the number of dimensions
- Torus = wraparound links
- Examples: IBM Blue Gene, Cray X* machines



Fat-tree network

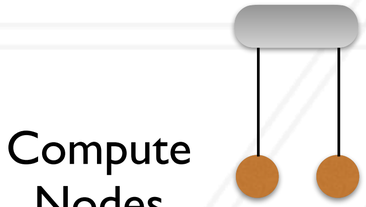
- Router radix = k, Number of nodes on each router = k/2
- A pod is a group of k/2 switches, Max. number of pods = k

Fat-tree network

- Router radix = k, Number of nodes on each router = k/2
- A pod is a group of k/2 switches, Max. number of pods = k

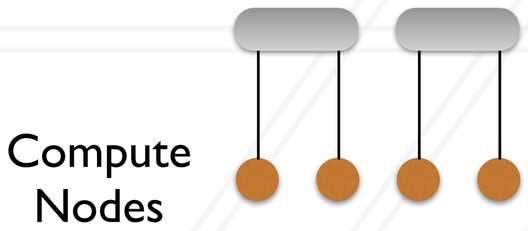
Fat-tree network

- Router radix = k, Number of nodes on each router = k/2
- A pod is a group of k/2 switches, Max. number of pods = k



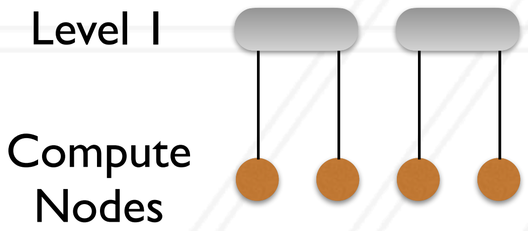
Fat-tree network

- Router radix = k, Number of nodes on each router = k/2
- A pod is a group of k/2 switches, Max. number of pods = k



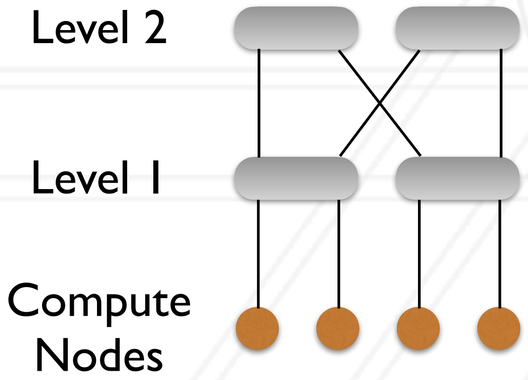
Fat-tree network

- Router radix = k, Number of nodes on each router = k/2
- A pod is a group of k/2 switches, Max. number of pods = k



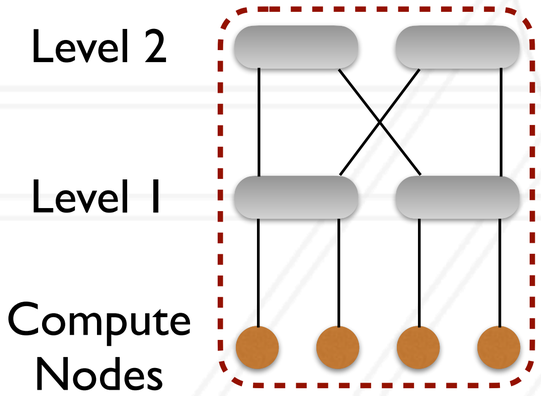
Fat-tree network

- Router radix = k, Number of nodes on each router = k/2
- A pod is a group of k/2 switches, Max. number of pods = k



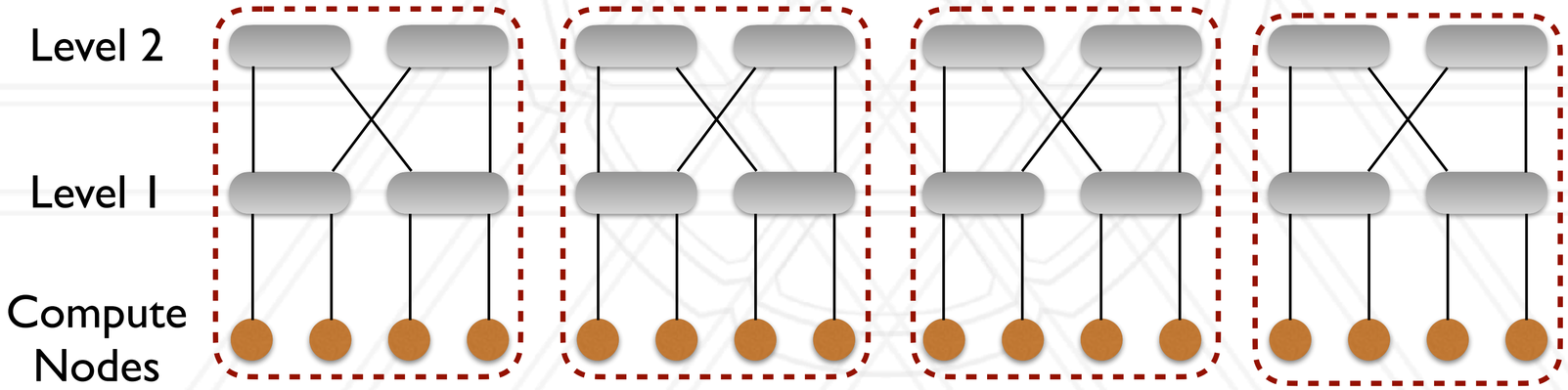
Fat-tree network

- Router radix = k, Number of nodes on each router = k/2
- A pod is a group of k/2 switches, Max. number of pods = k



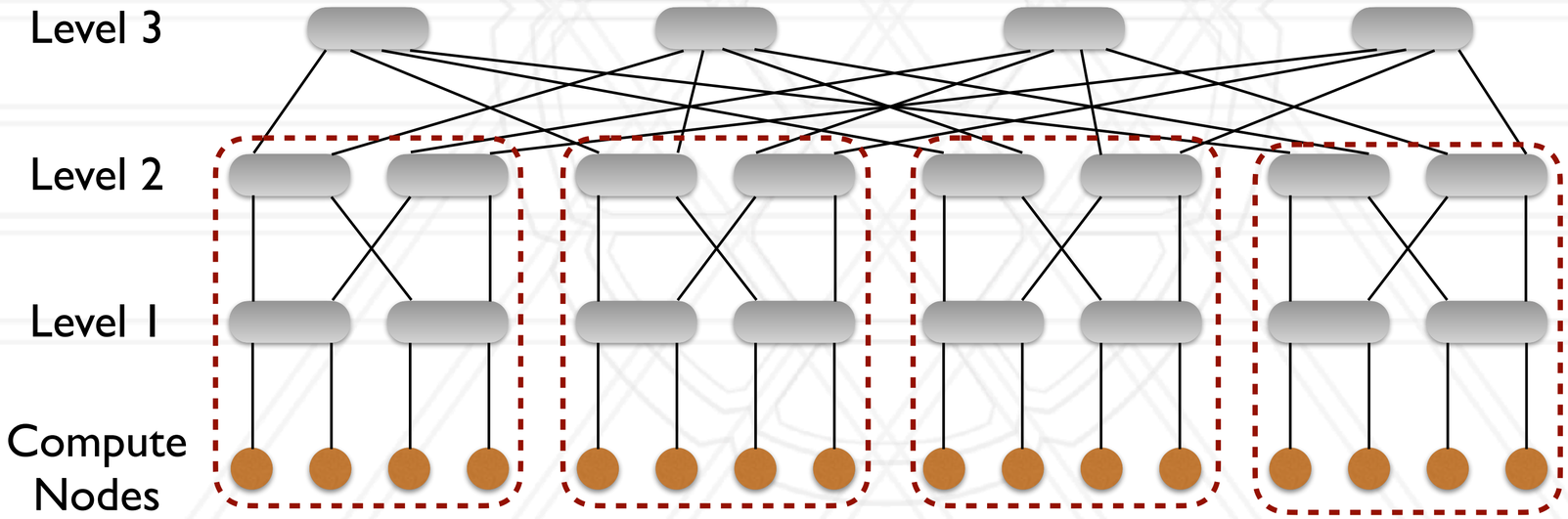
Fat-tree network

- Router radix = k, Number of nodes on each router = k/2
- A pod is a group of k/2 switches, Max. number of pods = k



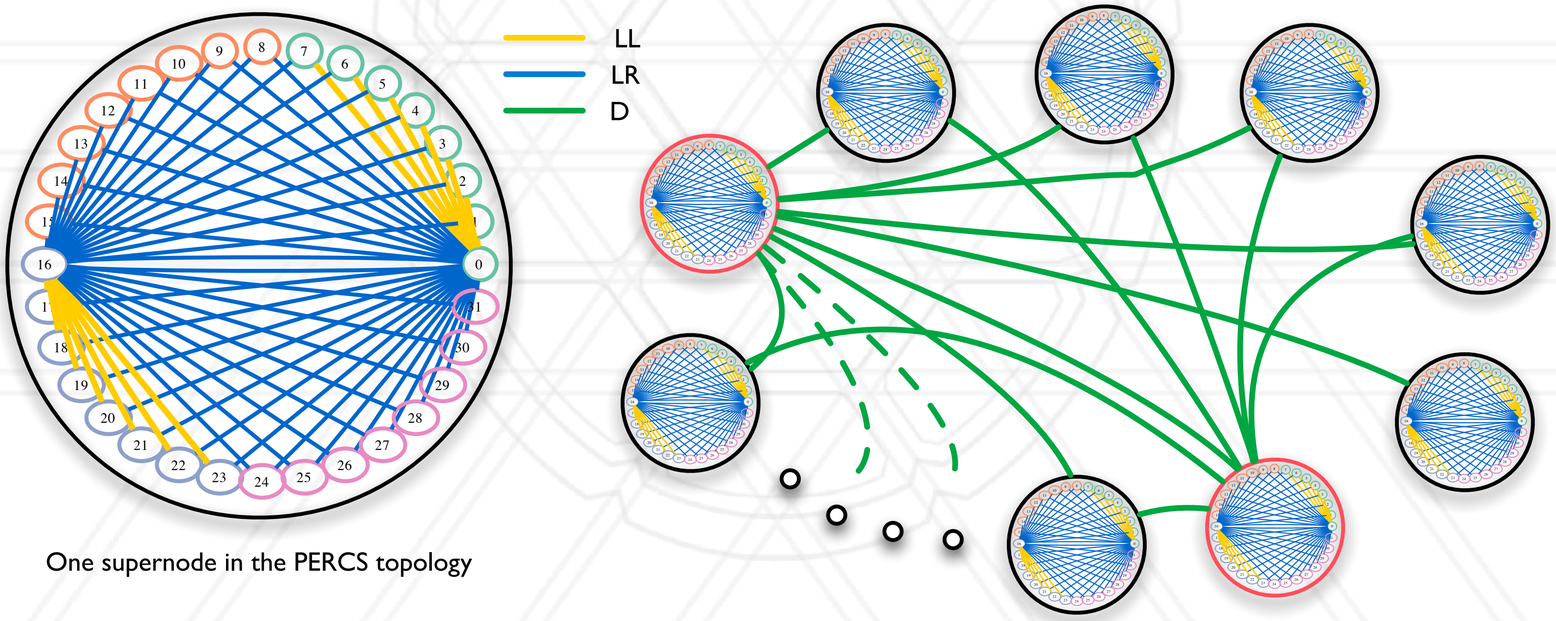
Fat-tree network

- Router radix = k, Number of nodes on each router = k/2
- A pod is a group of k/2 switches, Max. number of pods = k



Dragonfly network

- Two-level hierarchical network using high-radix routers
- Low network diameter



One supernode in the PERCS topology

Life-cycle of a message

Source

Source

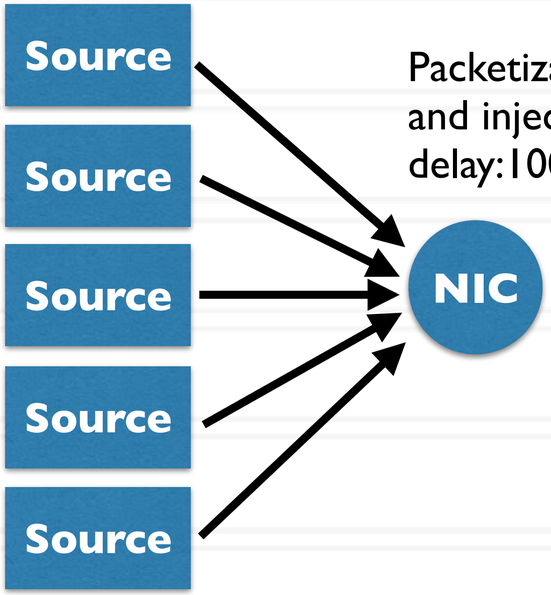
Source

Source

Source

Message origin points : destination, frequency, size, etc. determined by application
1 micro sec - 10s of sec

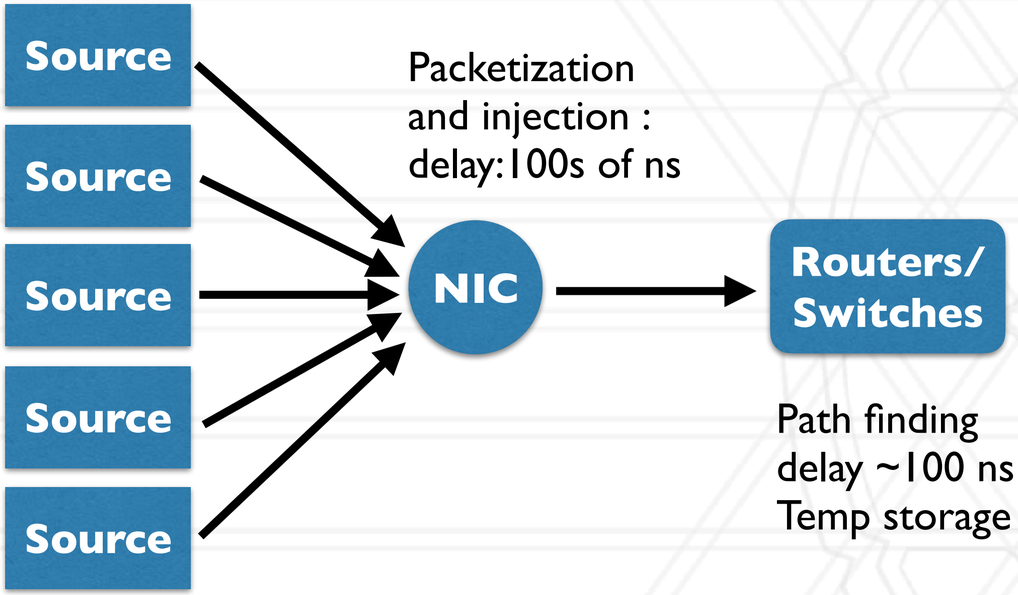
Life-cycle of a message



Packetization
and injection :
delay: 100s of ns

Message origin points :
destination, frequency,
size, etc. determined
by application
1 micro sec - 10s of sec

Life-cycle of a message



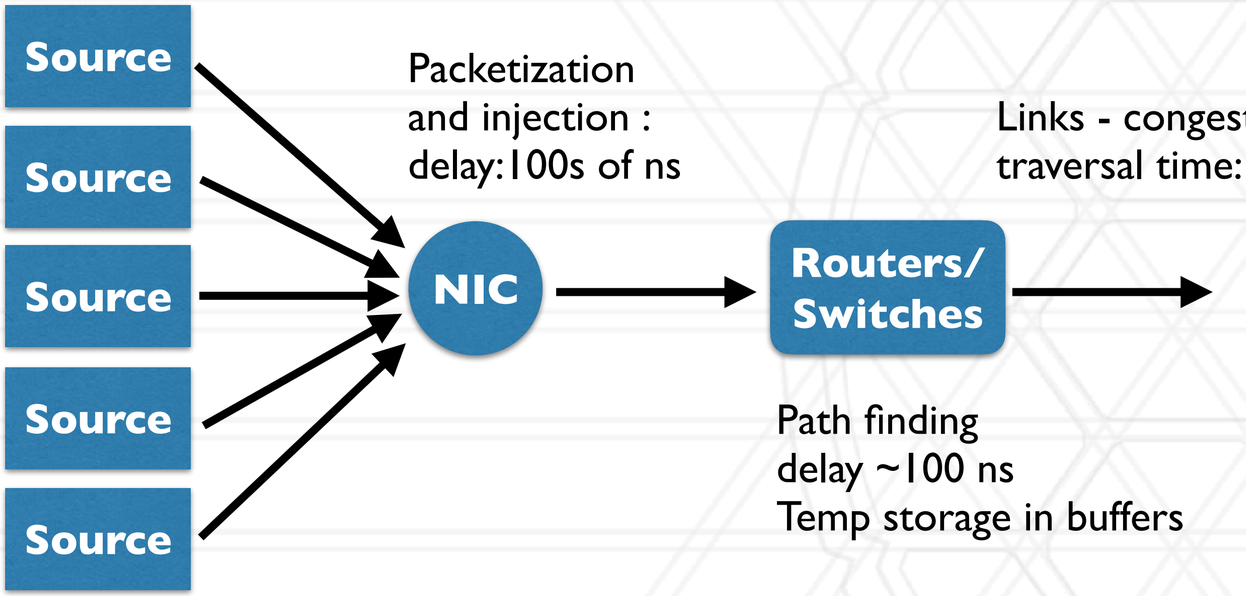
Packetization
and injection :
delay:100s of ns

Routers/
Switches

Path finding
delay ~100 ns
Temp storage in buffers

Message origin points :
destination, frequency,
size, etc. determined
by application
1 micro sec - 10s of sec

Life-cycle of a message



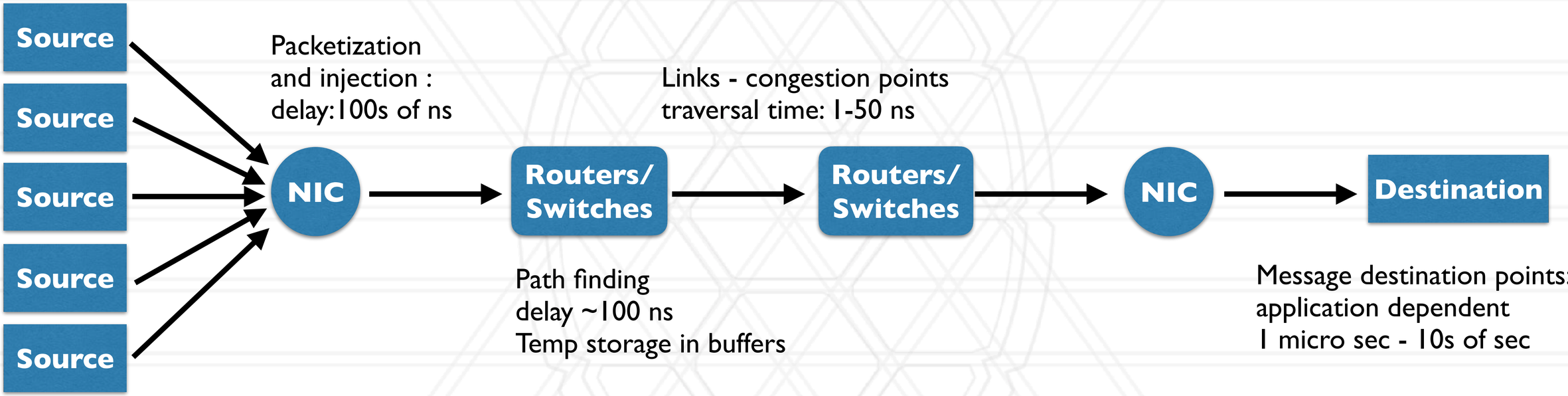
Packetization
and injection :
delay:100s of ns

Links - congestion points
traversal time: 1-50 ns

Path finding
delay ~100 ns
Temp storage in buffers

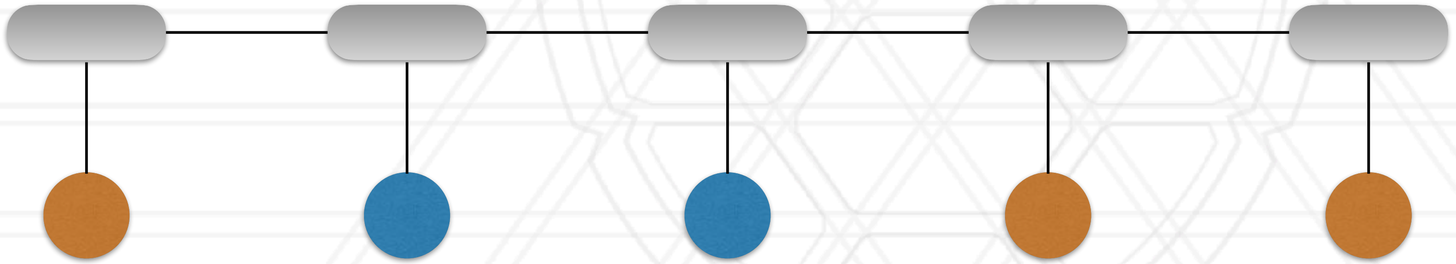
Message origin points :
destination, frequency,
size, etc. determined
by application
1 micro sec - 10s of sec

Life-cycle of a message



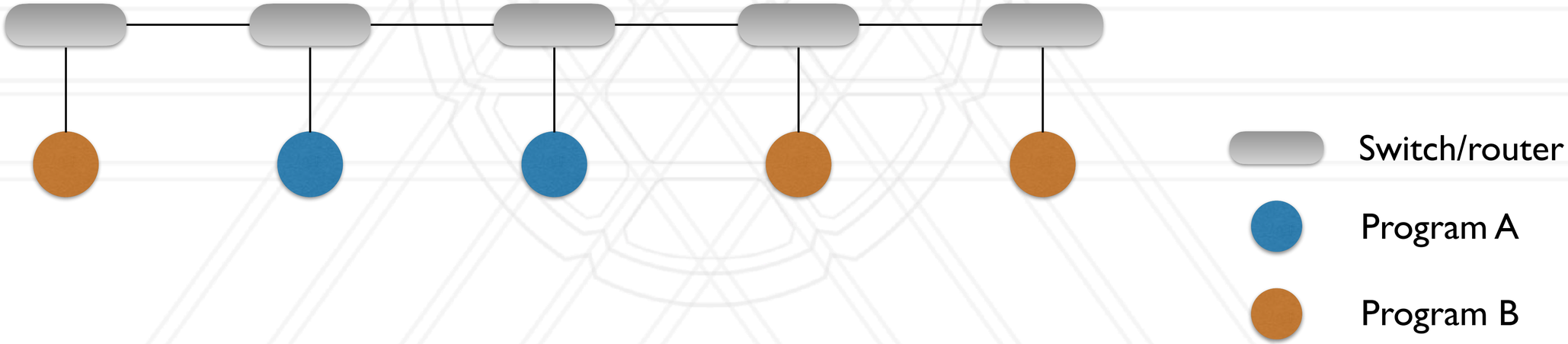
Congestion due to network sharing

- Sharing refers to network flows of different programs using the same hardware resources: links, switches
- When multiple programs communicate on the network, they all suffer from congestion on shared links



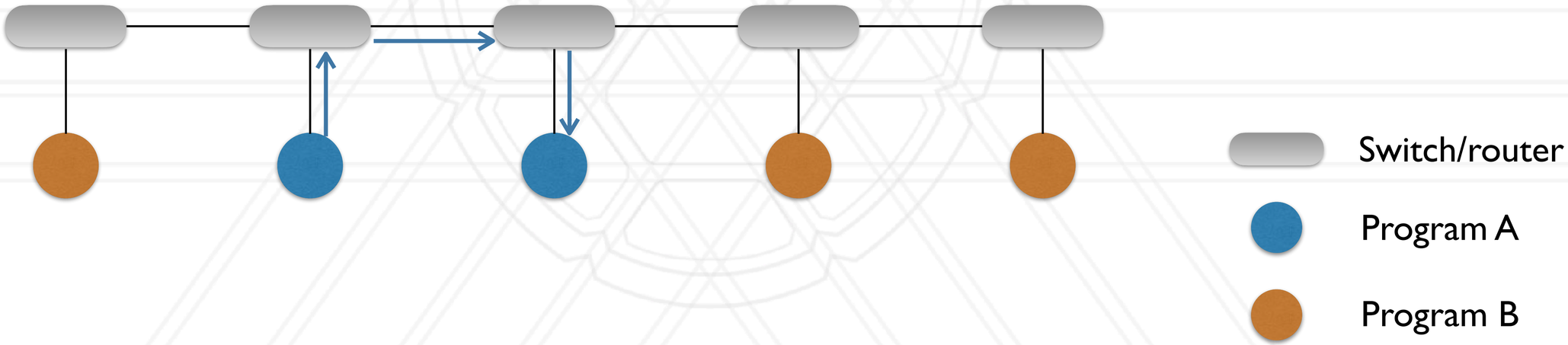
Congestion due to network sharing

- Sharing refers to network flows of different programs using the same hardware resources: links, switches
- When multiple programs communicate on the network, they all suffer from congestion on shared links



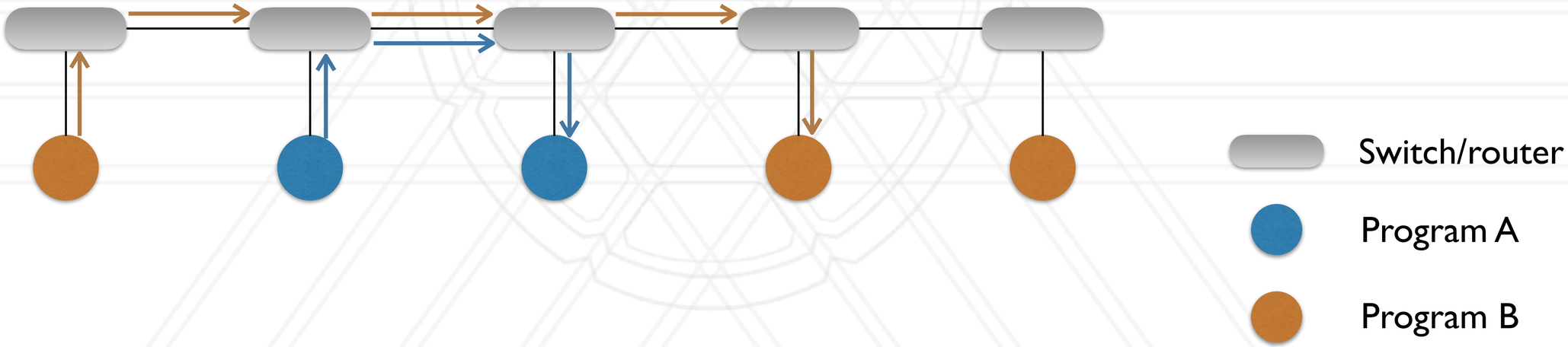
Congestion due to network sharing

- Sharing refers to network flows of different programs using the same hardware resources: links, switches
- When multiple programs communicate on the network, they all suffer from congestion on shared links



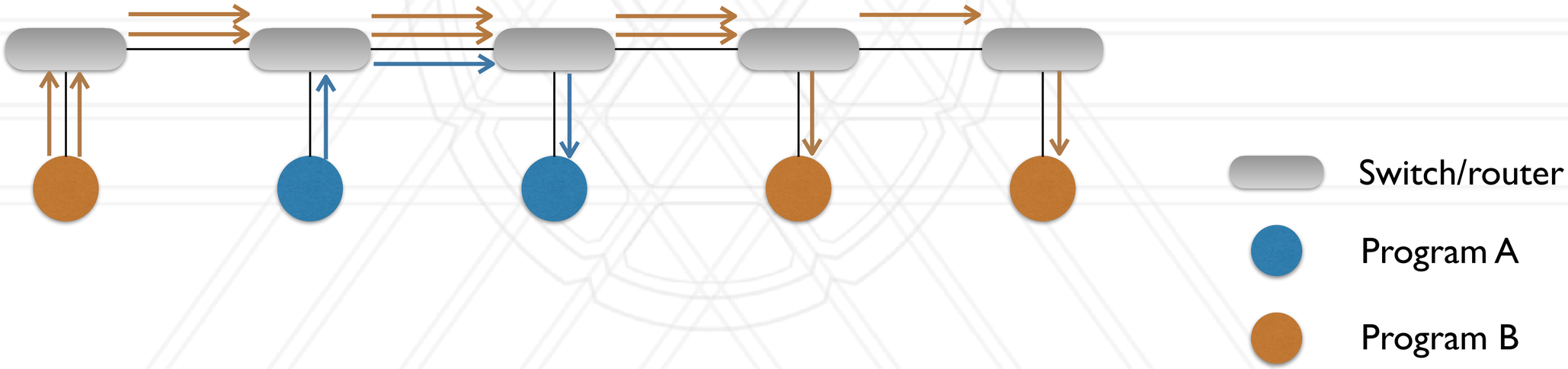
Congestion due to network sharing

- Sharing refers to network flows of different programs using the same hardware resources: links, switches
- When multiple programs communicate on the network, they all suffer from congestion on shared links



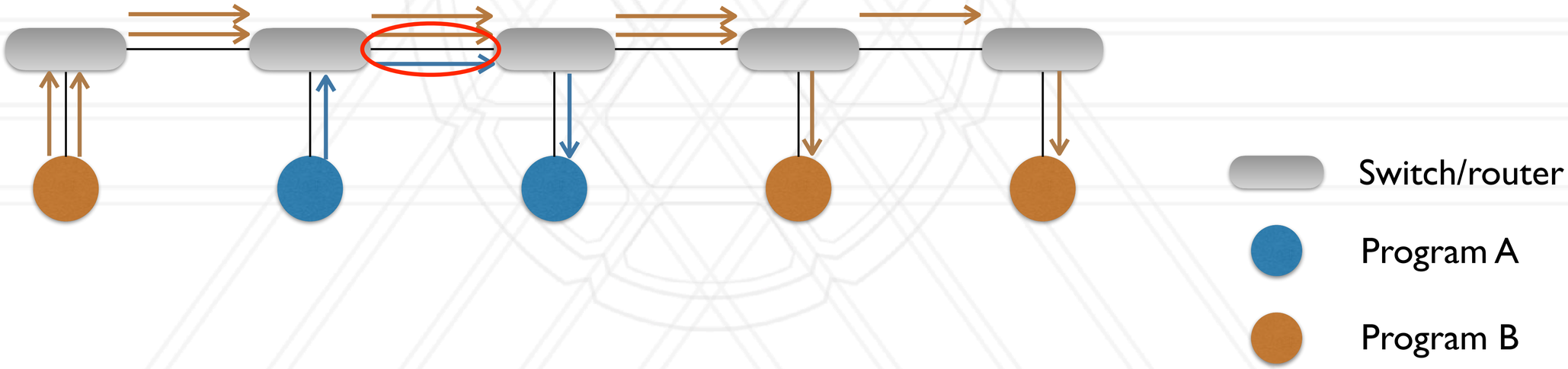
Congestion due to network sharing

- Sharing refers to network flows of different programs using the same hardware resources: links, switches
- When multiple programs communicate on the network, they all suffer from congestion on shared links



Congestion due to network sharing

- Sharing refers to network flows of different programs using the same hardware resources: links, switches
- When multiple programs communicate on the network, they all suffer from congestion on shared links



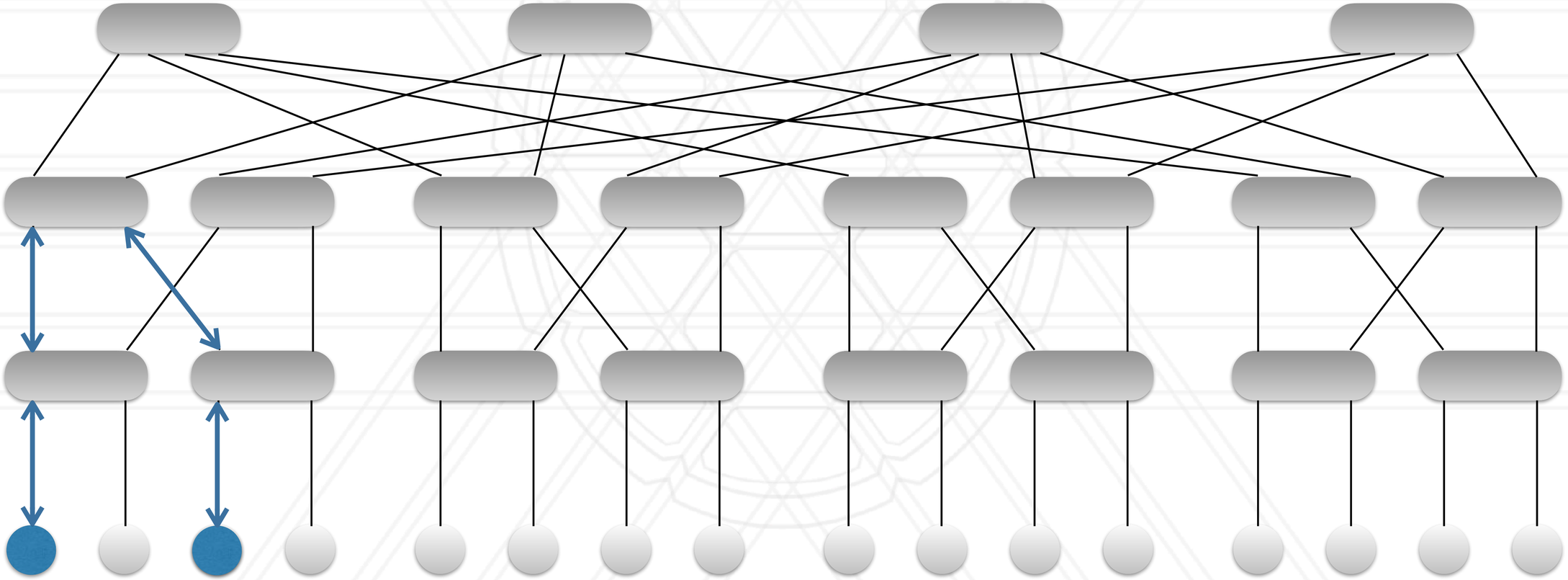
Routing algorithm

- Decides how a packet is routed between a source and destination switch
- Static routing: each router is pre-programmed with a routing table
  - Can change it at boot time
- Dynamic routing: routing can change at runtime
- Adaptive routing: adapts to network congestion

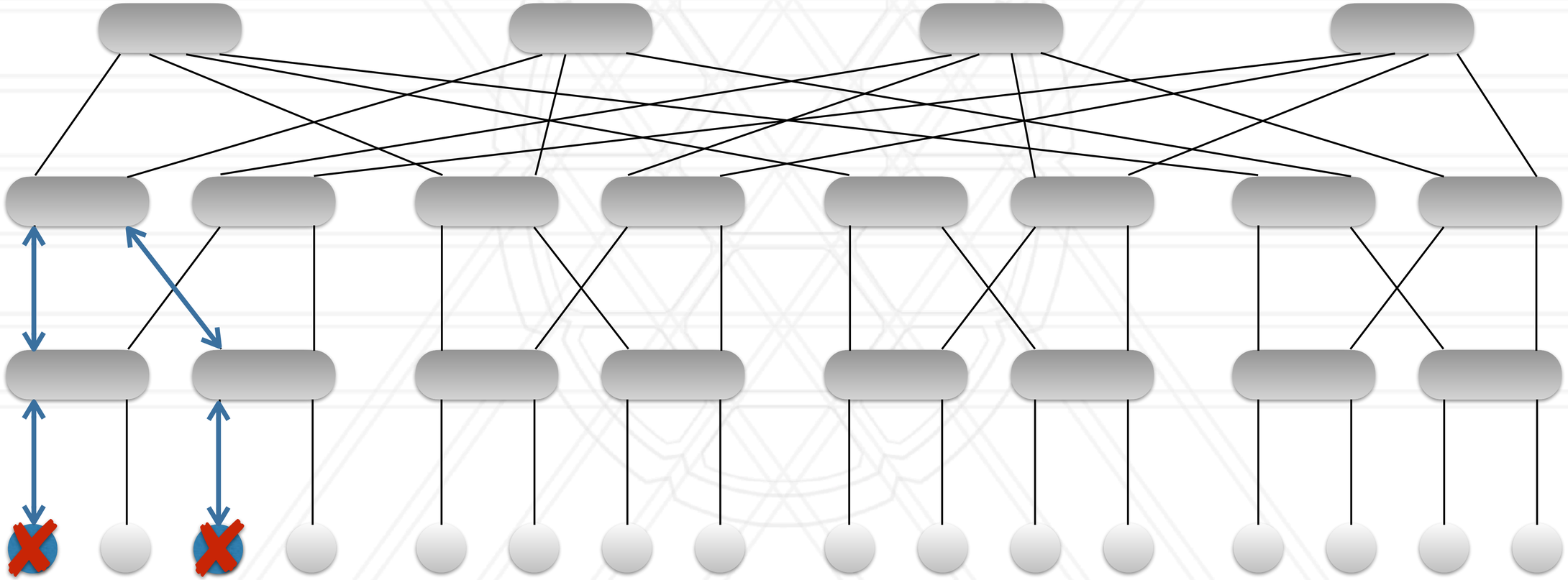
Different approaches to mitigating congestion

- Network topology aware node allocation
- Congestion or network flow aware adaptive routing
- Within a job: network topology aware mapping of processes or chares to allocated nodes

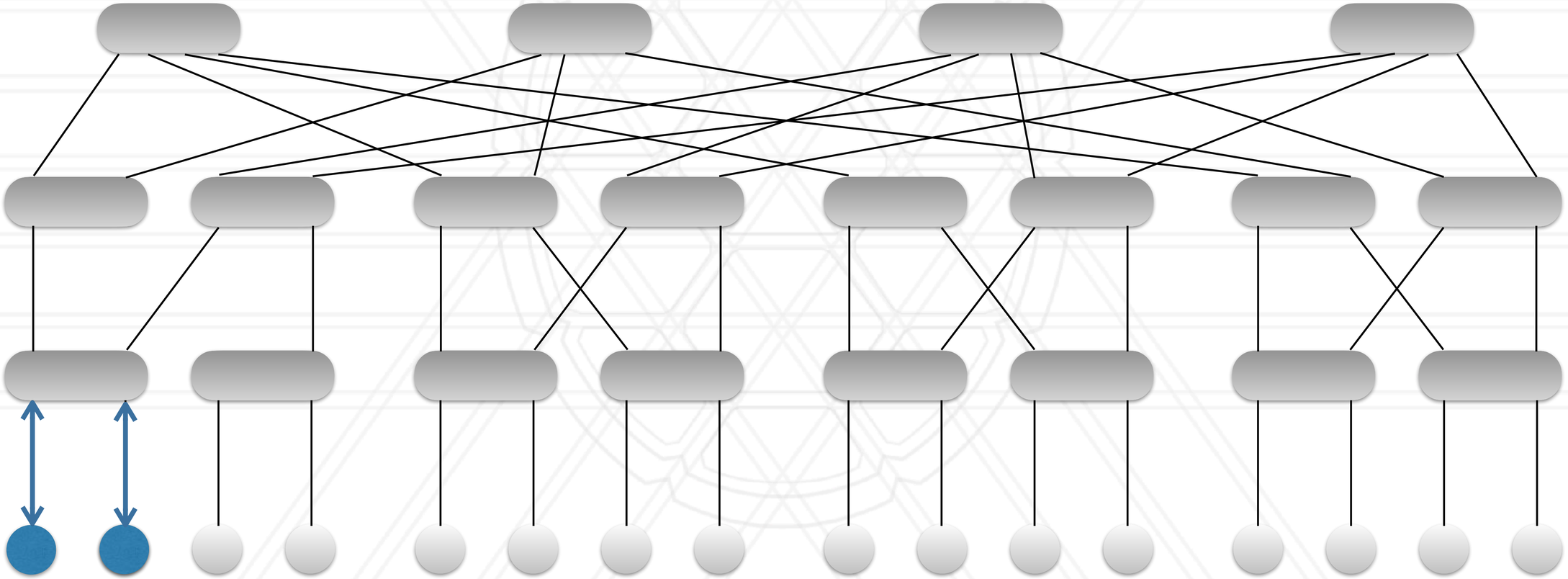
topology-aware node allocation



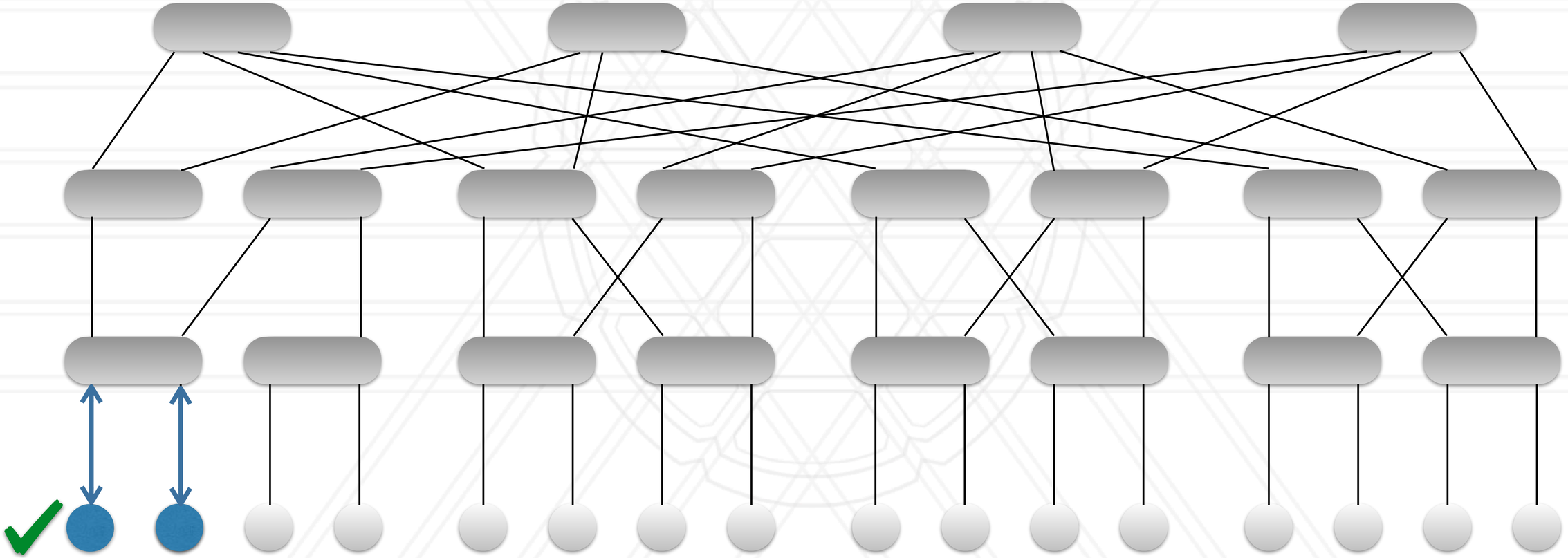
topology-aware node allocation



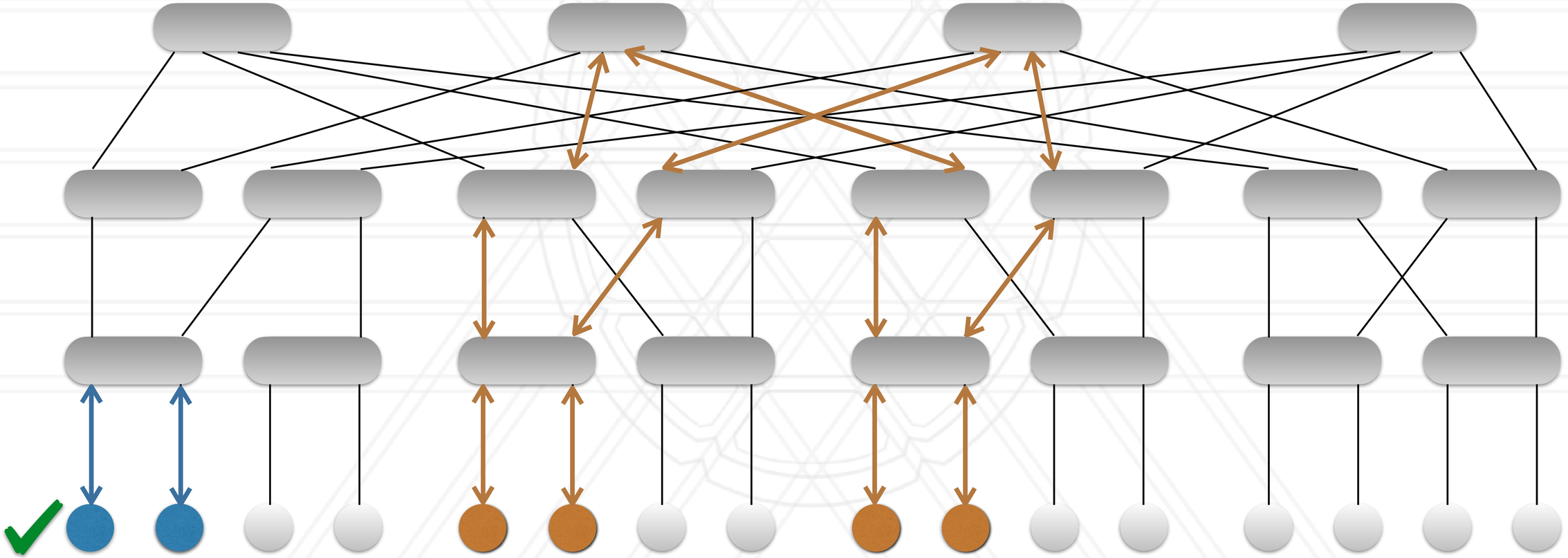
topology-aware node allocation



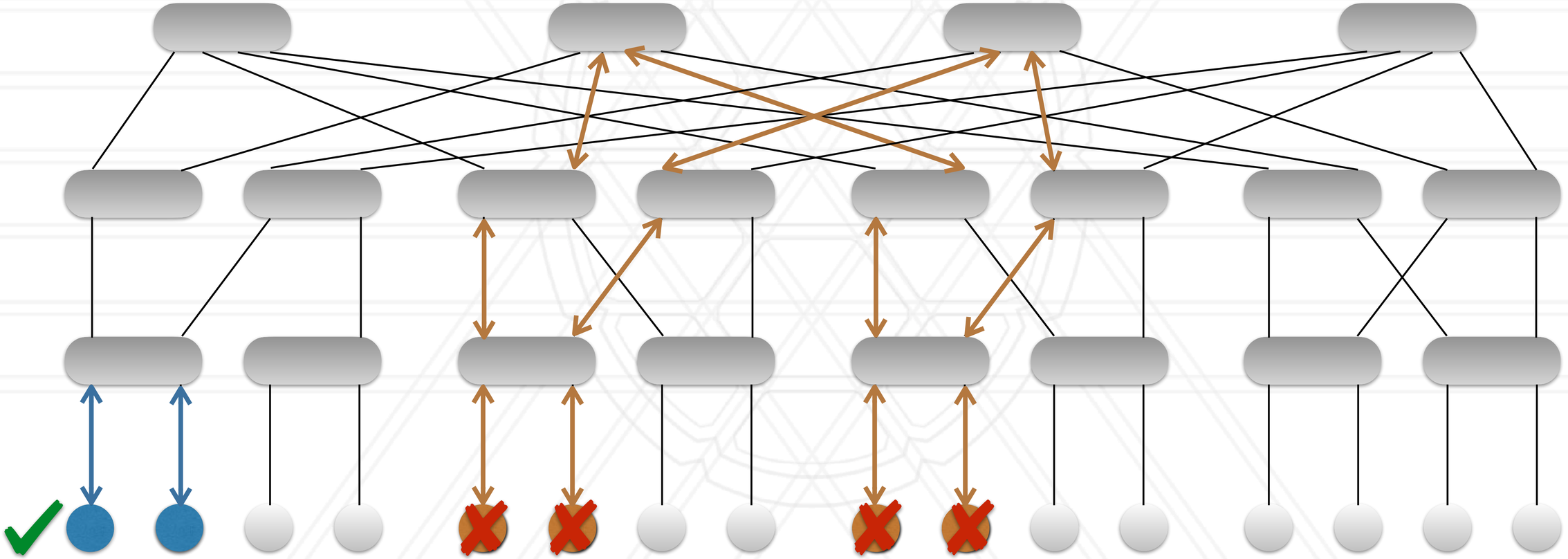
topology-aware node allocation



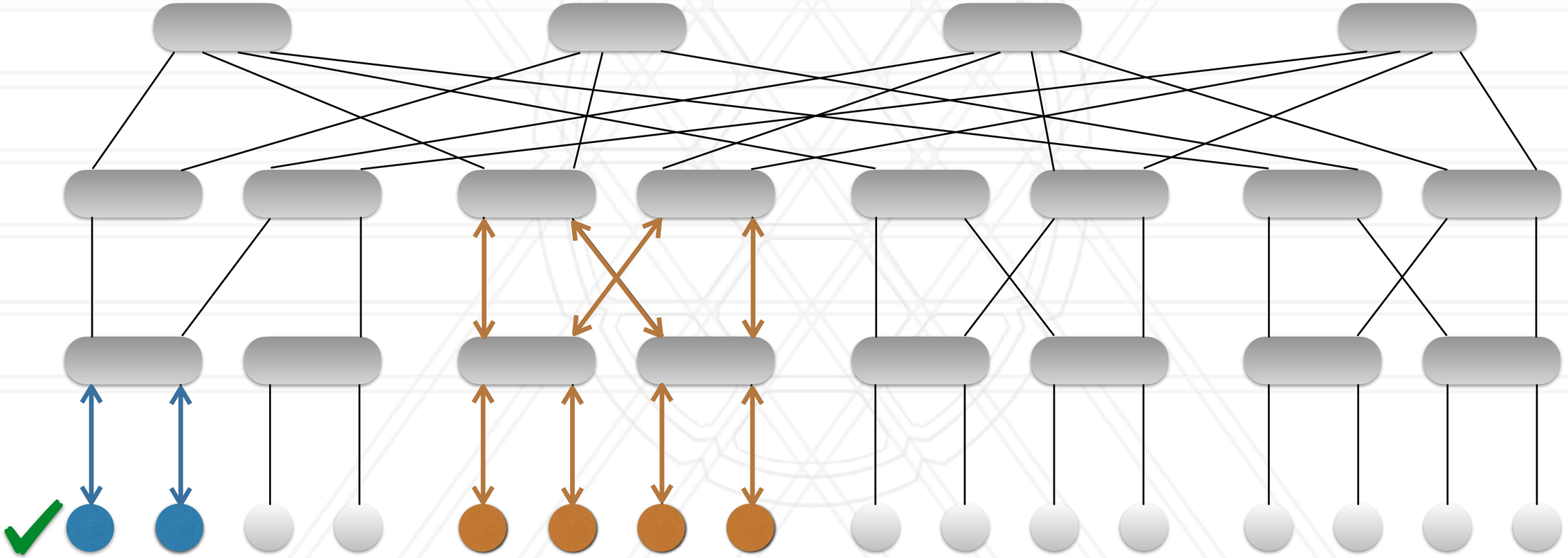
topology-aware node allocation



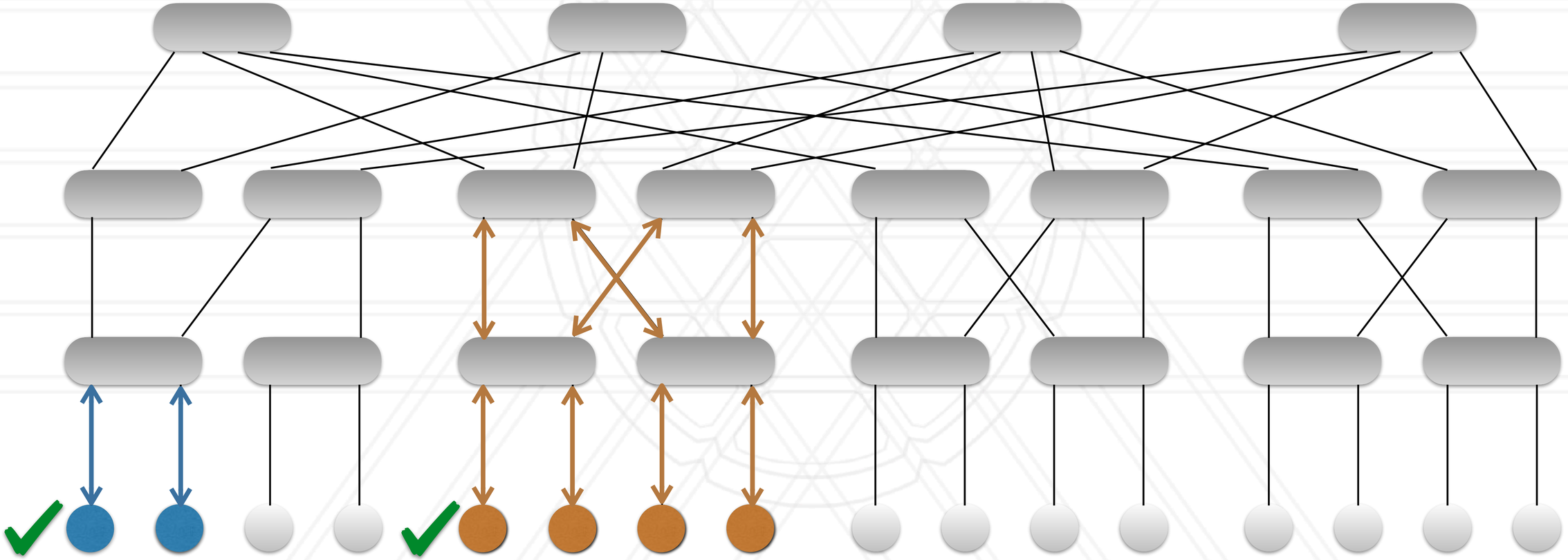
topology-aware node allocation



topology-aware node allocation

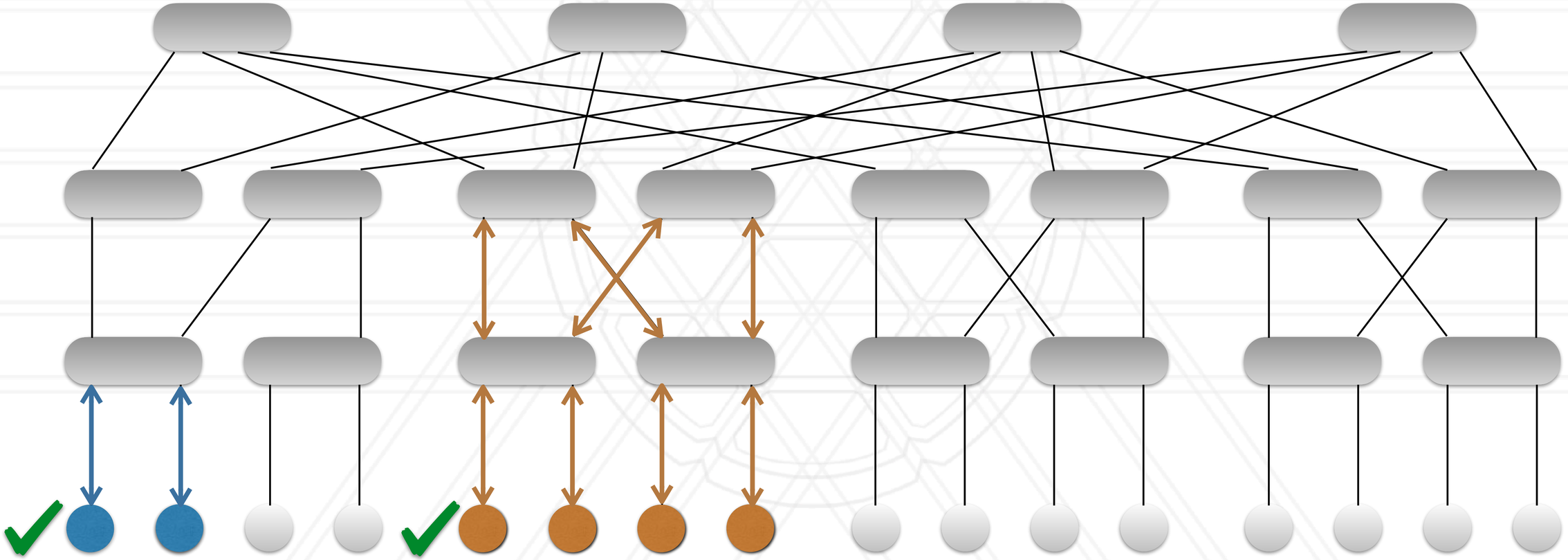


topology-aware node allocation

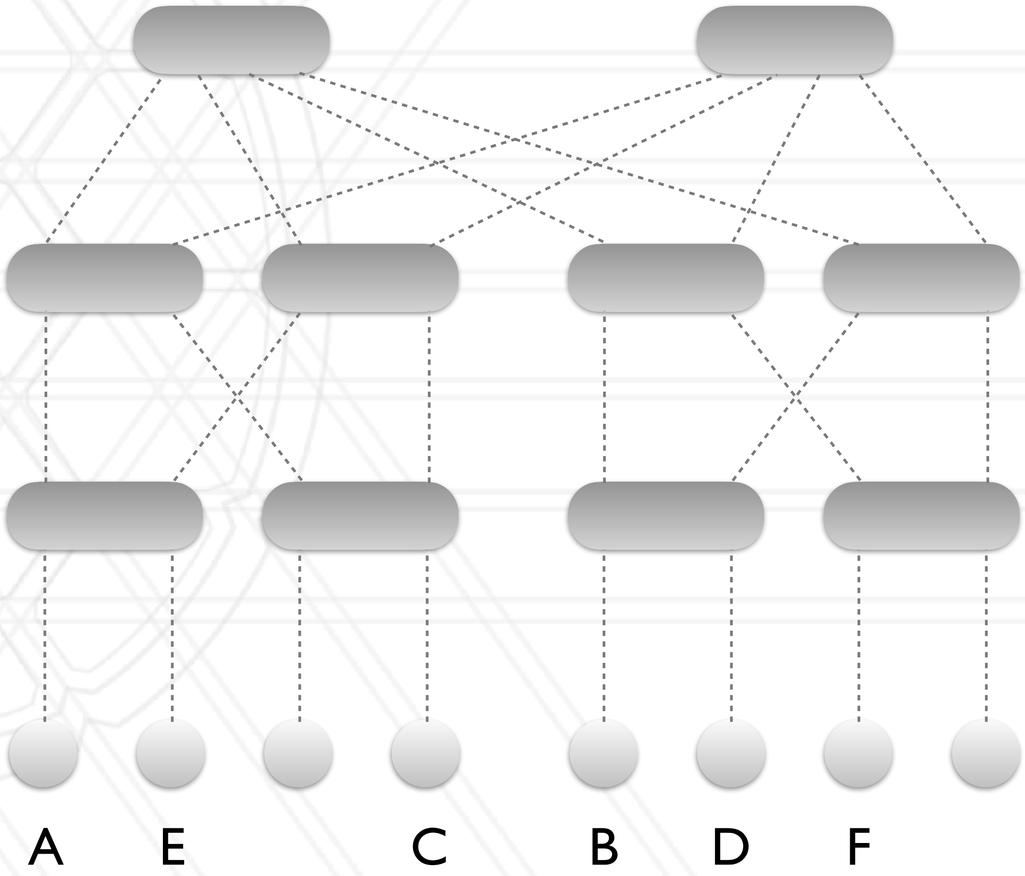


topology-aware node allocation

Solution: allocate nodes in a manner that prevents sharing of links by multiple jobs while maintaining high utilization



AFAR: adaptive flow aware routing

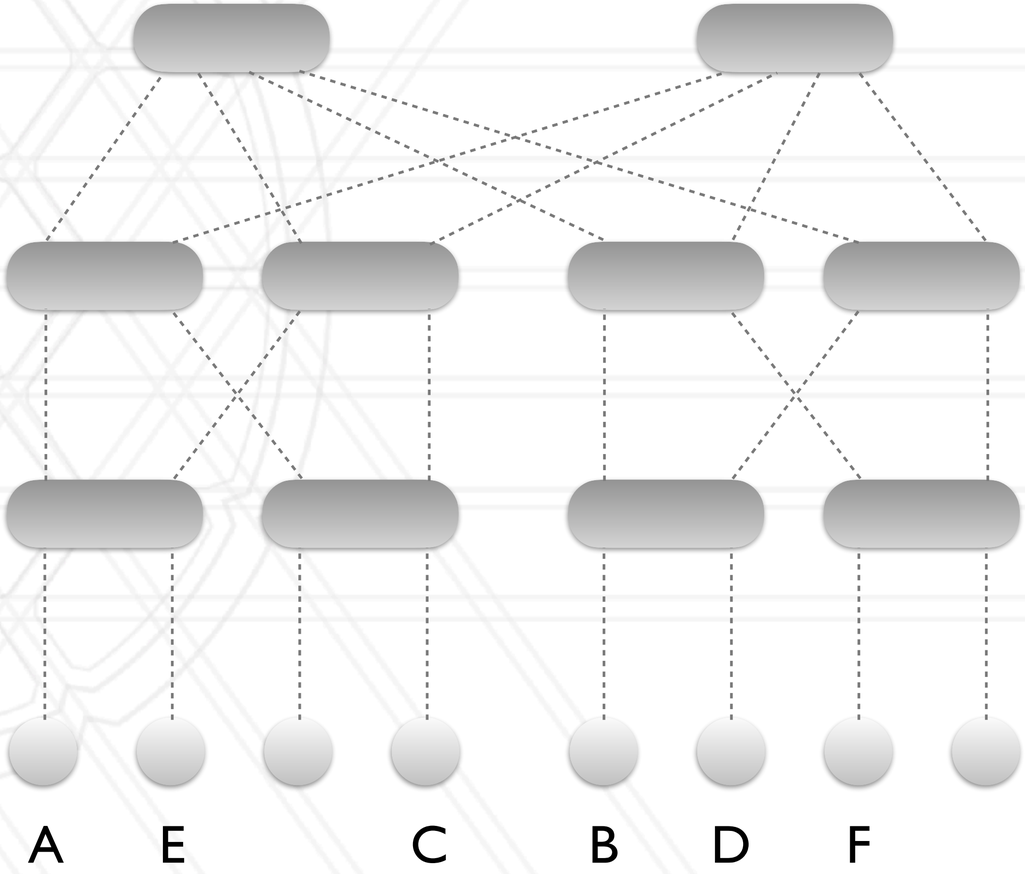


AFAR: adaptive flow aware routing

Solution: dynamically re-route traffic to alleviate hot-spots

Given: traffic for each pair of nodes in the system and the current routing

1. Calculate current load (network traffic) on all links in system
2. Find link with maximum load
3. If maximum > threshold, re-route one flow crossing that link to an under-utilized link
4. Repeat from 1. using new routing

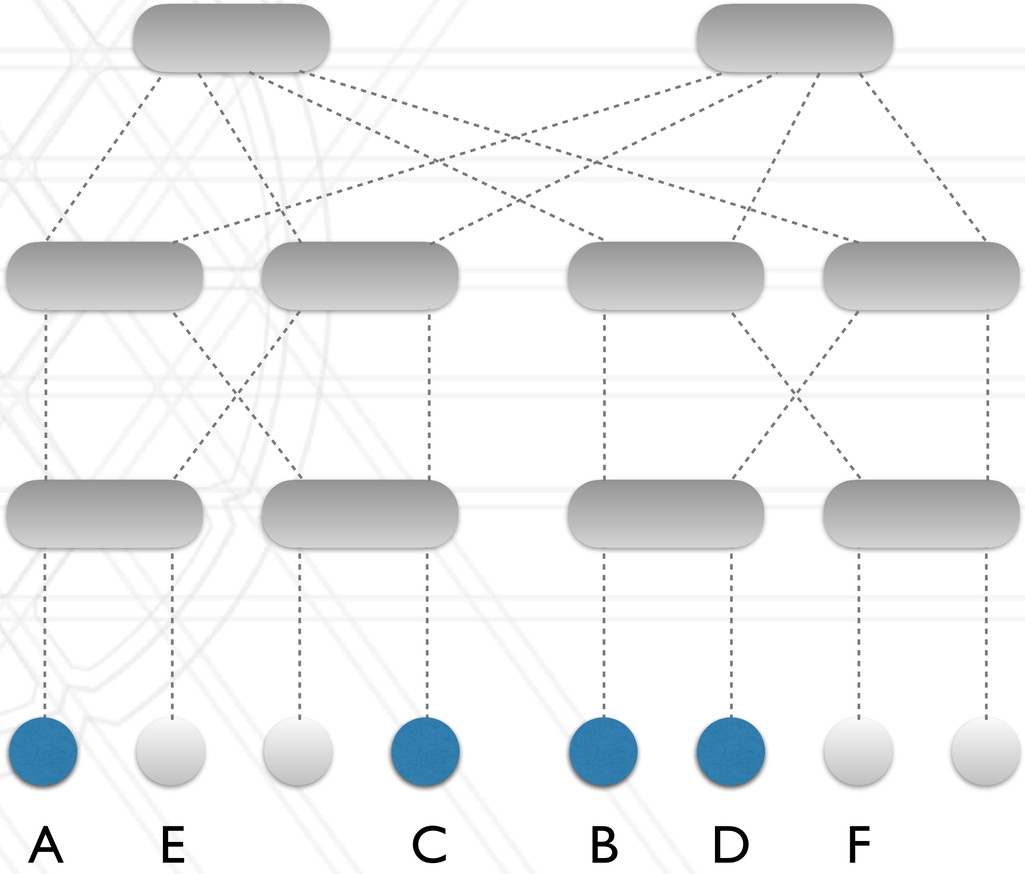


AFAR: adaptive flow aware routing

Solution: dynamically re-route traffic to alleviate hot-spots

Given: traffic for each pair of nodes in the system and the current routing

1. Calculate current load (network traffic) on all links in system
2. Find link with maximum load
3. If maximum > threshold, re-route one flow crossing that link to an under-utilized link
4. Repeat from 1. using new routing

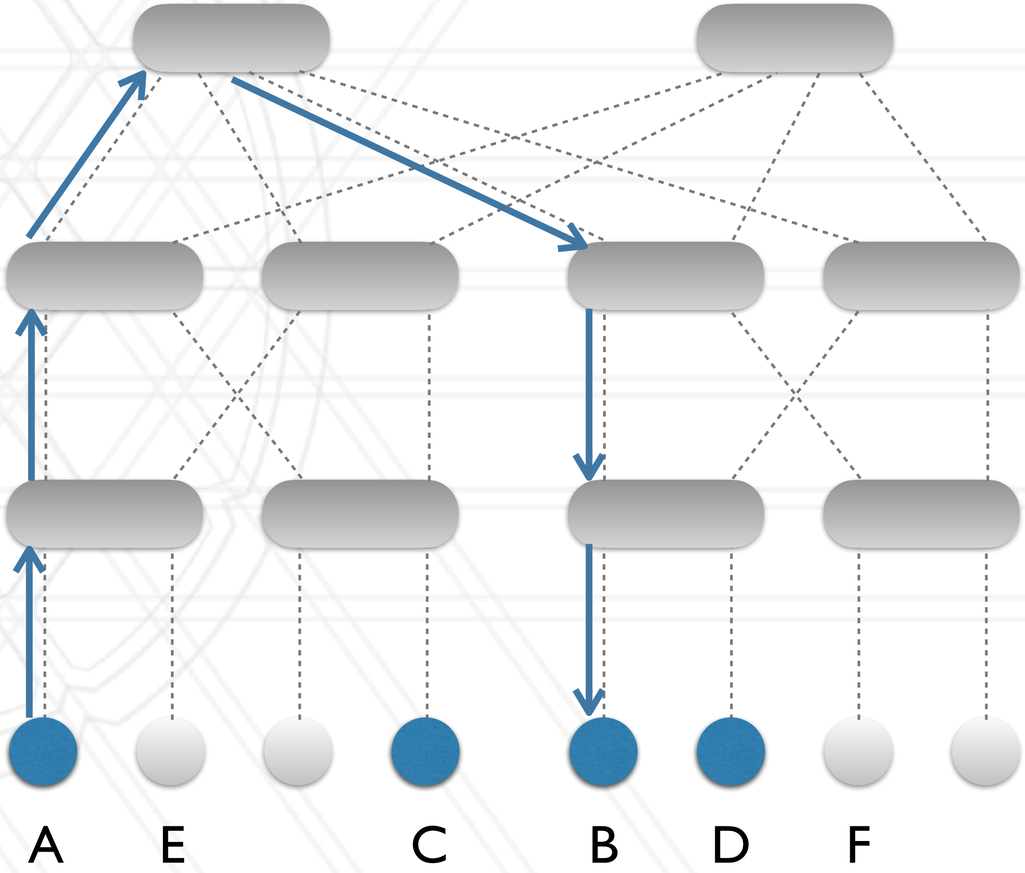


AFAR: adaptive flow aware routing

Solution: dynamically re-route traffic to alleviate hot-spots

Given: traffic for each pair of nodes in the system and the current routing

1. Calculate current load (network traffic) on all links in system
2. Find link with maximum load
3. If maximum > threshold, re-route one flow crossing that link to an under-utilized link
4. Repeat from 1. using new routing

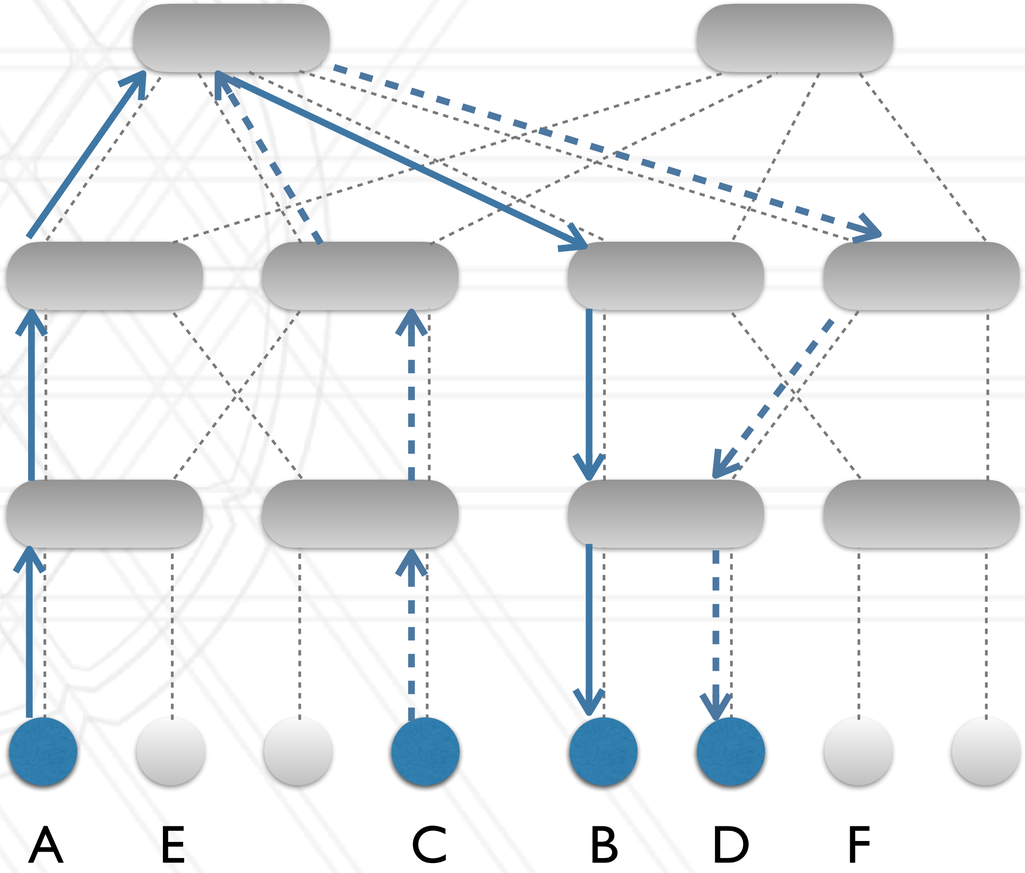


AFAR: adaptive flow aware routing

Solution: dynamically re-route traffic to alleviate hot-spots

Given: traffic for each pair of nodes in the system and the current routing

1. Calculate current load (network traffic) on all links in system
2. Find link with maximum load
3. If maximum > threshold, re-route one flow crossing that link to an under-utilized link
4. Repeat from 1. using new routing

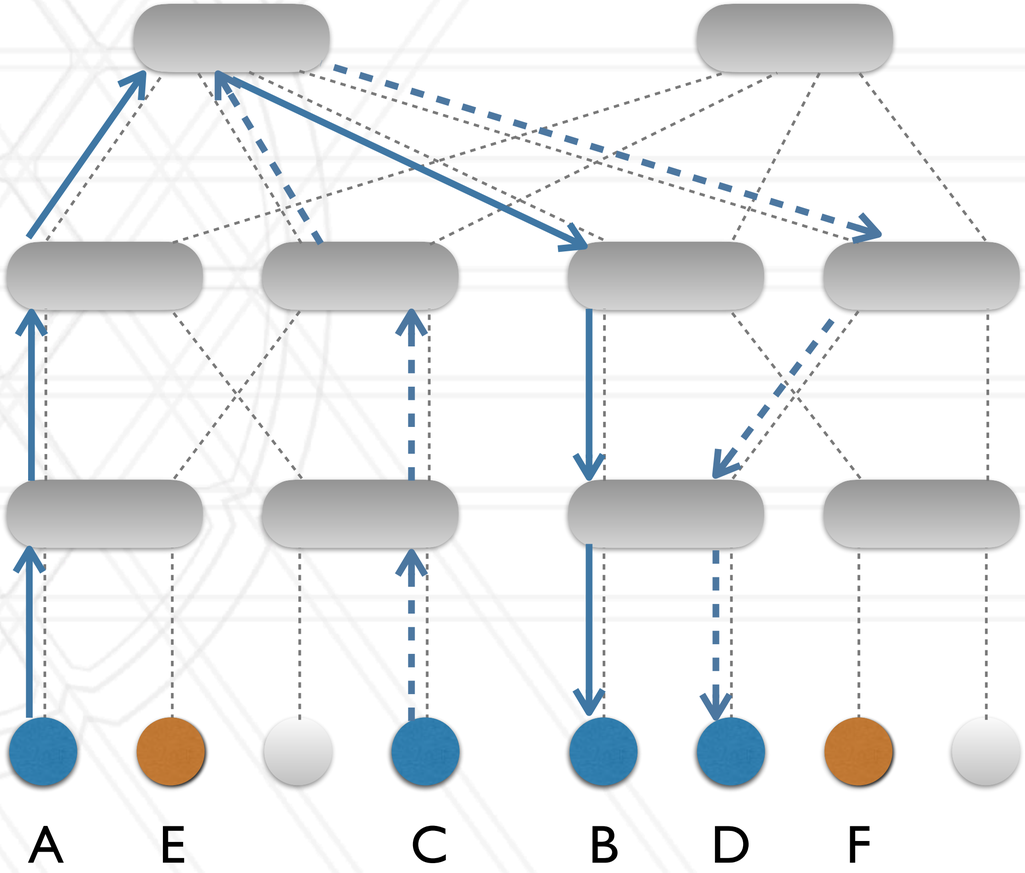


AFAR: adaptive flow aware routing

Solution: dynamically re-route traffic to alleviate hot-spots

Given: traffic for each pair of nodes in the system and the current routing

1. Calculate current load (network traffic) on all links in system
2. Find link with maximum load
3. If maximum > threshold, re-route one flow crossing that link to an under-utilized link
4. Repeat from 1. using new routing

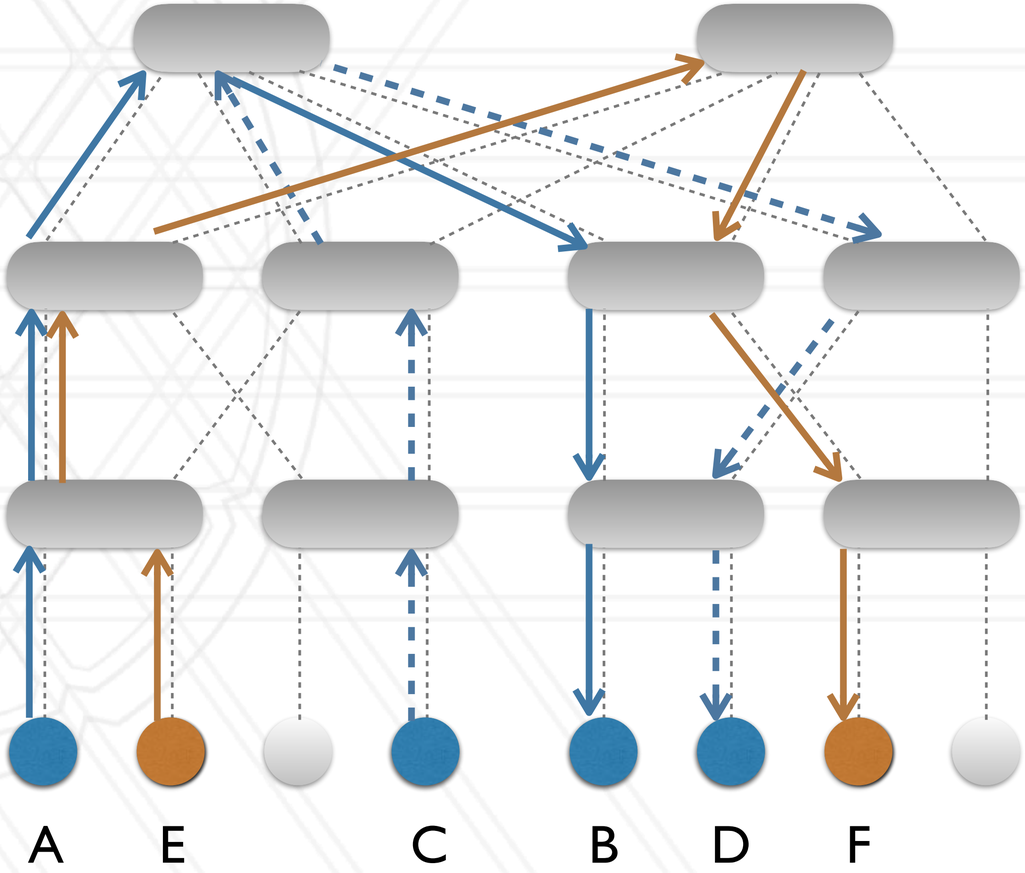


AFAR: adaptive flow aware routing

Solution: dynamically re-route traffic to alleviate hot-spots

Given: traffic for each pair of nodes in the system and the current routing

1. Calculate current load (network traffic) on all links in system
2. Find link with maximum load
3. If maximum > threshold, re-route one flow crossing that link to an under-utilized link
4. Repeat from 1. using new routing

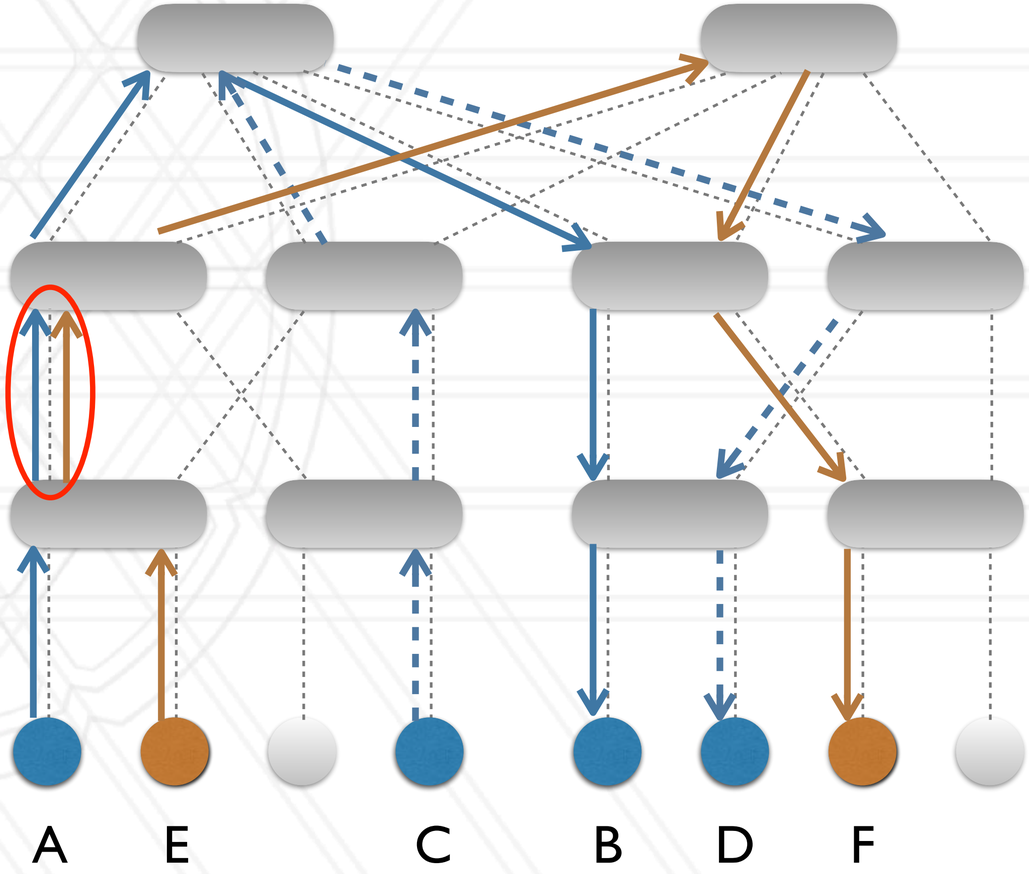


AFAR: adaptive flow aware routing

Solution: dynamically re-route traffic to alleviate hot-spots

Given: traffic for each pair of nodes in the system and the current routing

1. Calculate current load (network traffic) on all links in system
2. Find link with maximum load
3. If maximum > threshold, re-route one flow crossing that link to an under-utilized link
4. Repeat from 1. using new routing

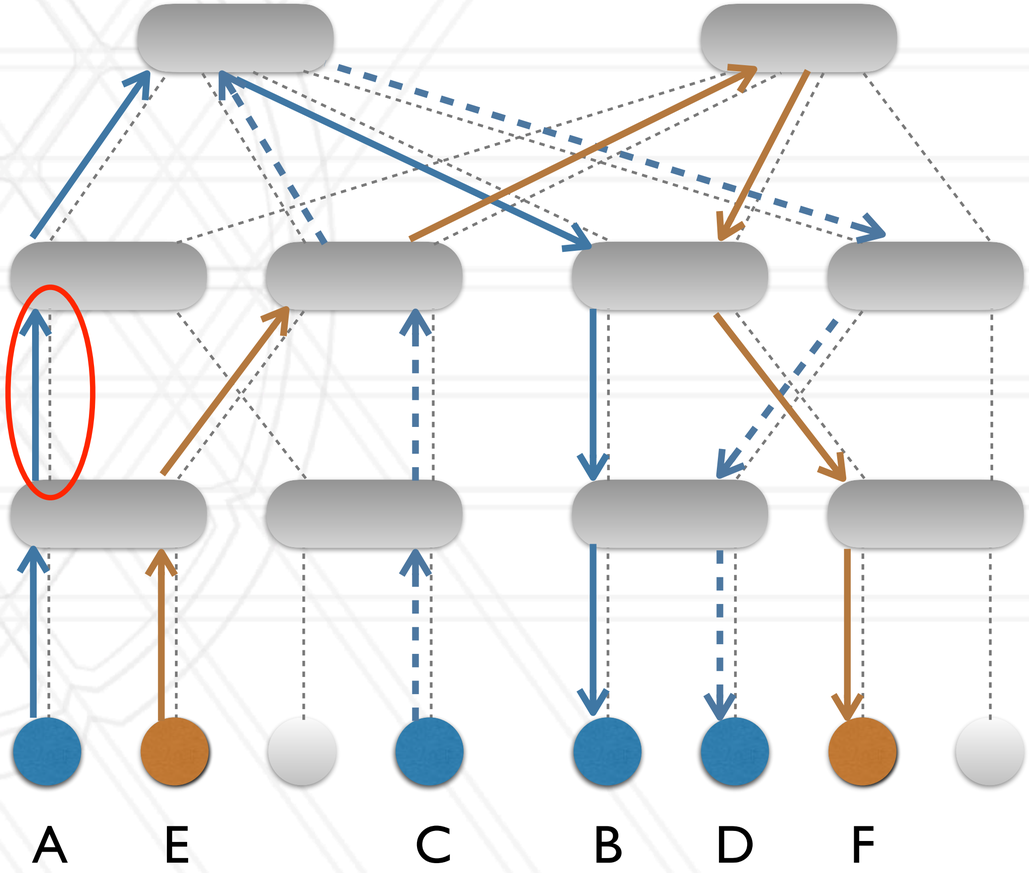


AFAR: adaptive flow aware routing

Solution: dynamically re-route traffic to alleviate hot-spots

Given: traffic for each pair of nodes in the system and the current routing

1. Calculate current load (network traffic) on all links in system
2. Find link with maximum load
3. If maximum > threshold, re-route one flow crossing that link to an under-utilized link
4. Repeat from 1. using new routing



UNIVERSITY OF
MARYLAND

Abhinav Bhatele

5218 Brendan Iribe Center (IRB) / College Park, MD 20742

phone: 301.405.4507 / e-mail: bhatele@cs.umd.edu