Microsoft Research

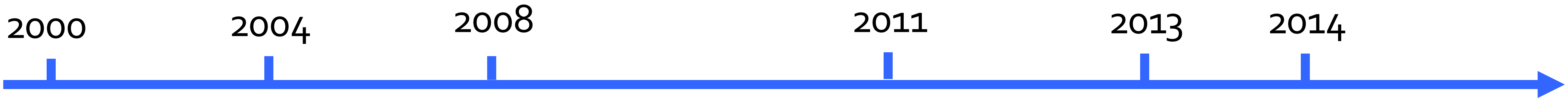
Microsoft

Rethinking the Network Stack for Rack-Scale Computers

Paolo Costa
paolo.costa@microsoft.com

joint work with Hitesh Ballani and Dushyanth Narayanan

Hardware Evolution in Data Centers



Trend towards customization

Increase work done per dollar (CapEx + OpEx)

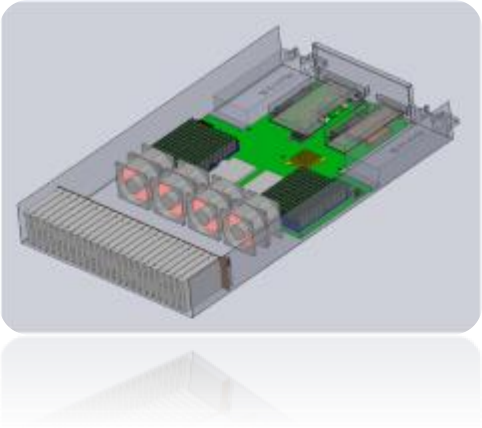
Hardware Evolution in Data Centers

2000 2004 2008 2011 2013 2014

Scale out vs. scale up

Many commodity servers rather than few expensive servers

Hardware Evolution in Data Centers

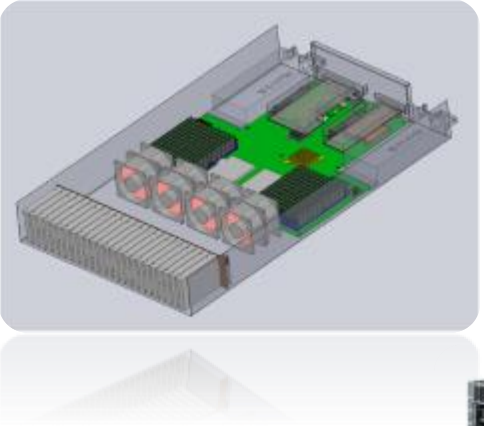


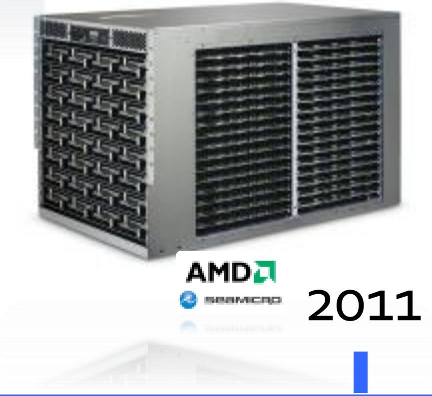
2000 2004 2008 2011 2013 2014

Custom layout

Remove unnecessary components (e.g., GPGPUs, USB ports)

Hardware Evolution in Data Centers



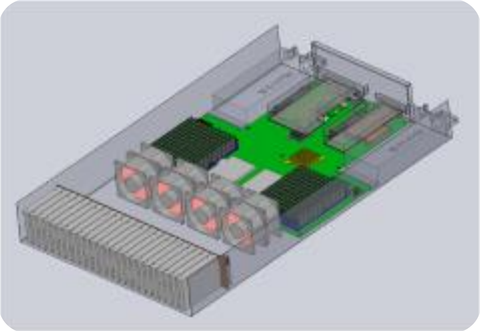


2000 2004 2008 2011 2013 2014

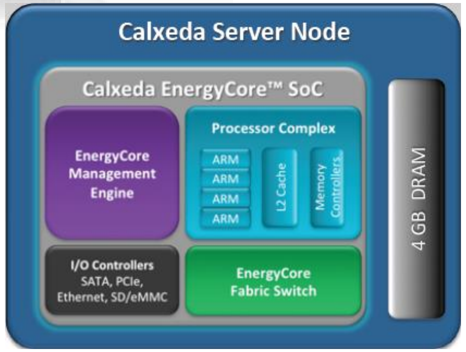
Integrated fabrics

Higher density and bandwidth with lower power consumption

Hardware Evolution in Data Centers



AMD

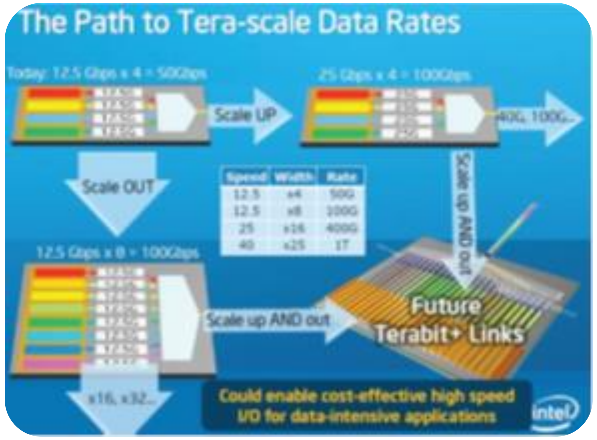


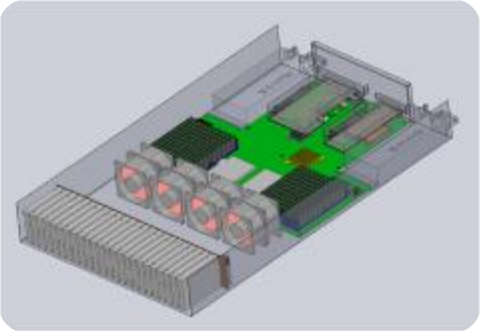
2000 2004 2008 2011 2013 2014

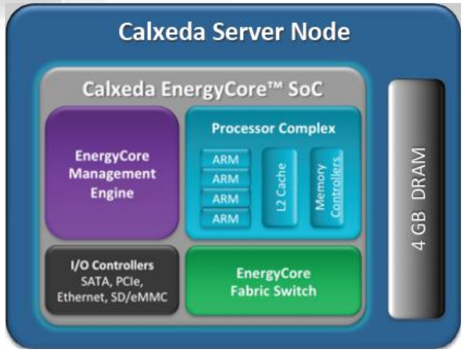
System-on-Chip (SoC)

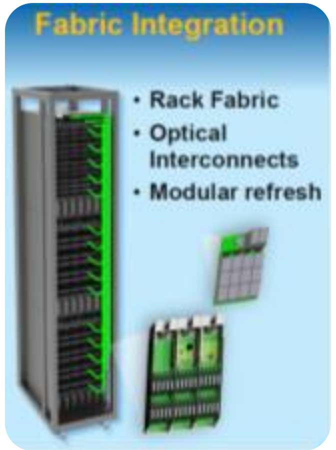
CPU, IO controllers, NIC/fabric switch on the same die

Hardware Evolution in Data Centers







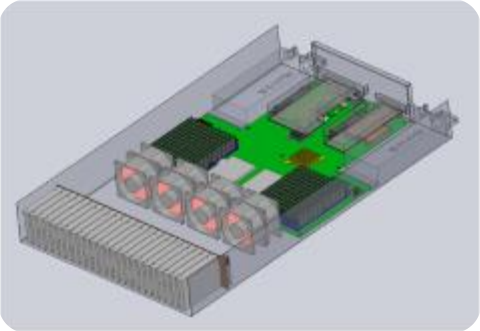


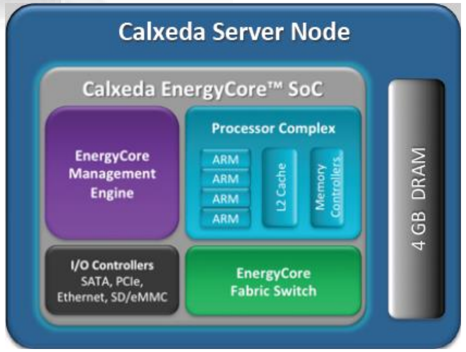
2000 2004 2008 2011 2013 2014

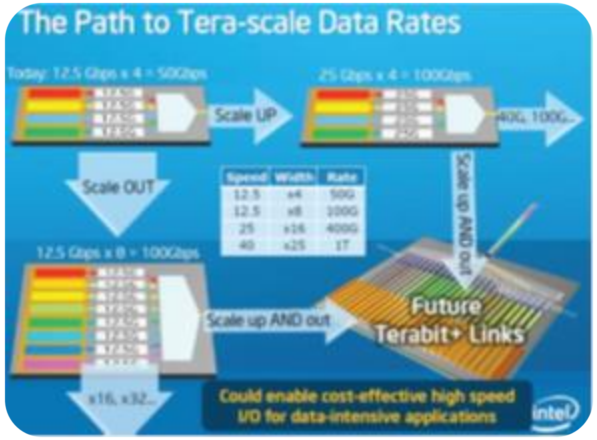
Silicon Photonics

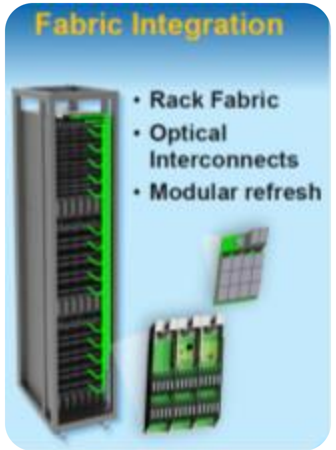
High-bandwidth / low-latency interconnect (resource disaggregation)

Hardware Evolution in Data Centers









2000 2004 2008 2011 2013 2014

1 rack unit (RU) 2 Ru 4-10 Ru Rack-scale

intra-server BW << inter-server BW

intra-server BW ≈ inter-server BW

Hardware Evolution in Data Centers

Rack-scale Computers

The rack is becoming the new unit of computing in data centers

Great opportunity to revisit early design assumptions and rethink the software stack and hw/sw co-design

2000

1 rack unit (RU) | 2 Ru | 4-10 Ru | Rack-scale

intra-server BW << inter-server BW

intra-server BW ≈ inter-server BW

A First Step: Rethinking the Network Stack

A First Step: Rethinking the Network Stack

Is it just a faster network
or is it fundamentally different?

A First Step: Rethinking the Network Stack

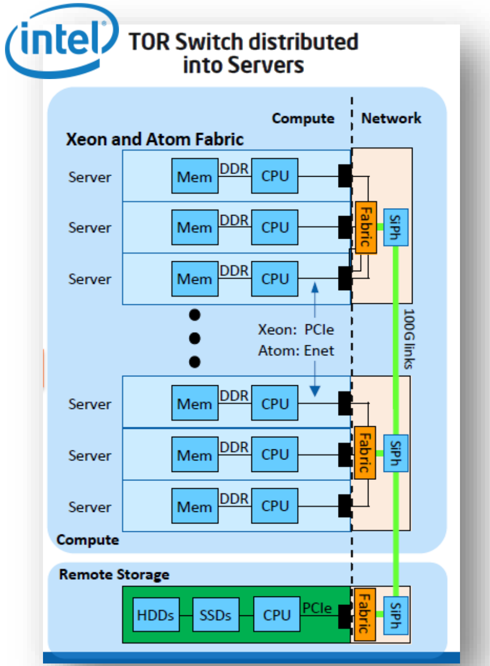
Many designs are possible but some trends are emerging:

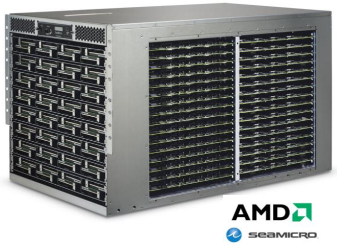
A First Step: Rethinking the Network Stack

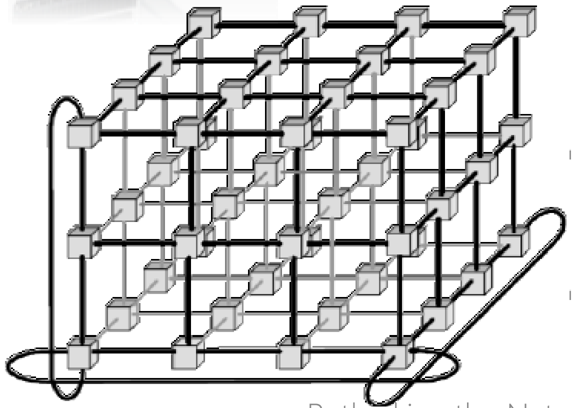
Many designs are possible but some trends are emerging:

1. Distributed Switching Fabric

✓ *High path diversity*





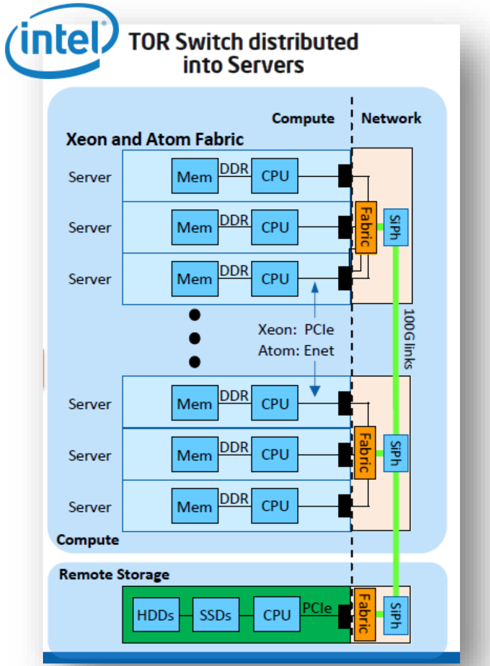


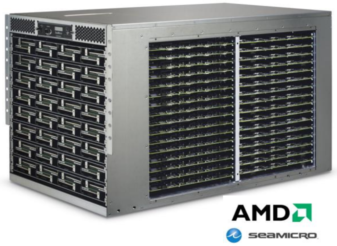
A First Step: Rethinking the Network Stack

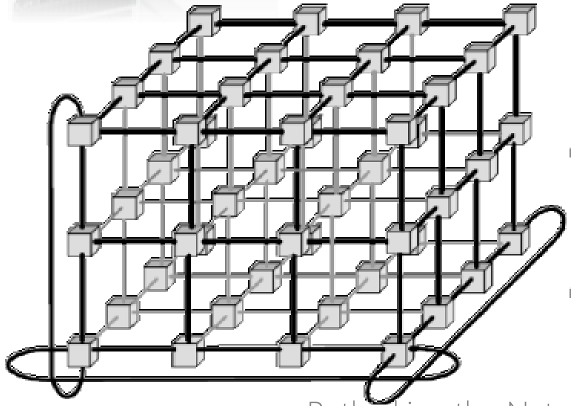
Many designs are possible but some trends are emerging:

1. Distributed Switching Fabric

✓ *High path diversity*

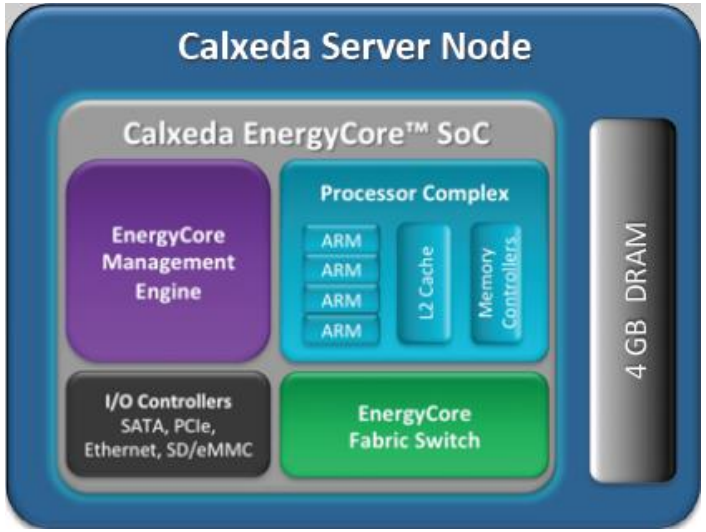






2. Tight CPU / network integration

✓ *Direct control on network resources*



A First Step: Rethinking the Network Stack

Many designs are possible but some trends are emerging:

Research Question:

How can we take advantage of these features
to design a new network stack optimized for rack-scale computers?

Focus of this work

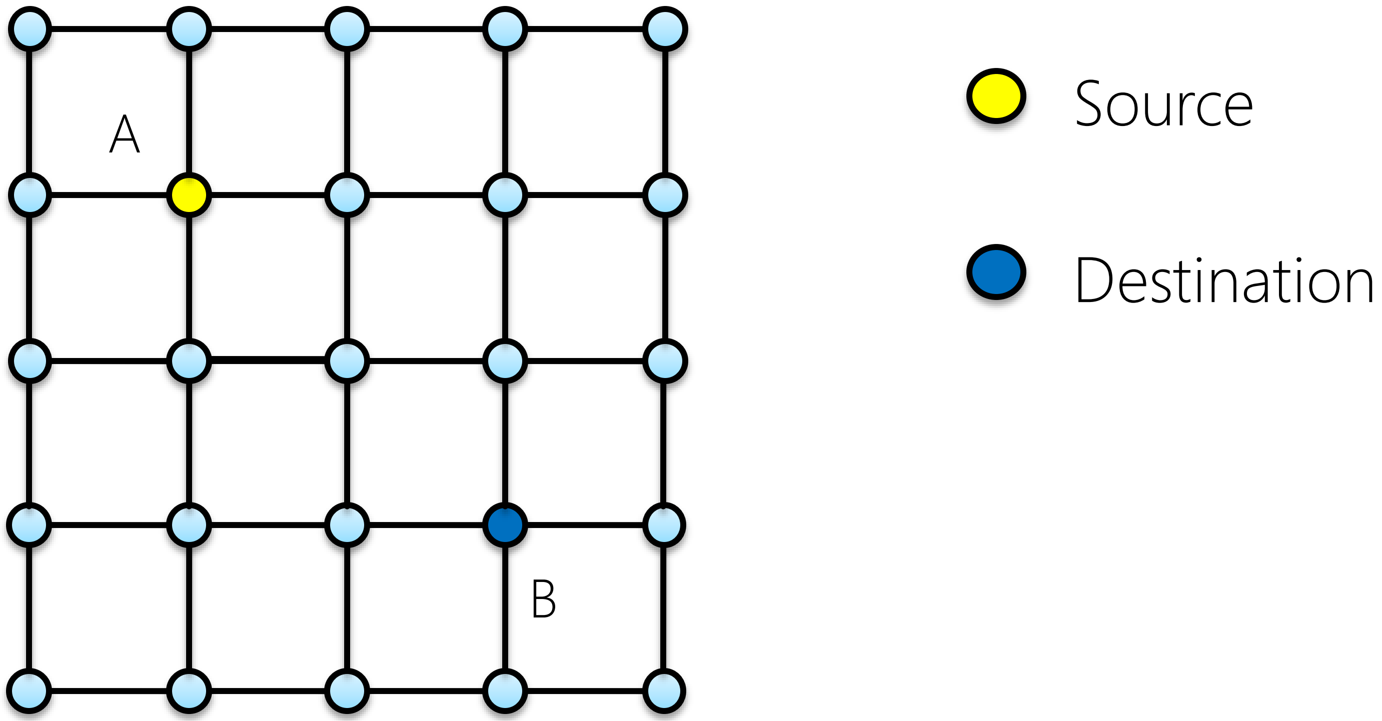
How to route packets (routing)?
At what rate should the packets be sent (rate control)?

RaSC-Net Design

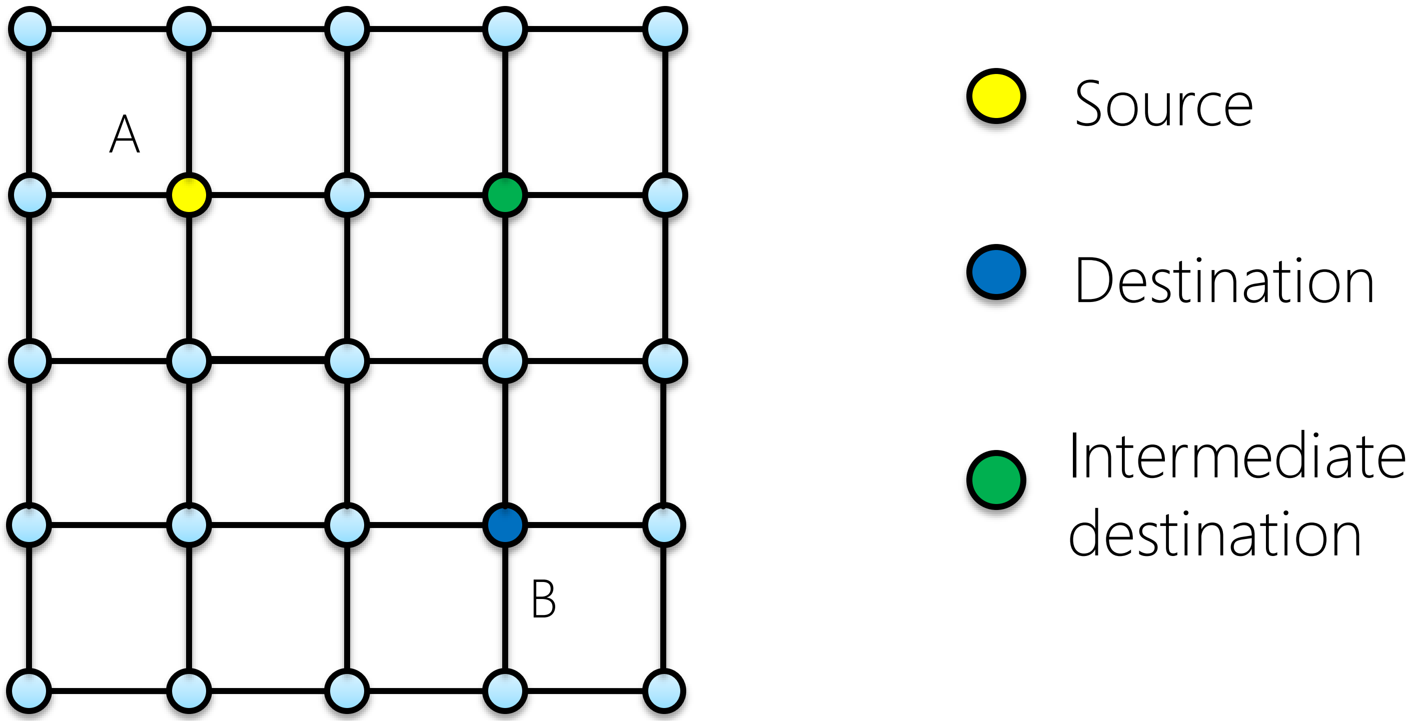
Goals

1. Leverage path diversity and ensure load balance
2. Support arbitrary and dynamic traffic matrixes
3. Achieve low queuing
4. Support custom rate allocation policies
   – priority, deadline-based, tenant-based, resource-based, …

RaSC-Net Design: Routing



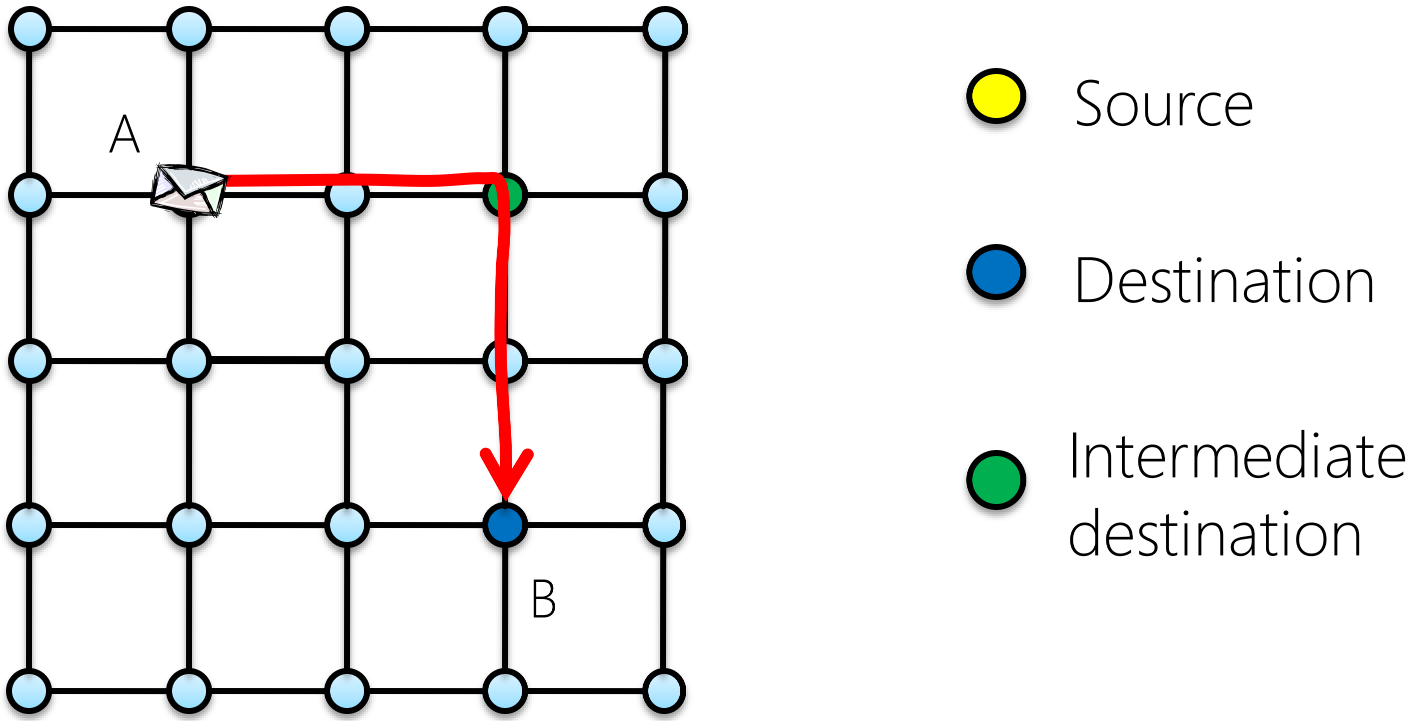
RaSC-Net Design: Routing



Valiant Load Balancing (VLB)

Randomly selects an intermediate destination per each packet

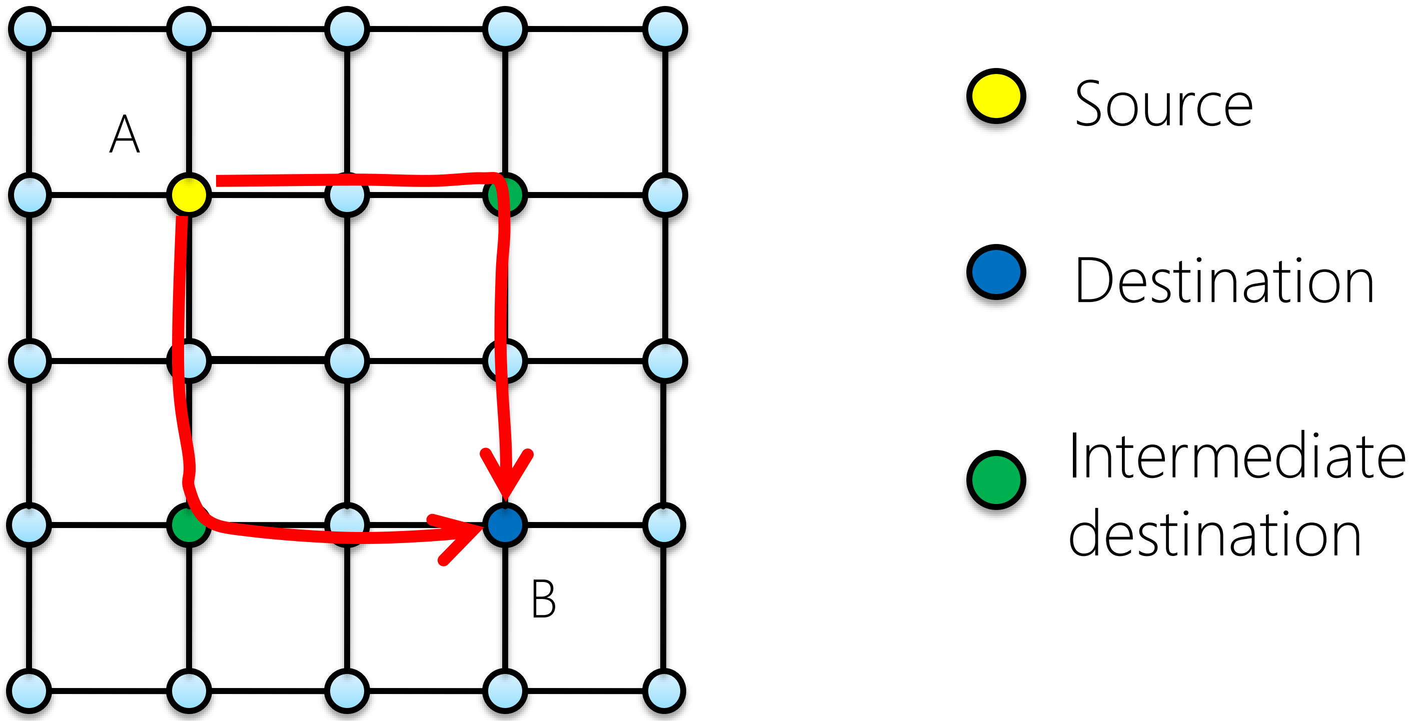
RaSC-Net Design: Routing



Valiant Load Balancing (VLB)

Randomly selects an intermediate destination per each packet

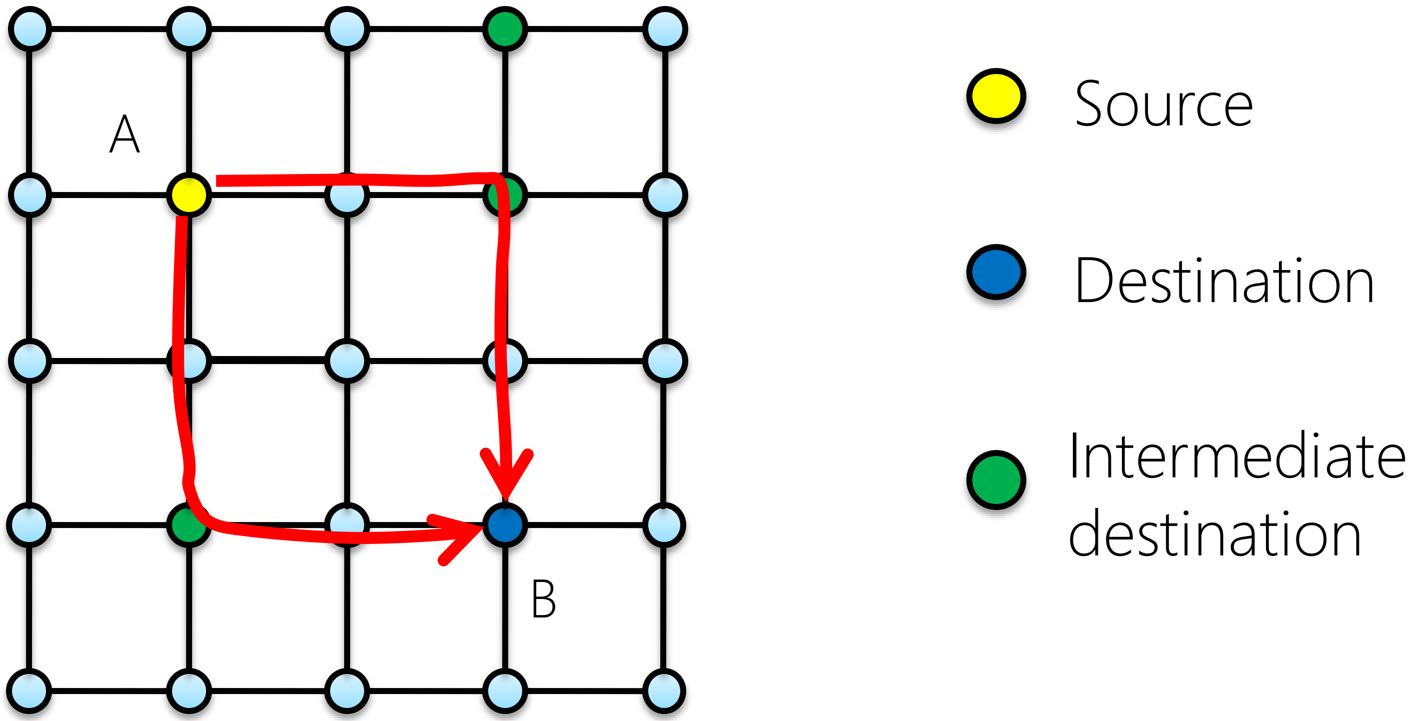
RaSC-Net Design: Routing



Valiant Load Balancing (VLB)

Randomly selects an intermediate destination per each packet

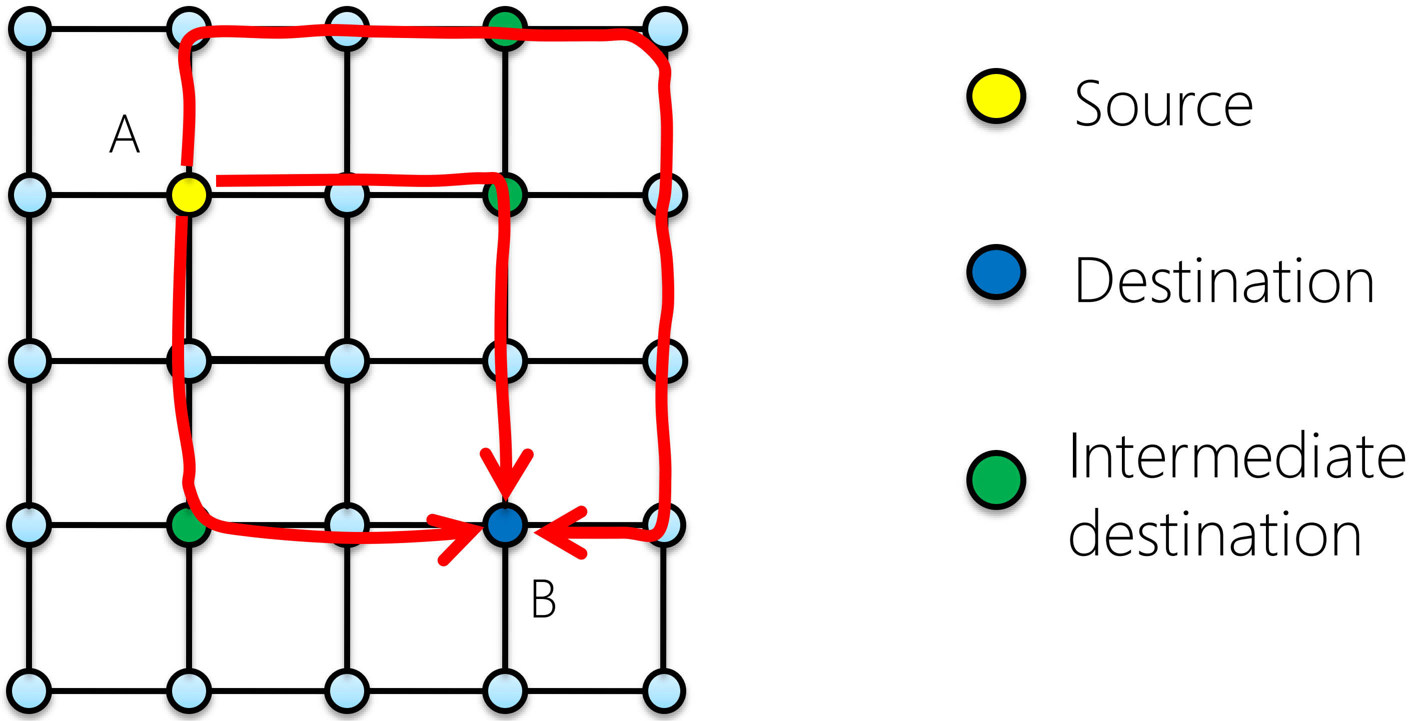
RaSC-Net Design: Routing



Non-minimal routing

Intermediate destinations do not need to lie on the shortest path

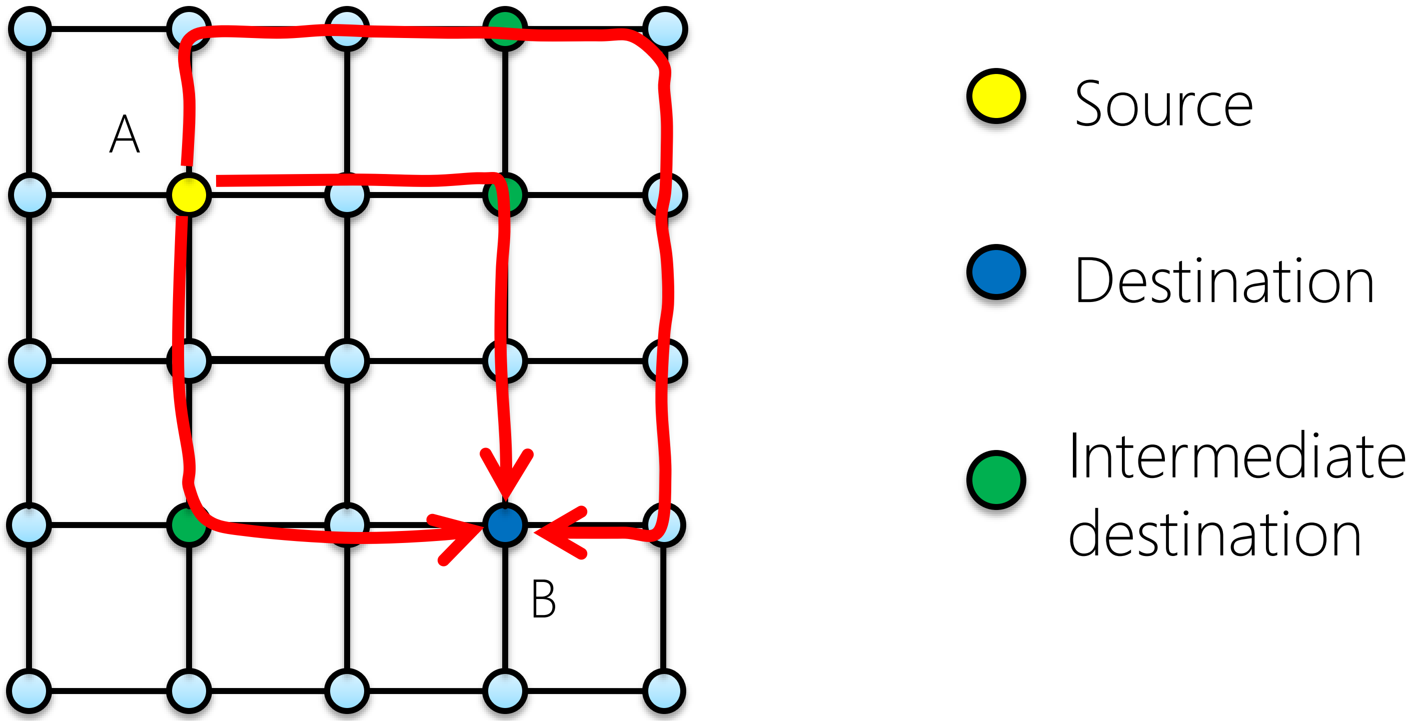
RaSC-Net Design: Routing



Non-minimal routing

Intermediate destinations do not need to lie on the shortest path

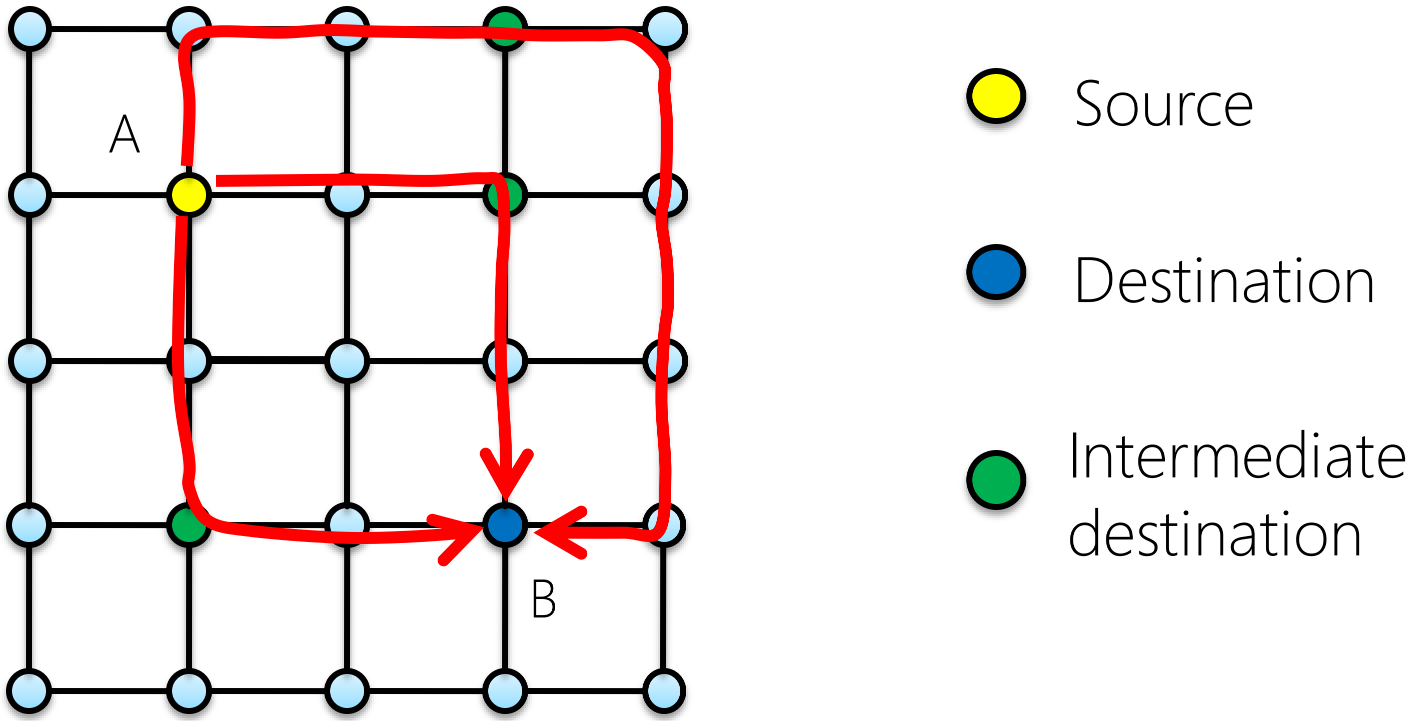
RaSC-Net Design: Routing



Why does it work?

VLB aims to transform any traffic matrix into a uniform one
Link load balancing (*Goal #1*) and independent of traffic matrix (*Goal #2*)

RaSC-Net Design: Routing

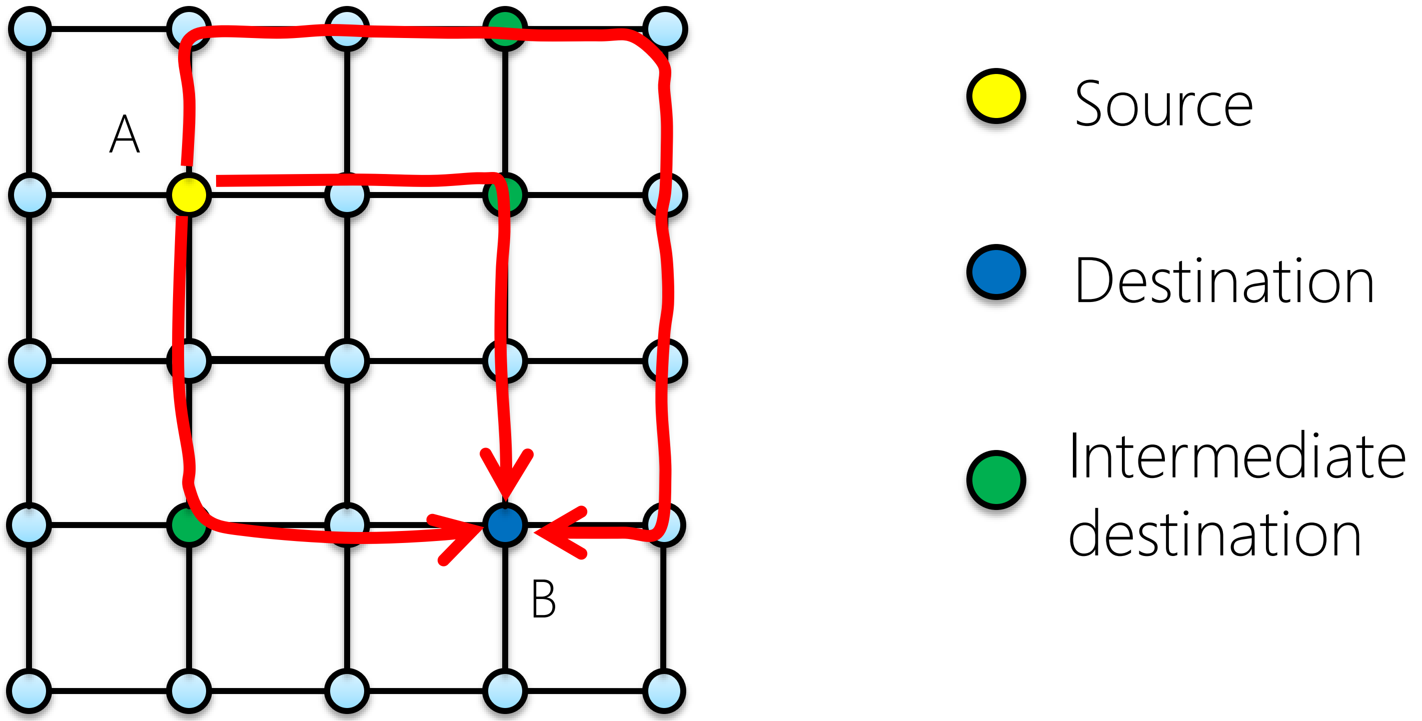


VLB Drawback #1: Path Stretch

Average path length and link utilization increases

Solution: Locality-aware variants of VLB can be used (weighted choice)

RaSC-Net Design: Routing

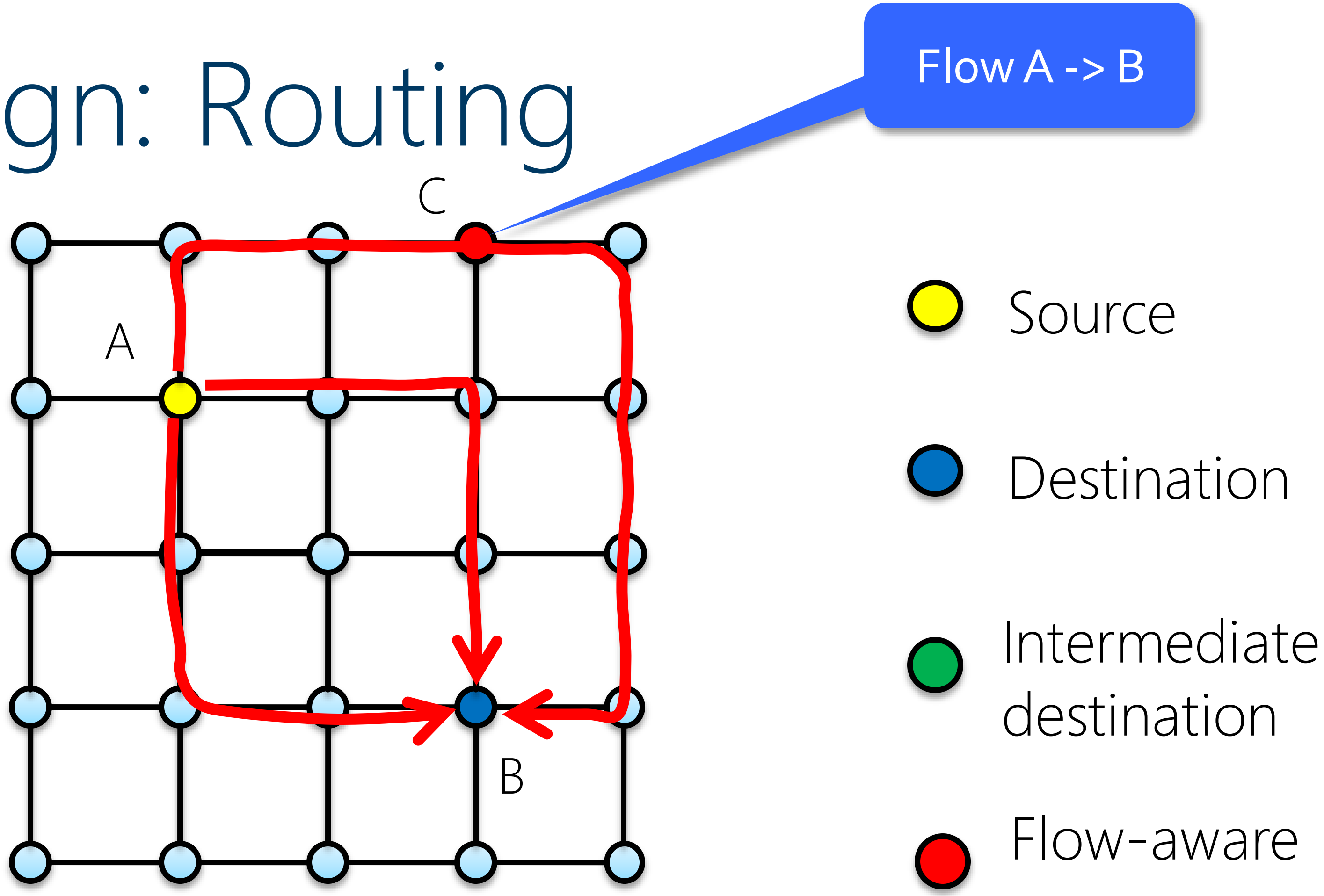


VLB Drawback #2: Out of order packet arrival

Packets need be reordered at the destination (jitter)

Solution: low diameter and minimize queuing delay

RaSC-Net Design: Routing

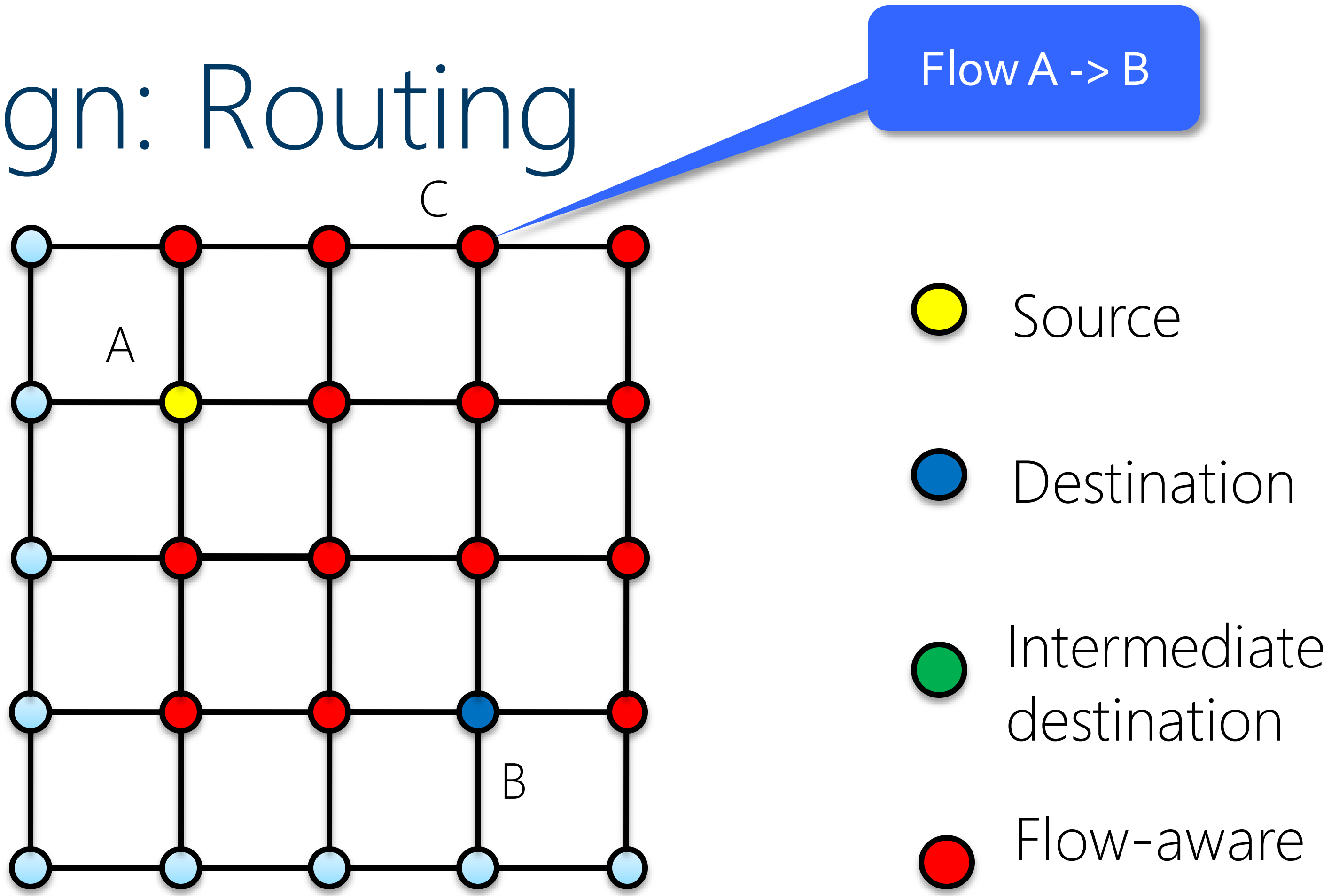


Key observation: VLB enables global visibility

Nodes inspect packet headers of forwarded packets

They can locally reconstruct the traffic matrix

RaSC-Net Design: Routing

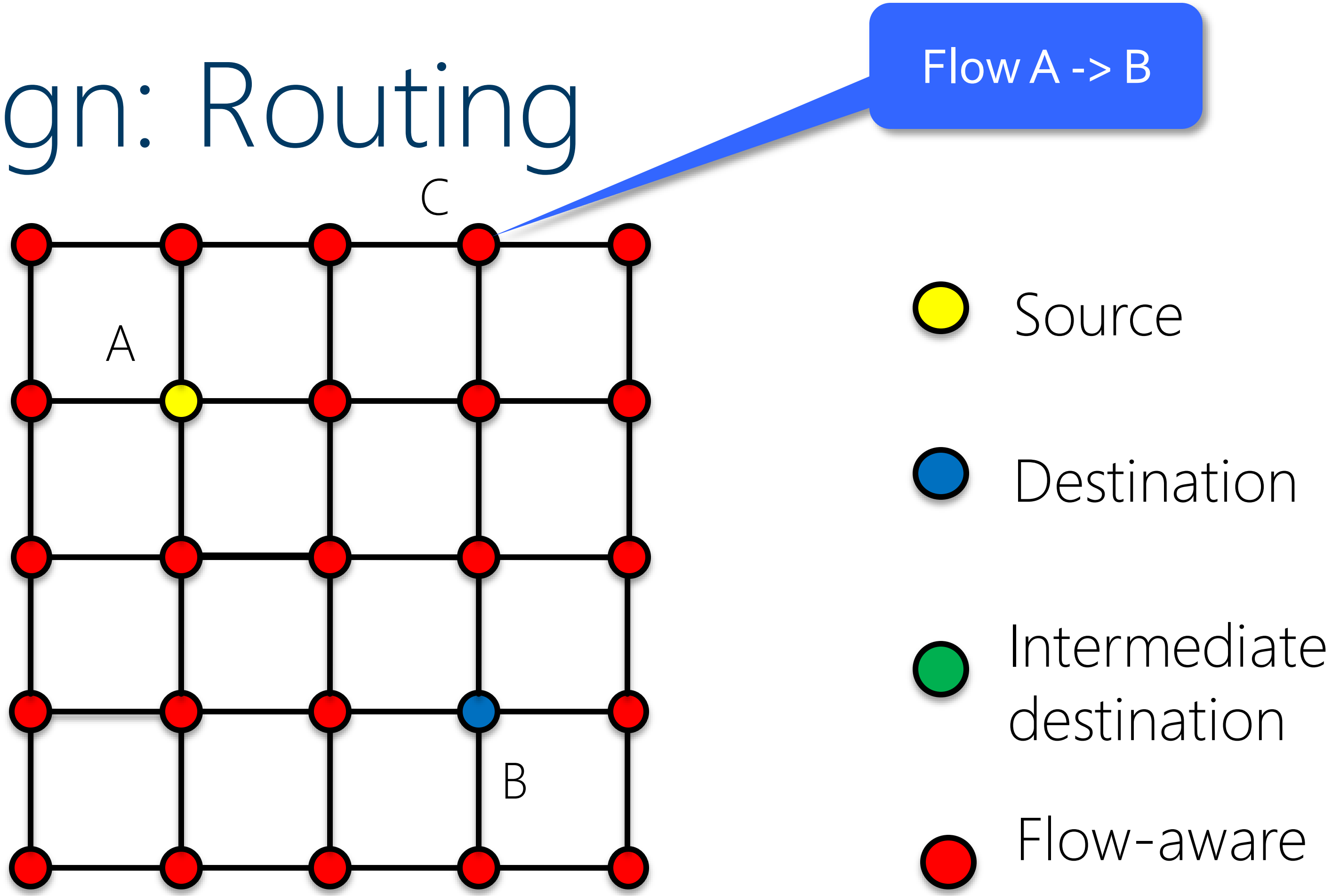


Key observation: VLB enables global visibility

Nodes inspect packet headers of forwarded packets

They can locally reconstruct the traffic matrix

RaSC-Net Design: Routing

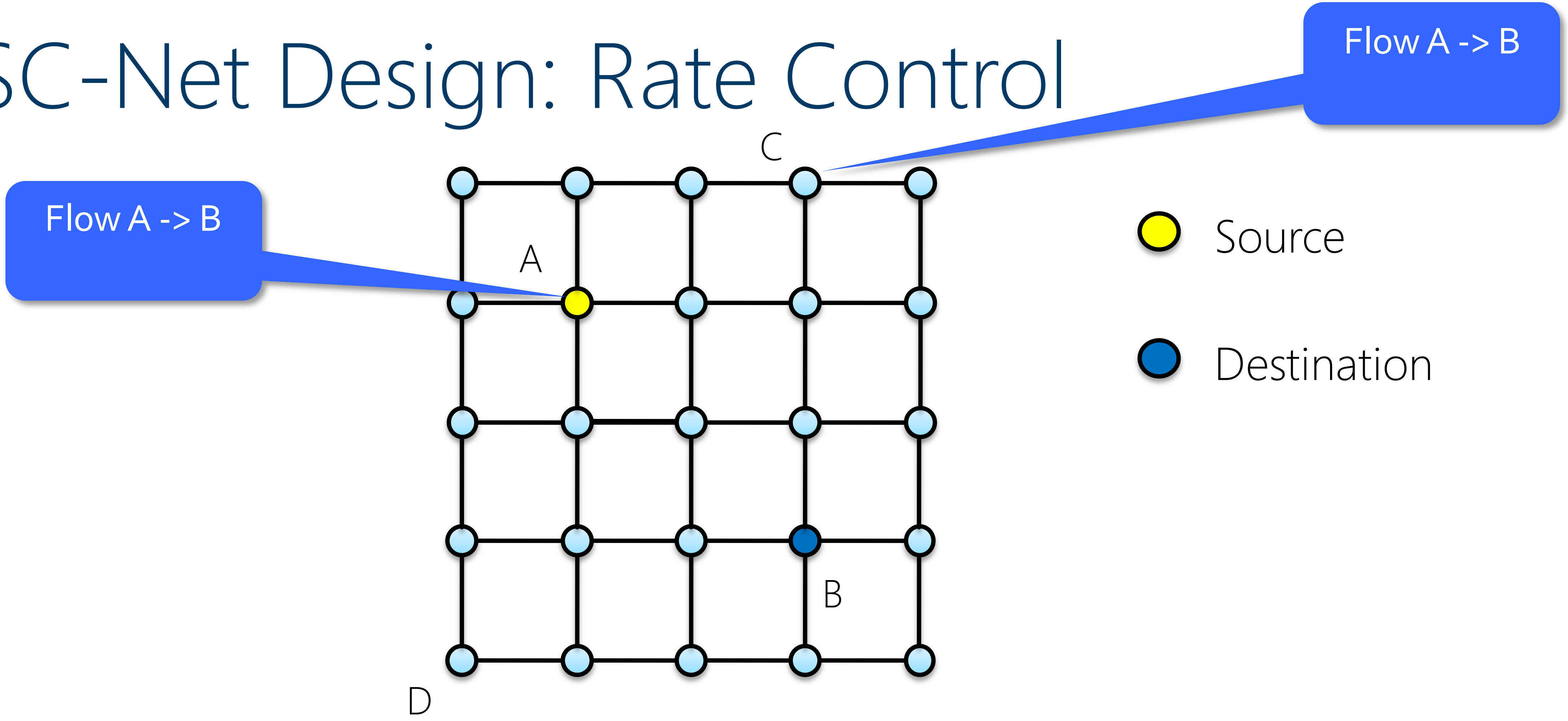


Key observation: VLB enables global visibility

Nodes inspect packet headers of forwarded packets

They can locally reconstruct the traffic matrix

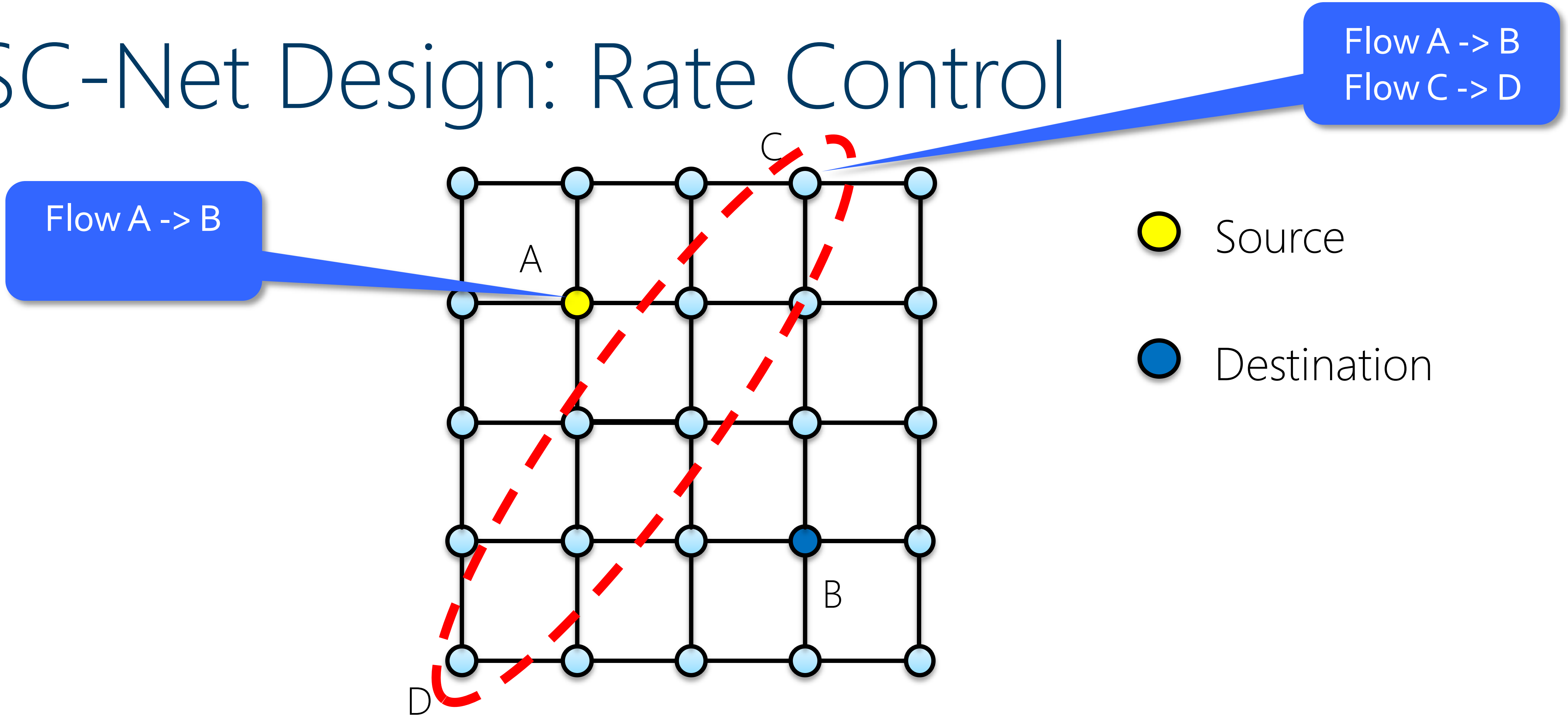
RaSC-Net Design: Rate Control



Rate Allocation

Given the knowledge of the topology and the traffic matrix, each node can locally derive the correct rate for its flows

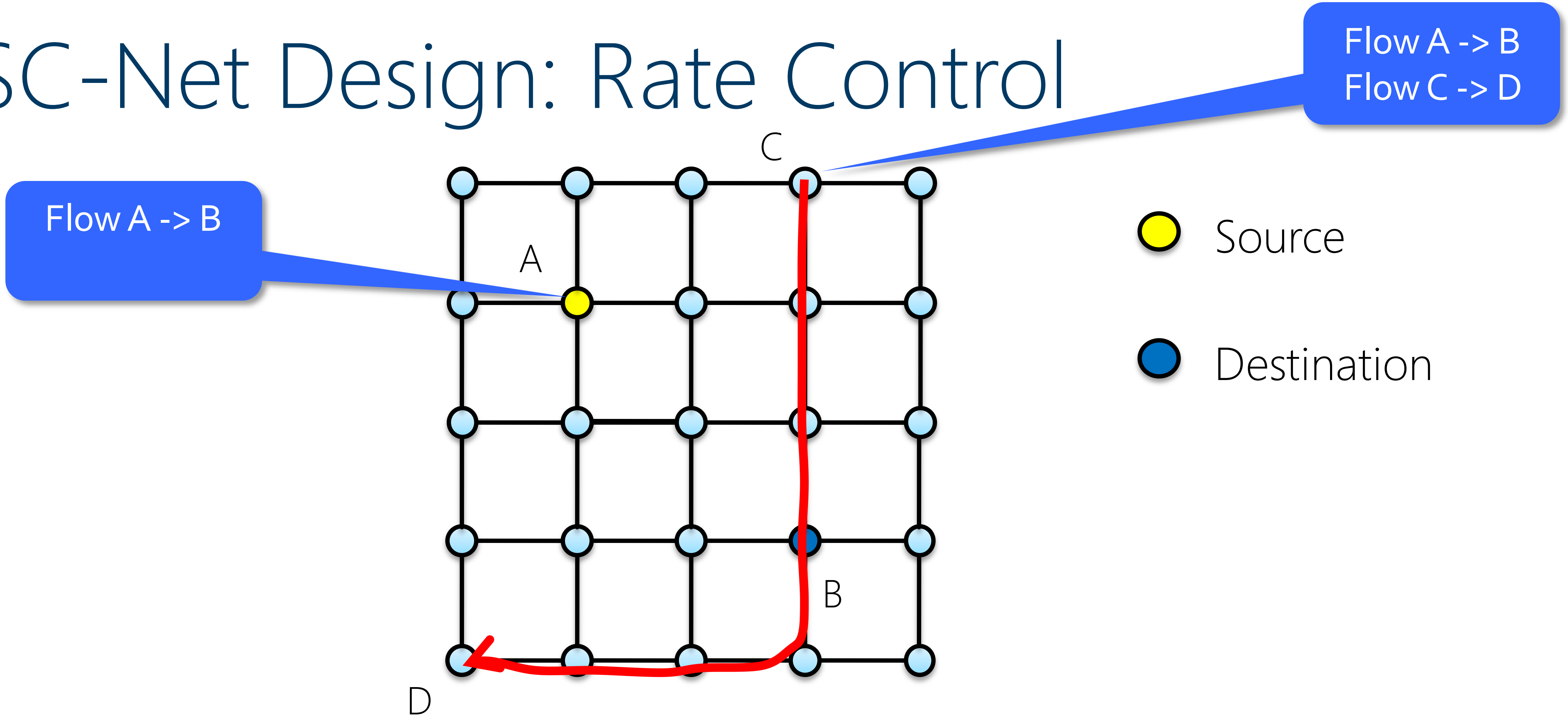
RaSC-Net Design: Rate Control



Rate Allocation

Given the knowledge of the topology and the traffic matrix, each node can locally derive the correct rate for its flows

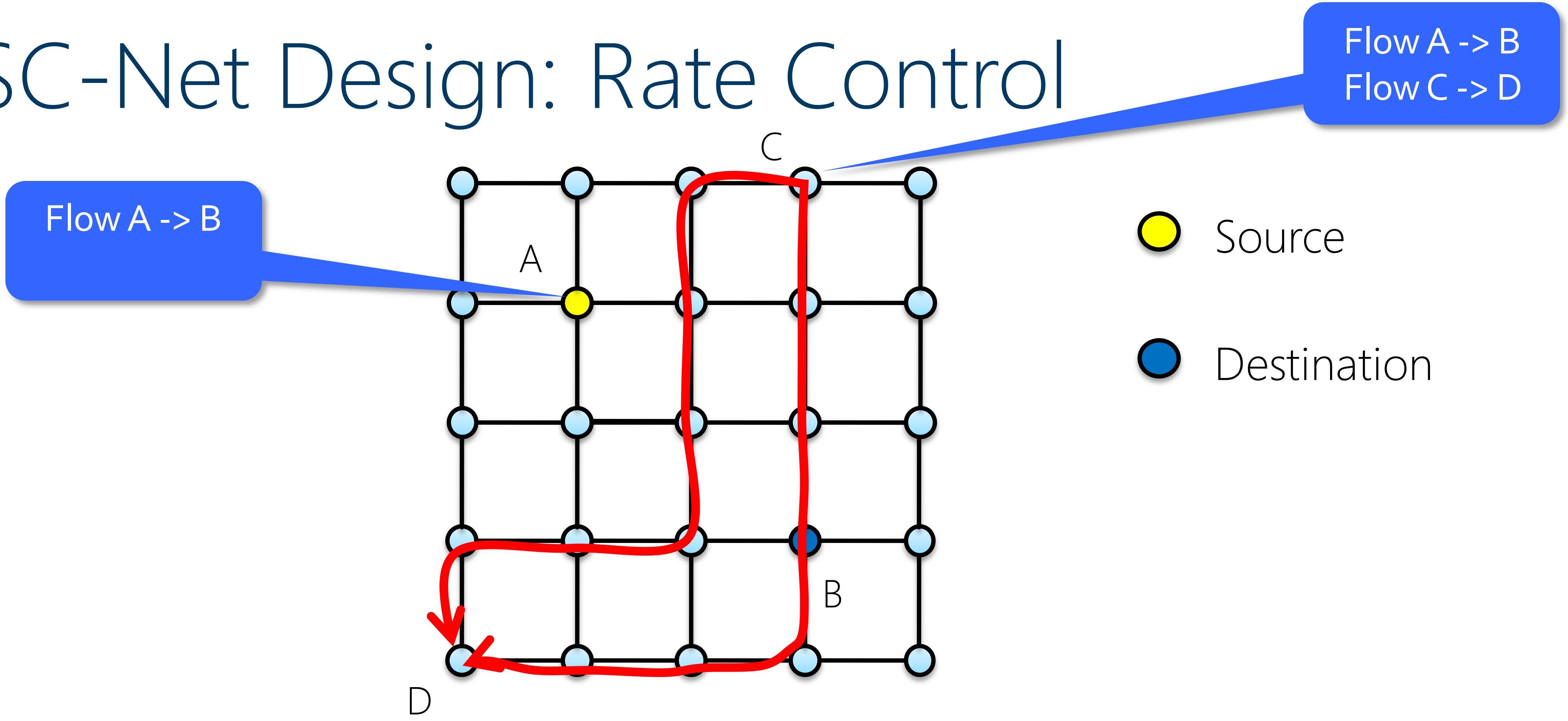
RaSC-Net Design: Rate Control



Rate Allocation

Given the knowledge of the topology and the traffic matrix, each node can locally derive the correct rate for its flows

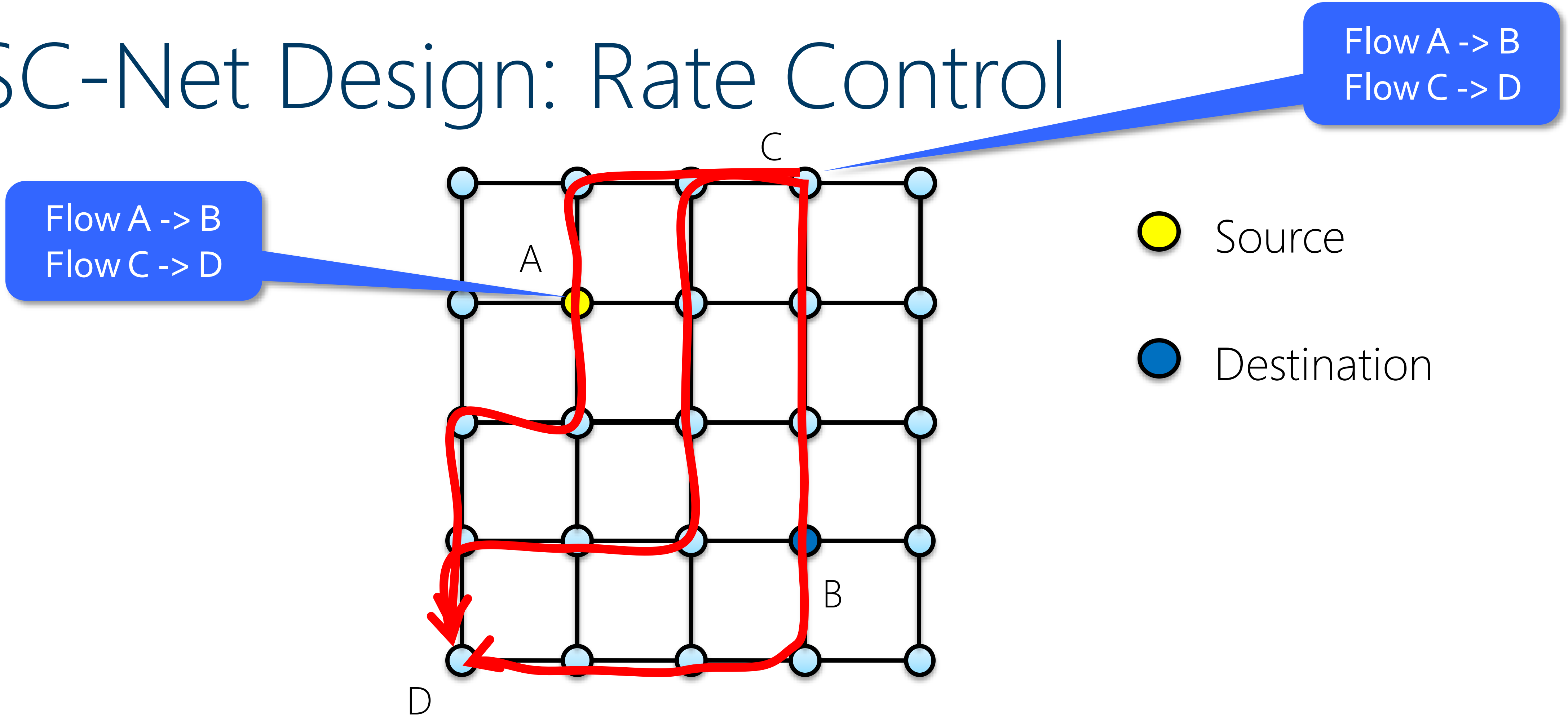
RaSC-Net Design: Rate Control



Rate Allocation

Given the knowledge of the topology and the traffic matrix, each node can locally derive the correct rate for its flows

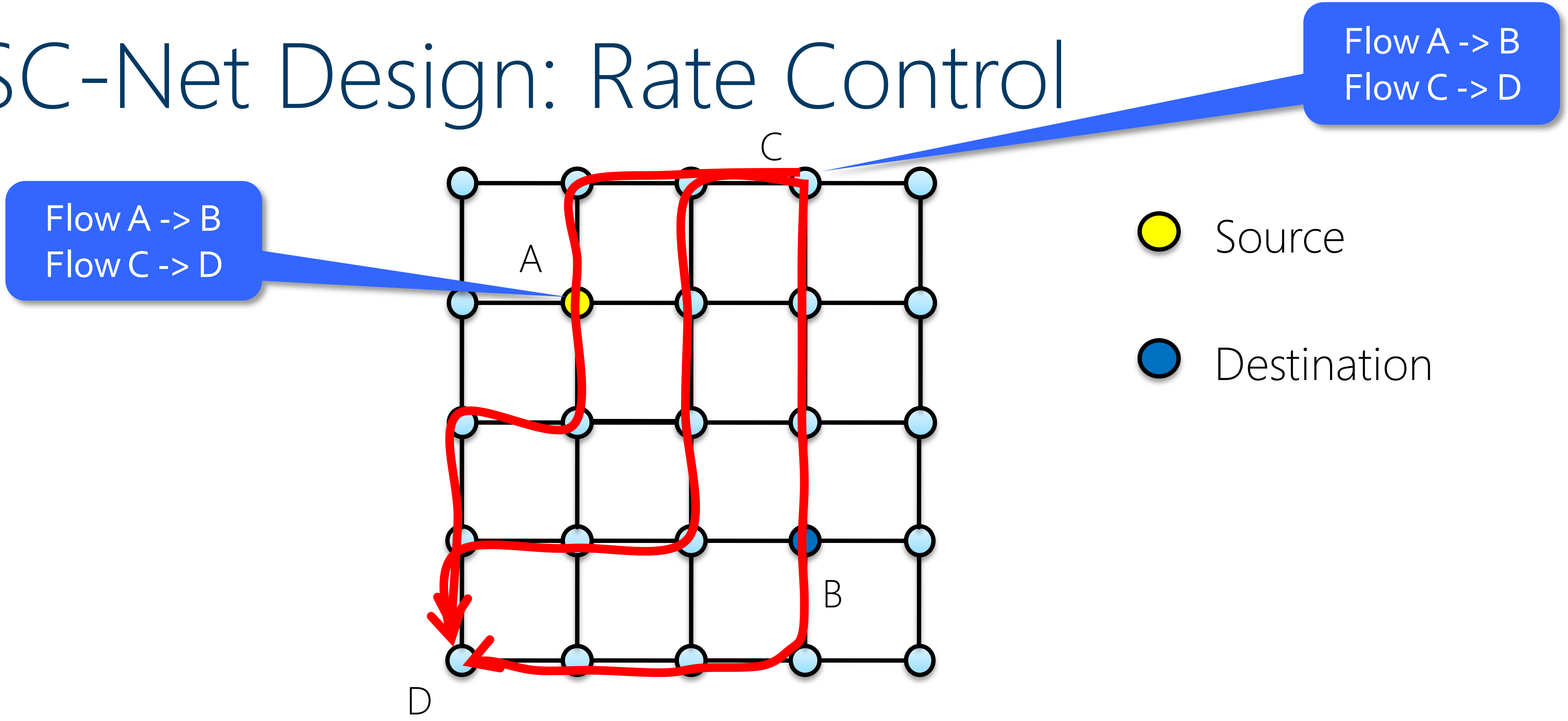
RaSC-Net Design: Rate Control



Rate Adaptation

When a packet from a new flow is received, nodes can update their rates accordingly

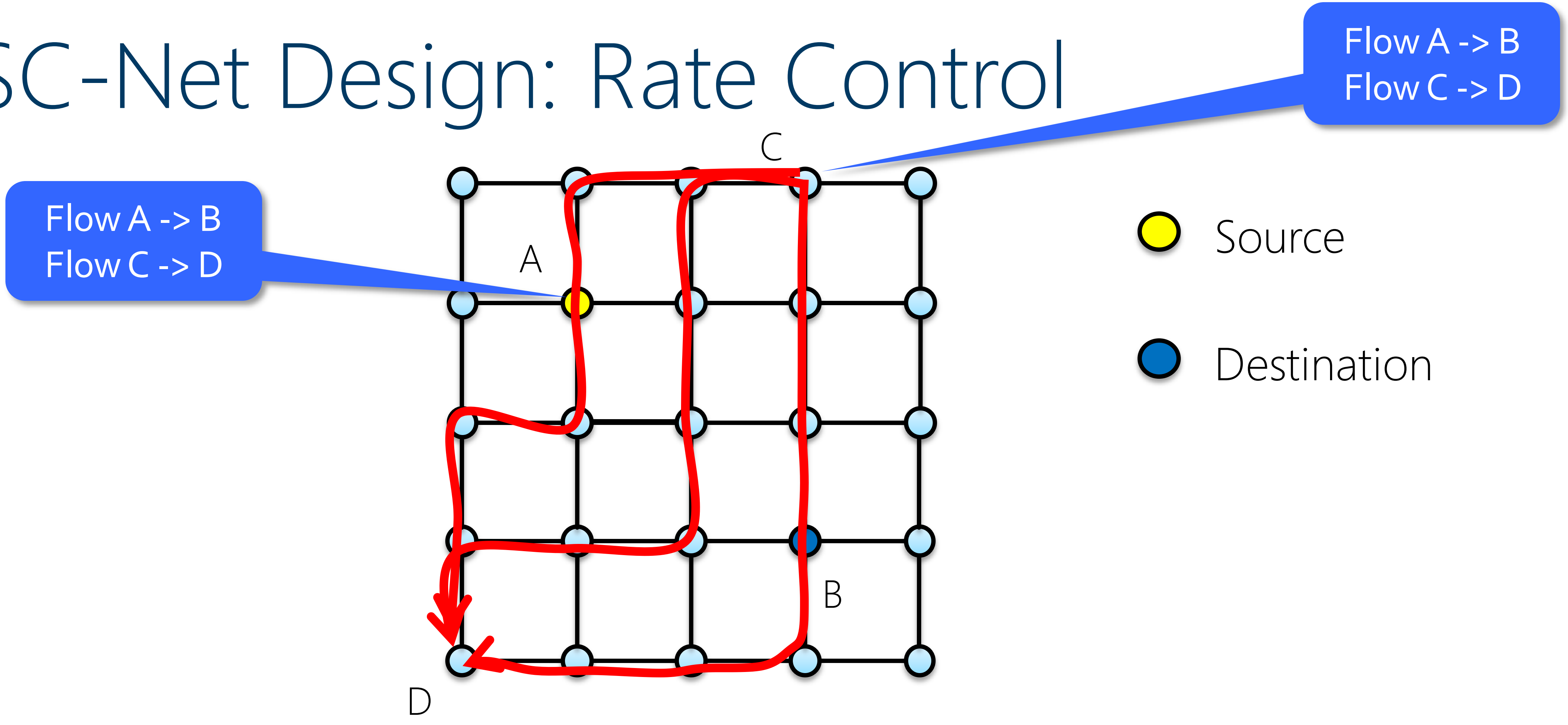
RaSC-Net Design: Rate Control



Key result

A traditionally distributed problem (rate control) becomes a local operation

RaSC-Net Design: Rate Control

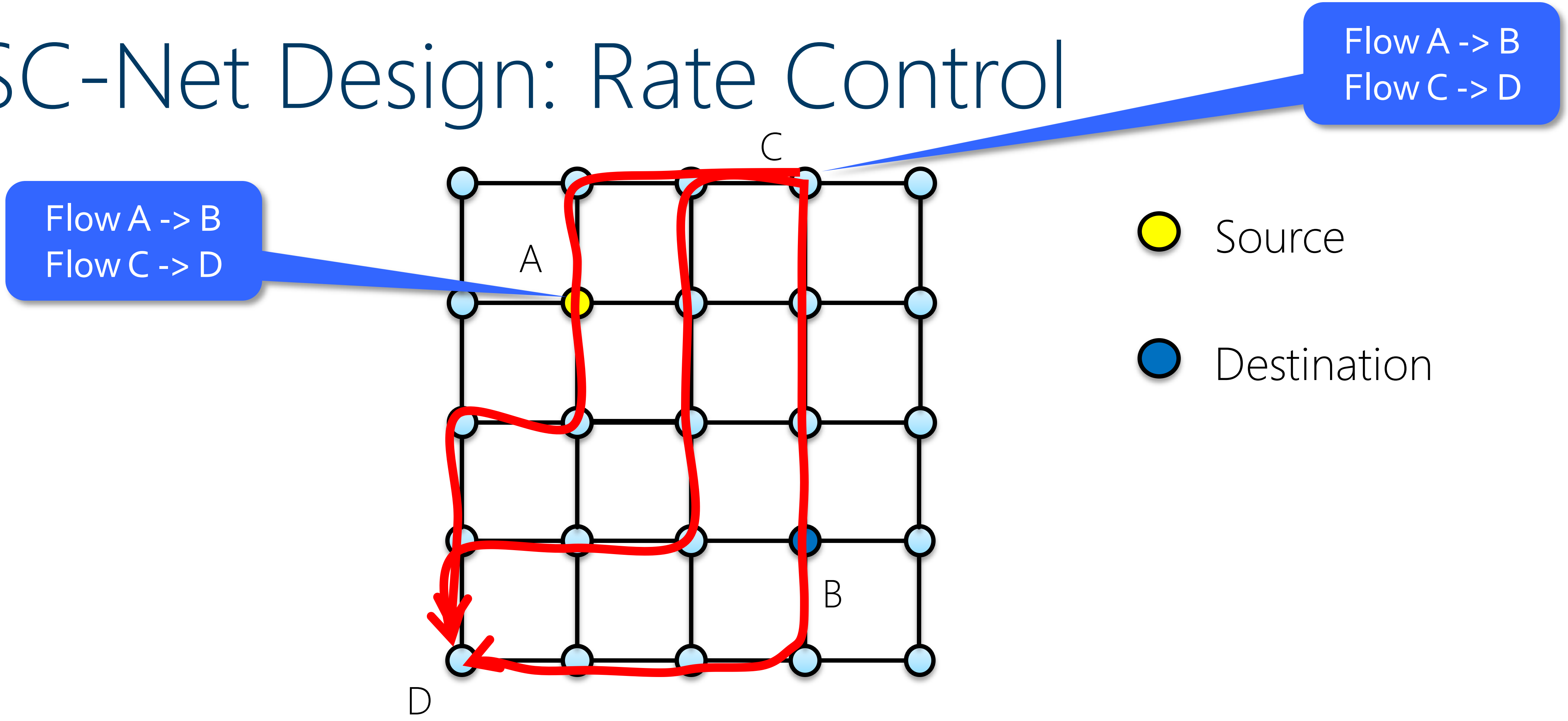


Low queuing (Goal #3)

No need to probe the network a la TCP

Rates are chosen so as to minimize queuing

RaSC-Net Design: Rate Control



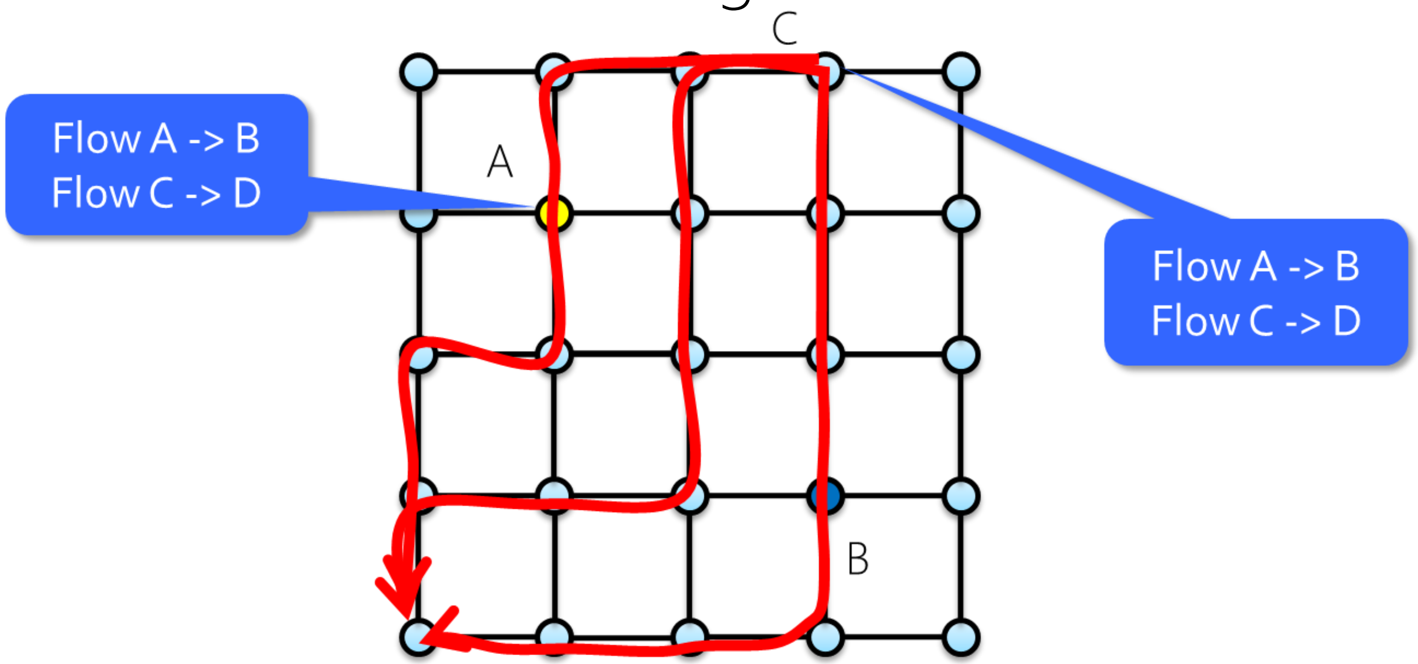
Beyond max-min fairness (Goal #4)

Since rate allocation is now a local computation, it is easy to encode different allocation policies

Something (NOT) to worry about…

Something (NOT) to worry about…

1. Short flows
   - Our protocol needs a few packets to converge…
   - We can reserve some spare capacity to absorb short (and newly created) flows
   - Small packets and deterministic routing can also reduce convergence time

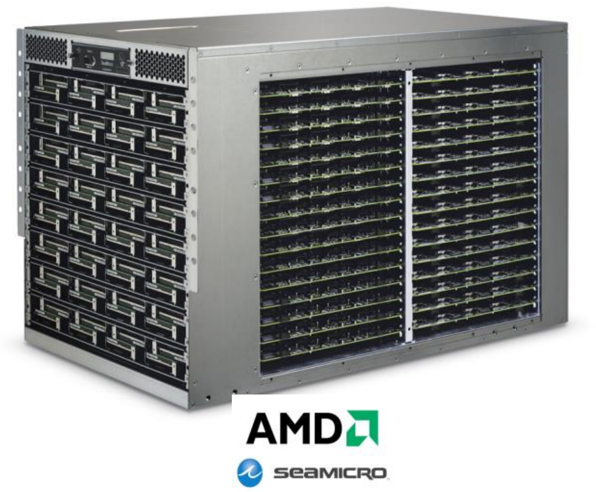


Something (NOT) to worry about…

1. Short flows
   - Our protocol needs a few packets to converge…
   - We can reserve some spare capacity to absorb short (and newly created) flows
   - Small packets and deterministic routing can also reduce convergence time

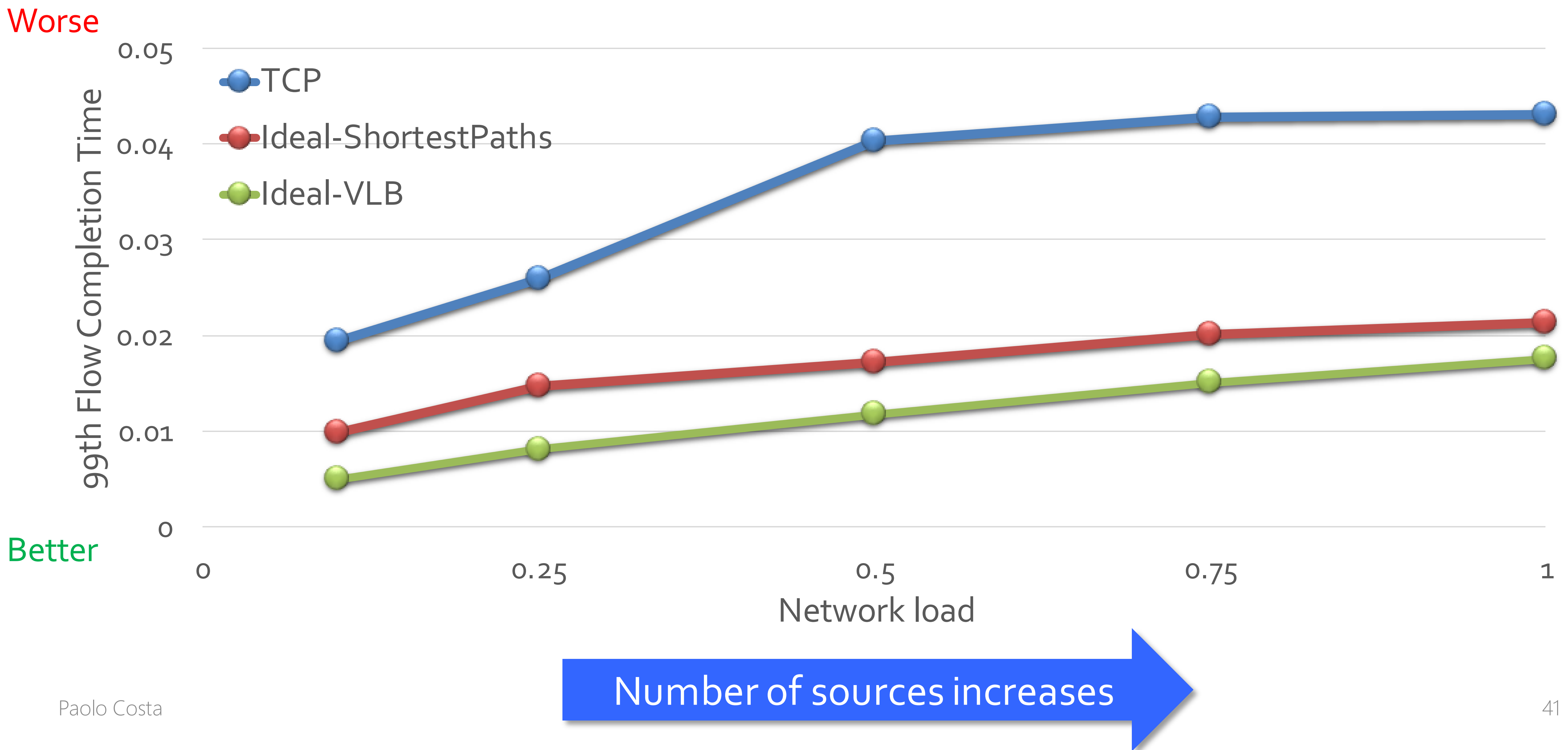
2. Computation overhead
   - In general, max/min computation is expensive …
   - …but with VLB a flow uses almost all links
     - Hence, the most bottleneck link is actually the bottleneck for almost all active flows
     - Approximate results can also be used (utilization vs. computation trade-off)

RaSC-Net: Preliminary Results

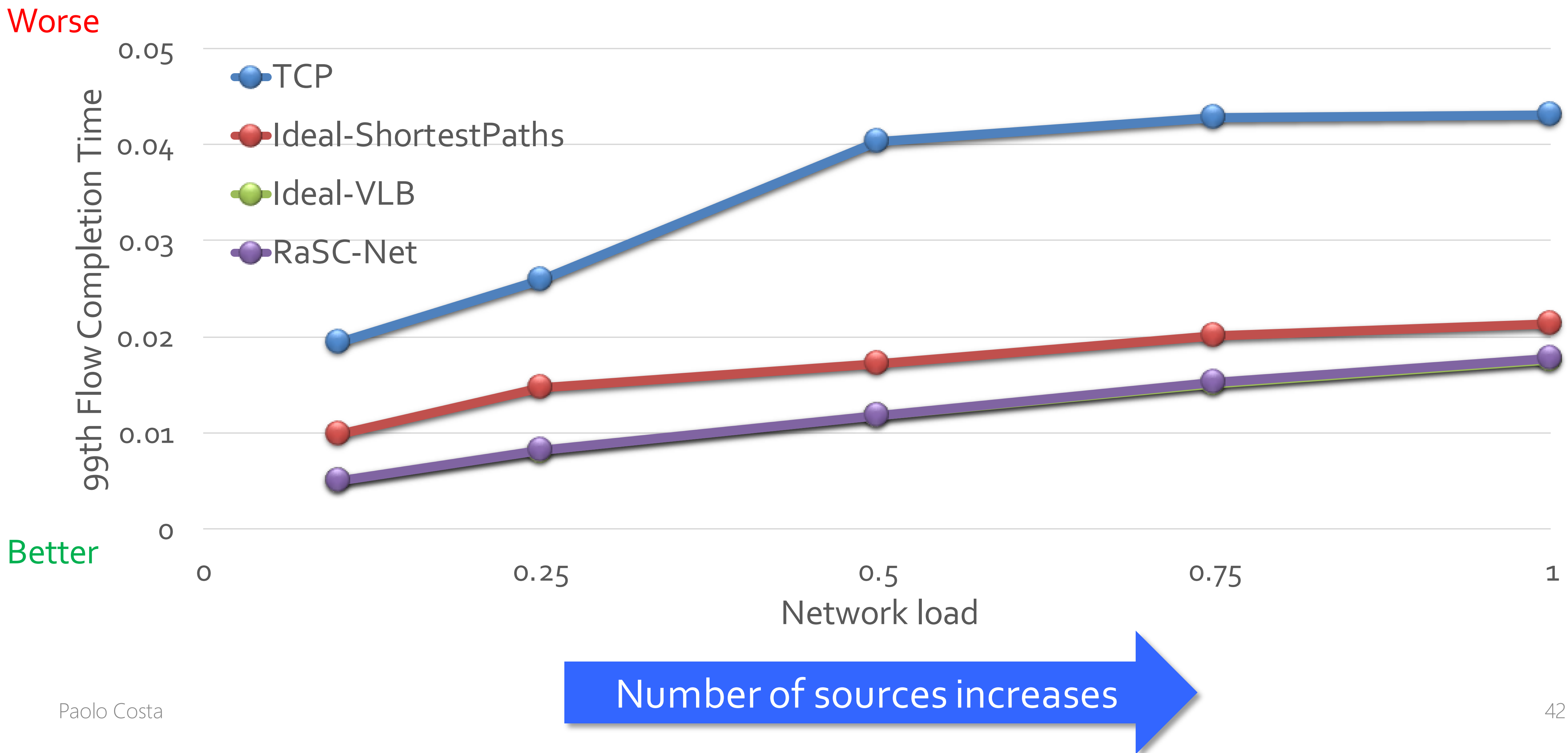


- Simulation setup
  - 512-node 3D torus (8 x 8 x 8)
  - 6 10Gbps links / server
  - Permutation matrix with varying number of sources (load)
  - 10-MB flows all starting at the same time
- Baselines
  - *TCP*: ECMP routing protocol (single path per flow) + TCP
  - Idealized (unlimited per-flow queues)
    - *Ideal-ShortestPaths*: Packet spraying (minimal routing)
    - *Ideal-VLB*: Valliant Load Balancing

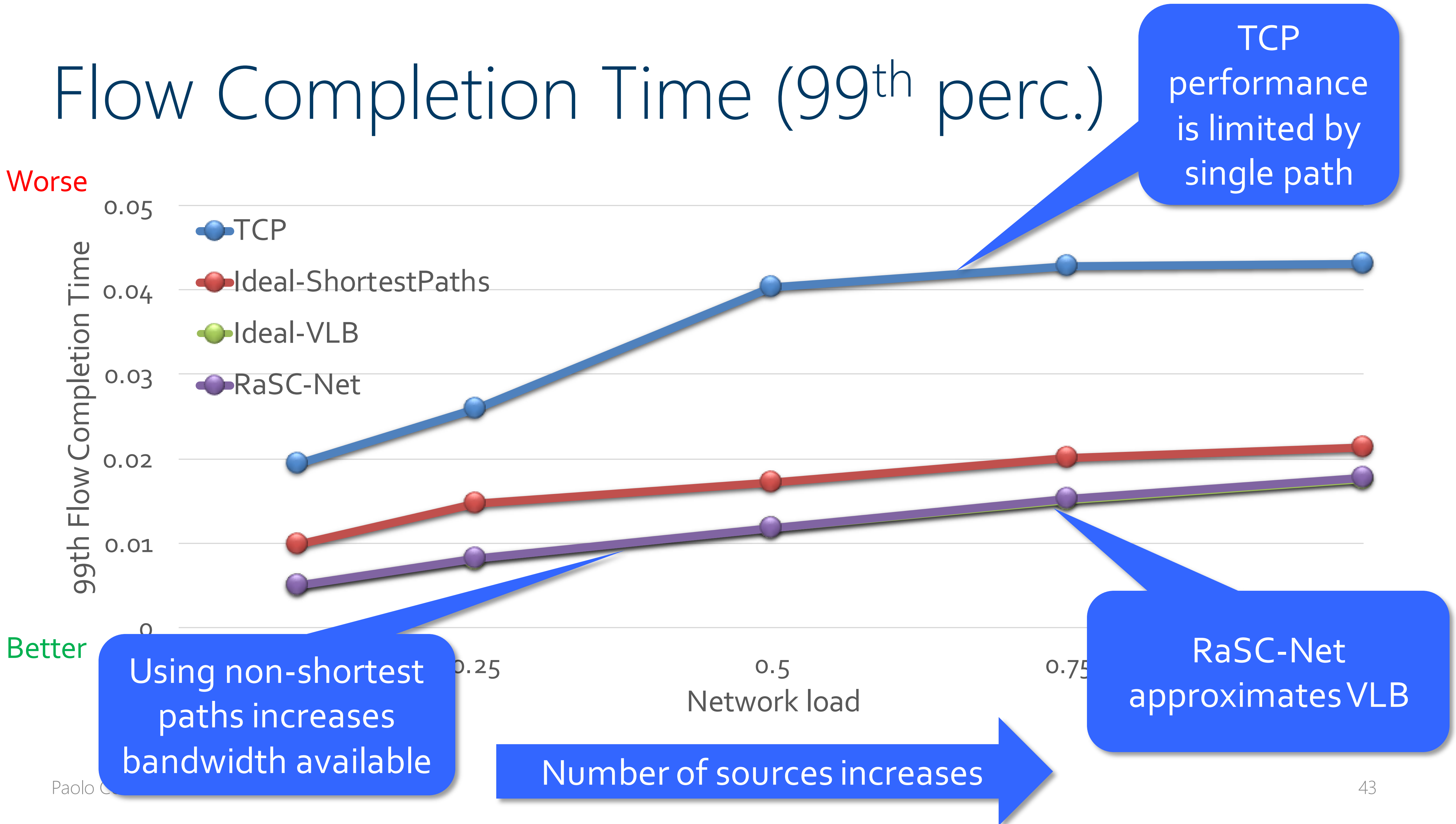
Flow Completion Time (99th perc.)



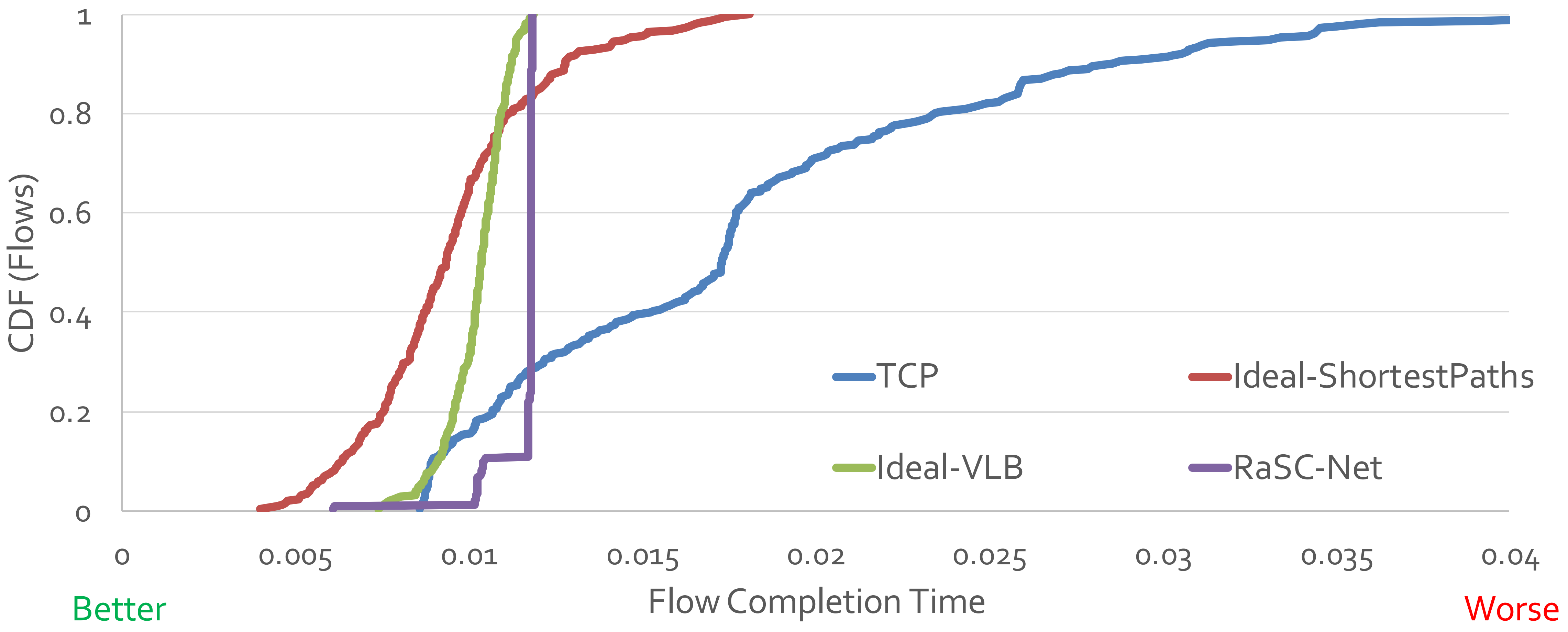
Flow Completion Time (99th perc.)



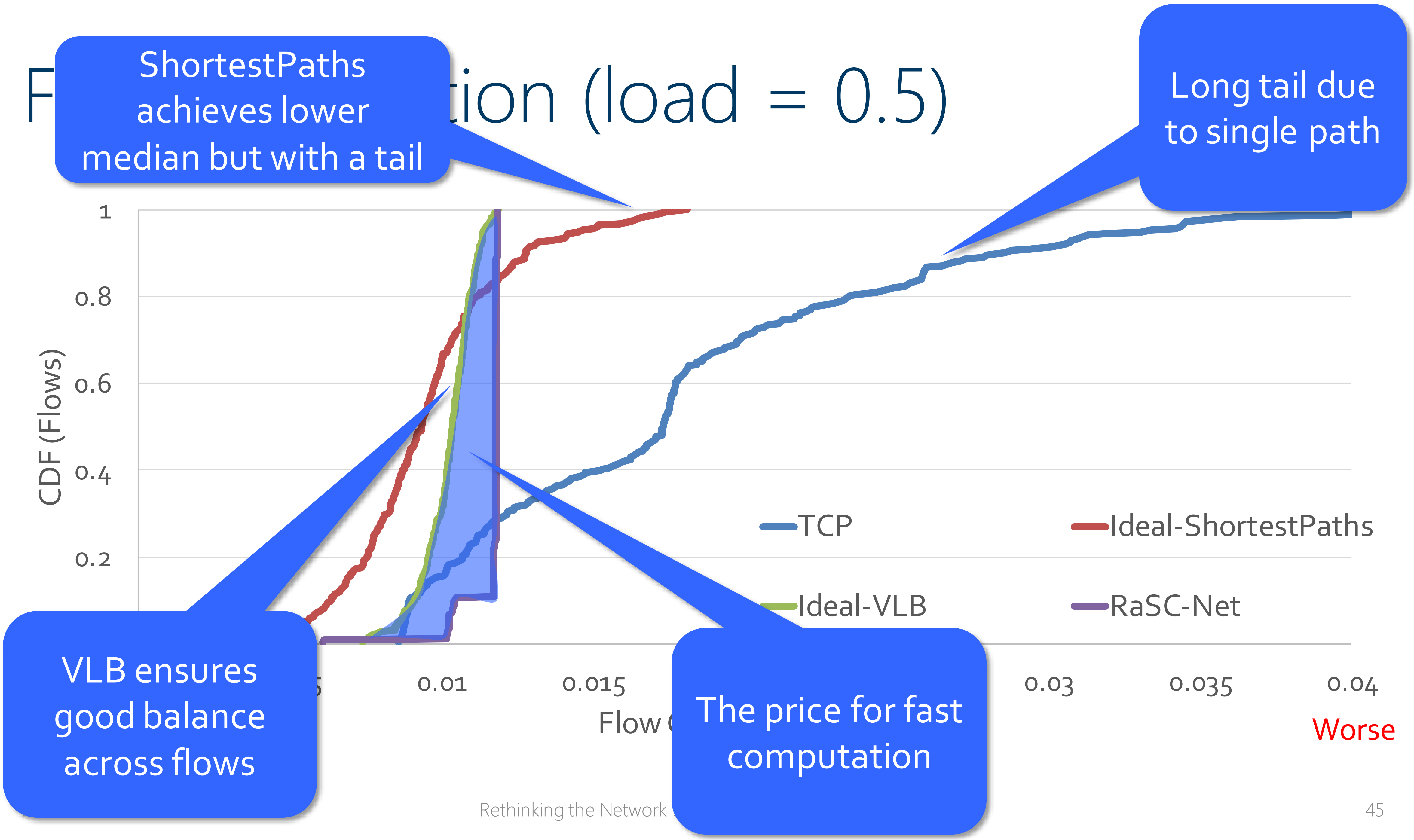
Flow Completion Time (99th perc.)



Flow Distribution (load = 0.5)



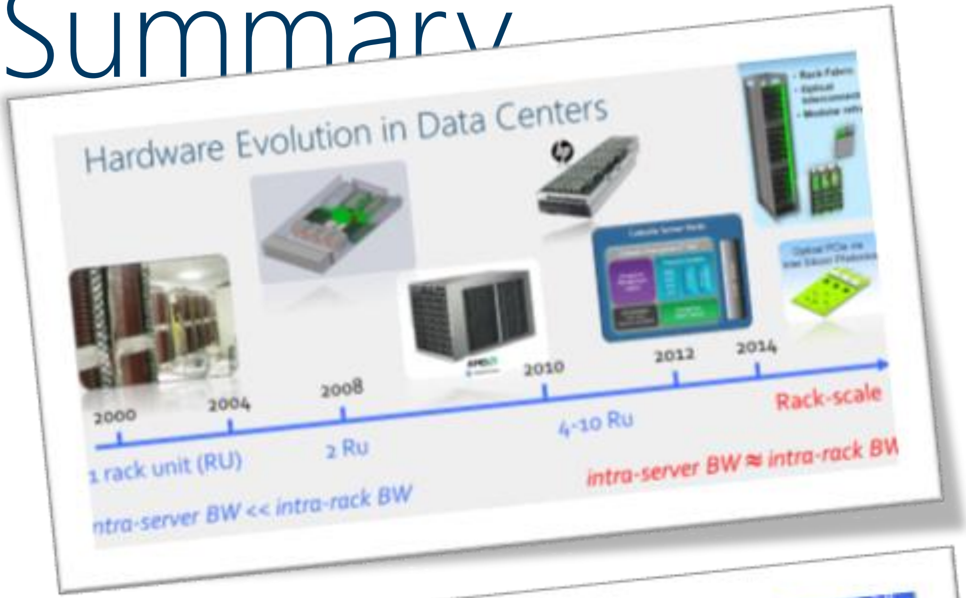
F tion (load = 0.5)

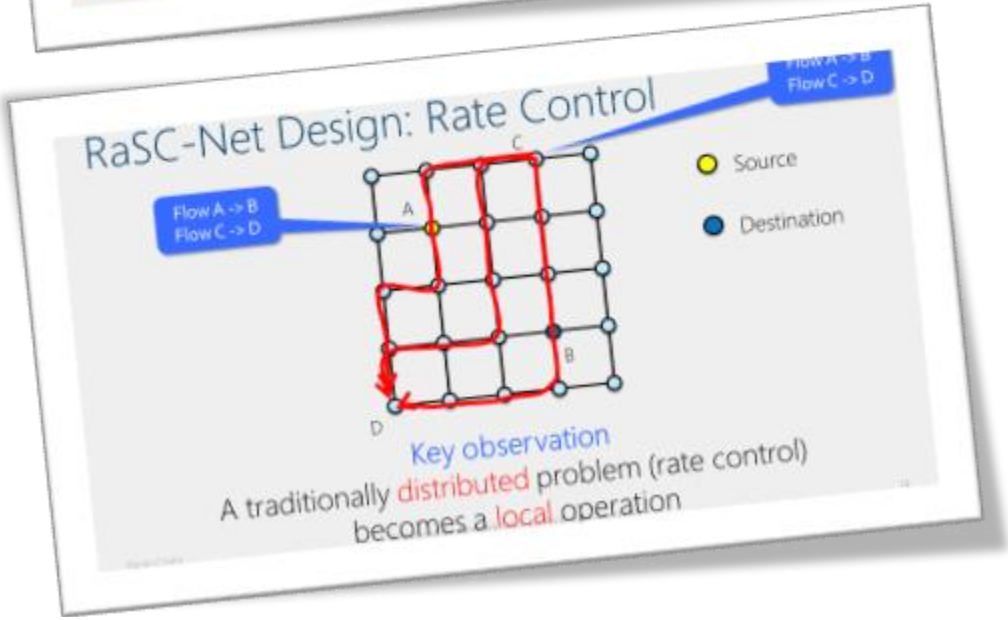


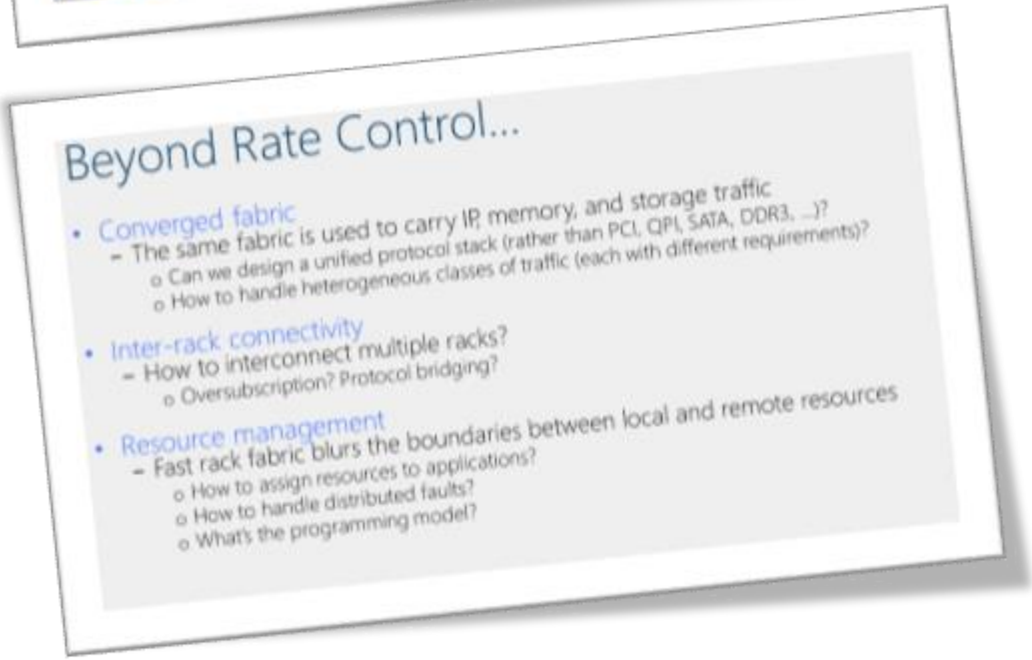
Beyond Rate Control…

1. Converged fabric
   - The same fabric is used to carry IP, memory, and storage traffic
     - Can we design a unified protocol stack (rather than PCI, QPI, SATA, DDR3, …)?
     - How to handle heterogeneous classes of traffic (each with different requirements)?
2. Inter-rack connectivity
   - How to interconnect multiple racks?
     - Oversubscription? Protocol bridging?
3. Resource management
   - Fast rack fabric blurs the boundaries between local and remote resources
     - How to assign resources to applications?
     - How to handle distributed faults?
     - What's the programming model?

Summary







Rack-scale Computers @ MSR Cambridge http://research.microsoft.com/rackscale