

Data Center Network Architecture

1. Two-stage Dual-mode forwarding
2. Load Balance Routing

Cheng-Chun Tu

Prof. Tzi-cker Chiueh

Goal

A **network architecture** that carries:

- 1 million VMs
- ~50,000 physical machines

Properties:

- Scalable
- Efficient
- Manageable
- Reliable

Outline

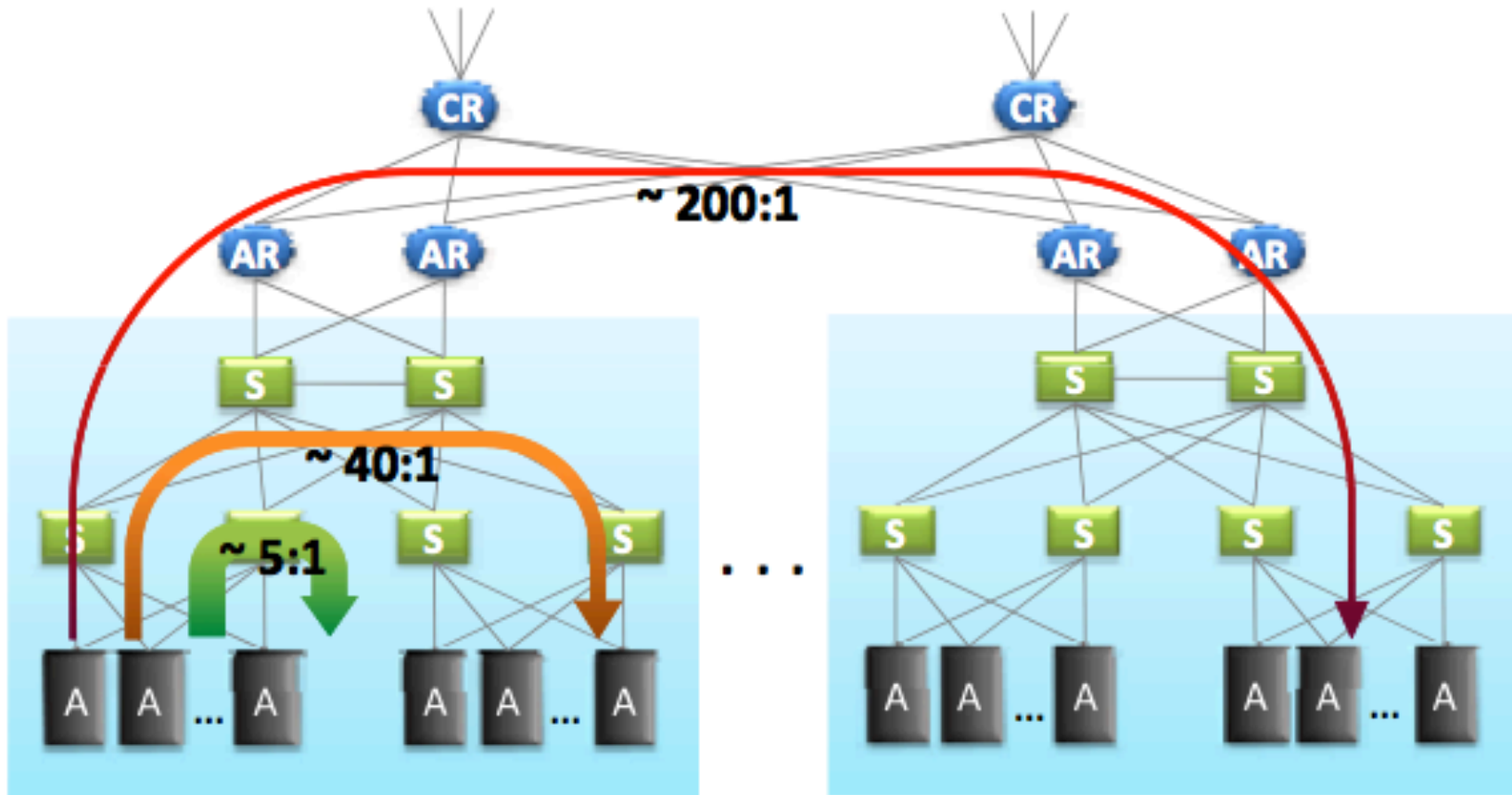
Find where the problem is:

- L3 + L2 Architecture
- Ethernet's scalability problems
- Cost Analysis

Solve the problem:

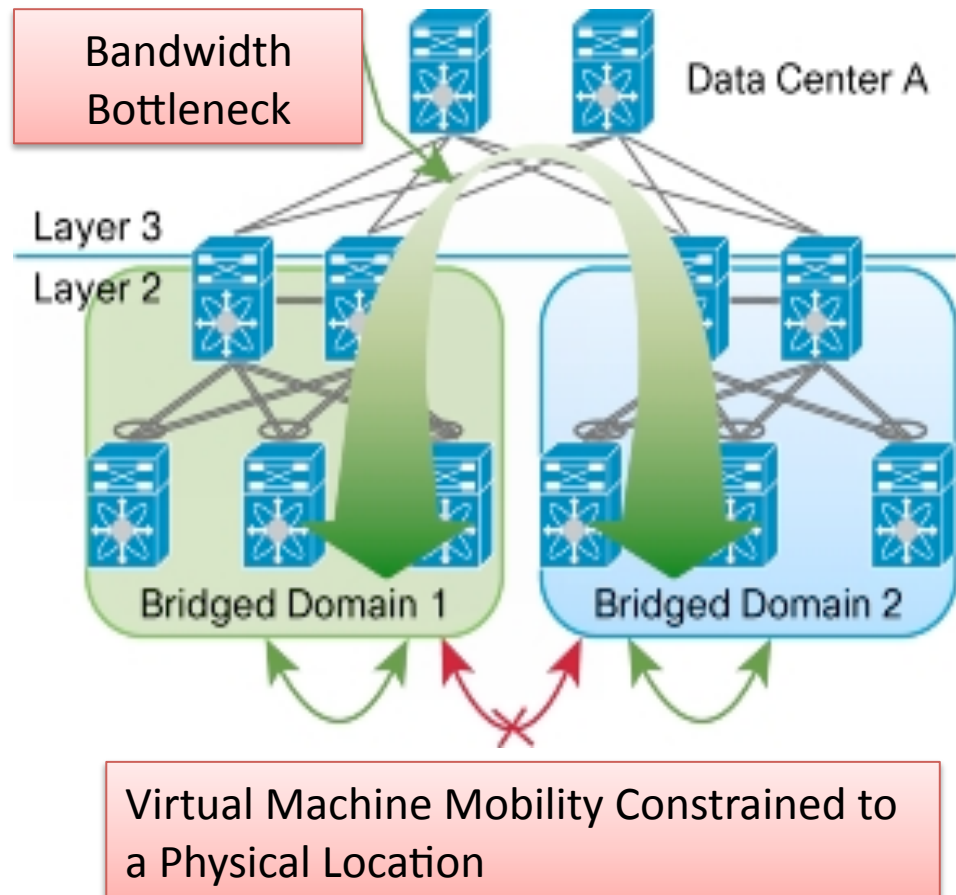
- CCMA All-L2 Network
 - Topology
 - Two-stage dual-mode forwarding
 - Fault-tolerant routing

L2 + L3 problem1: Efficiency



Tree topology, Spanning Tree Protocol

L2 + L3 Architecture: more problems



Problem2: Configuration:

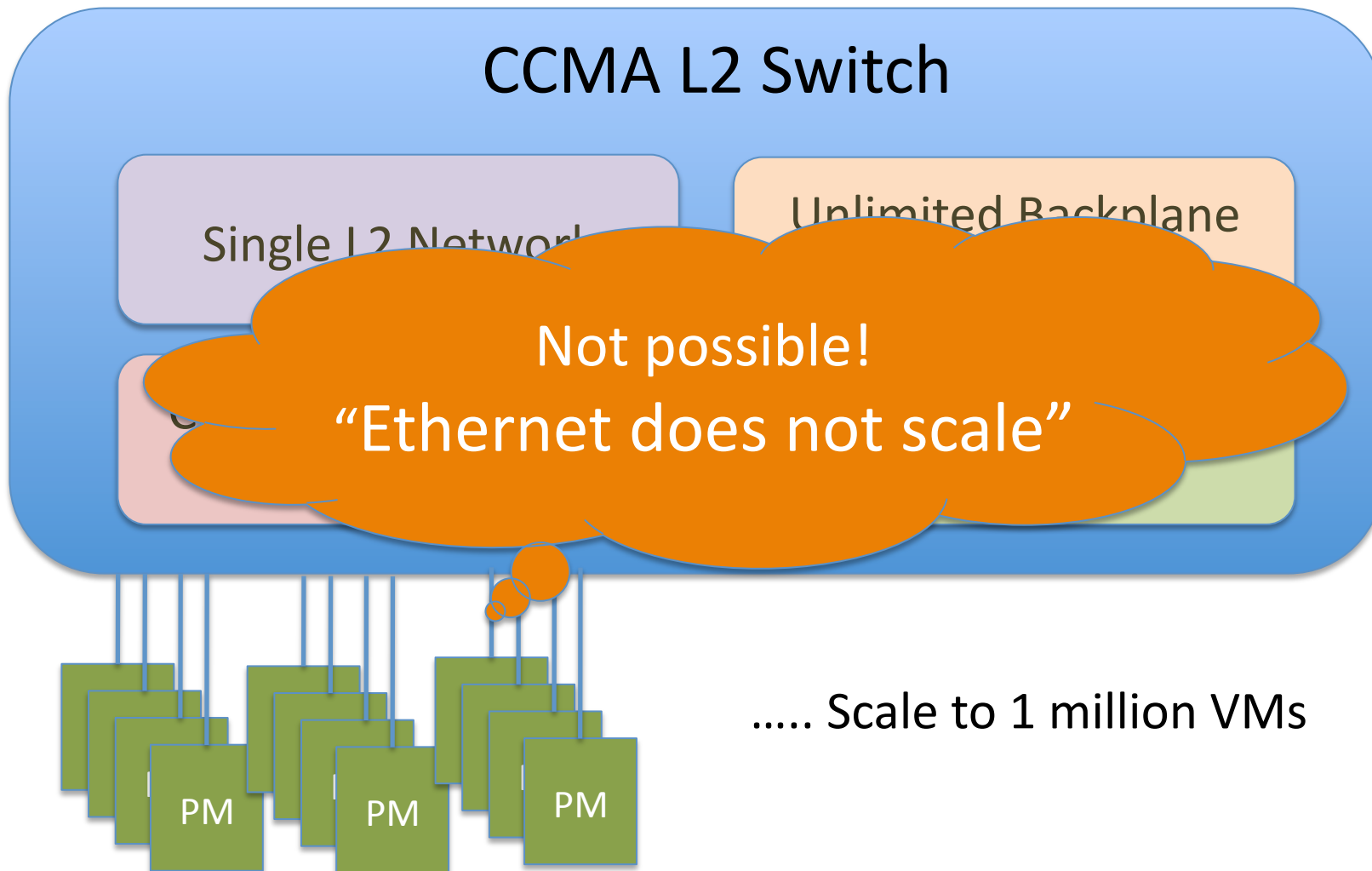
- Routing table in the routers
- IP assignment
- DHCP coordination
- VLAN and STP

Problem3: Scale up not Scale out:

Equipment higher in the hierarchy cost more and more efforts made at availability

4. Limited forwarding table size: Commodity switch 16-32k

All We Need Is Just A Huge L2 Switch!



What's Wrong with Ethernet?

Broadcast and flooding bootstrap protocol

- DHCP, ARP protocol rely on broadcast
- Locating unknown destination based on flooding
- **Lack of broadcast traffic scoping**

Mac-in-Mac +
Directory Service

Spanning Tree-based ensures loop free

- Not all physical links are used
- No load-sensitive dynamic routing
- Fail-over latency is high

Route Server

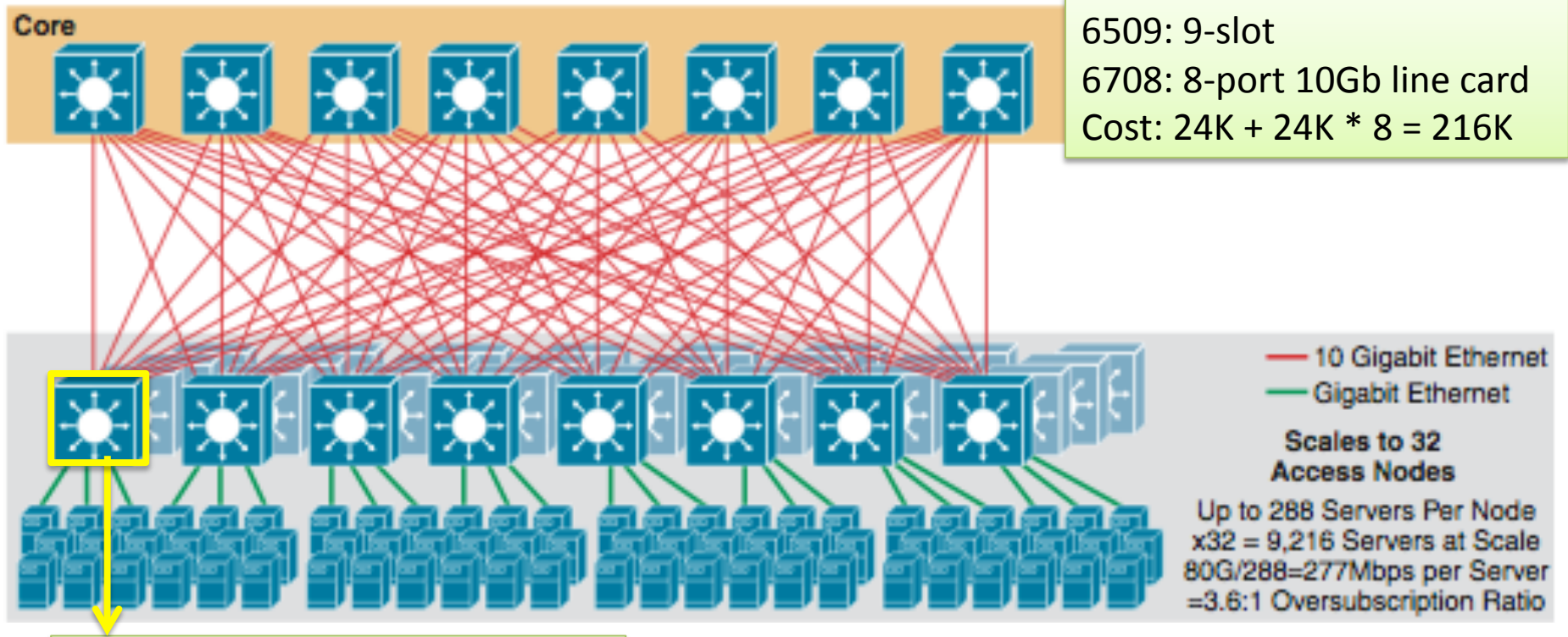
Populate forwarding table by self-learning

- Table size is limited
- Commodity switch with only 16 – 32k entries

Route Server

Cost Analysis: Cisco's Two-Tier Model

Goal: interconnect **9,216** servers using 8-way ECMP with 8 core node. Oversubscription: 3.6:1, **277Mbps per server**

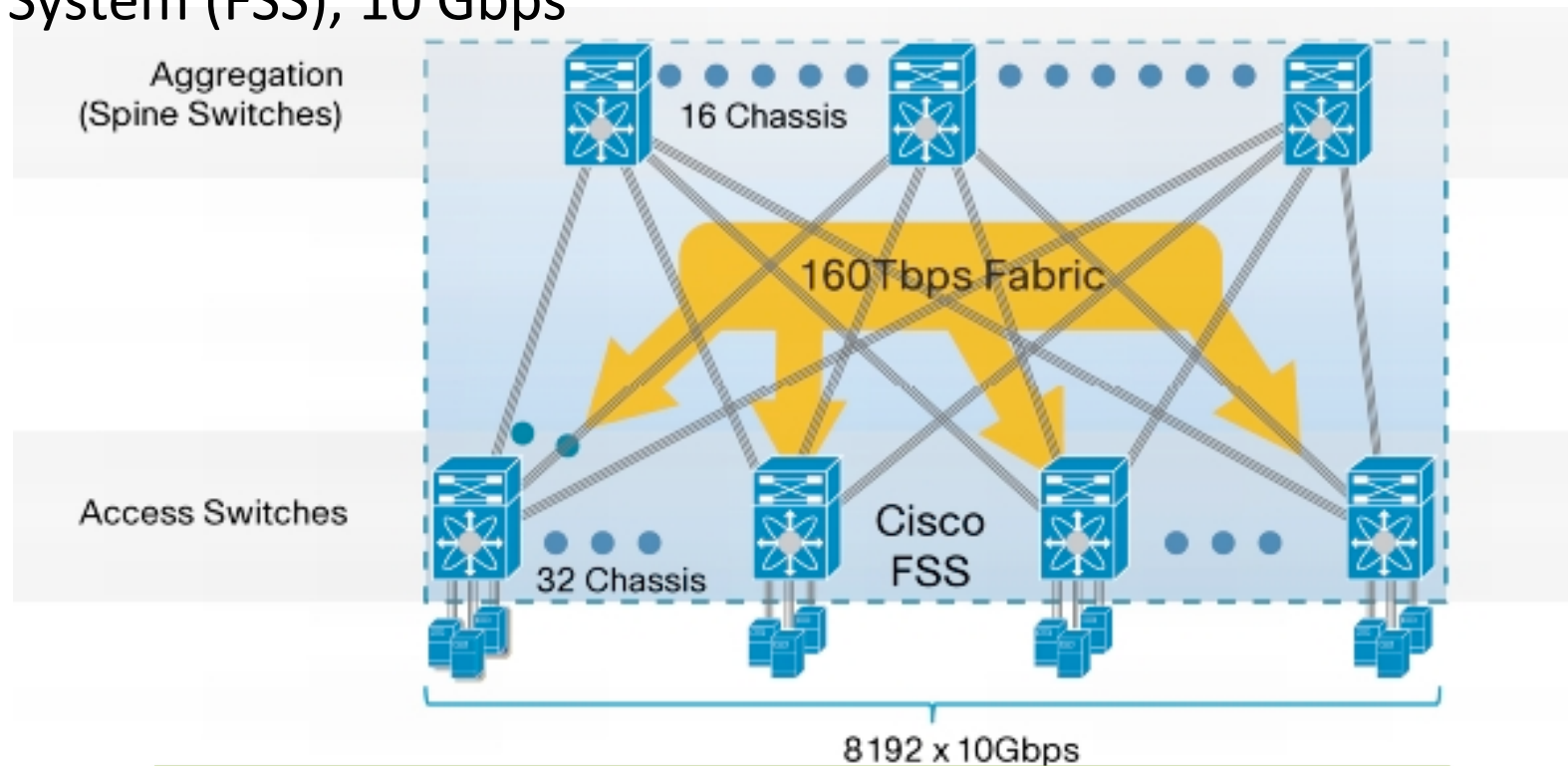


Chassis+8 port line card + switch
6 switches, 48 port = 288 servers
4948: 48-port 10Gb (7K)
Cost: 24K + 24K + 7K * 6 = 90K

Core: 216K US
Access: 90K * 32 = 2880K US
Total cost: 3096K = 3M US

Cost Analysis: Cisco's FSS

Goal: interconnect **8,192** servers using FabricPath Switching System (FSS), 10 Gbps


















Nexus 7000 Chassis: 14K US

Nexus 7000 32 port 10Gb switch: 43K US

Each chassis consists of 256 ports to servers = 16 switches

Total cost: $48 * 14K + 16 * 48 * 43K = 33696K$ US = 33M US

Can we take the best of all?

Features \ Architectures	Ethernet Bridging	IP Routing	CCMA L2
Configuration overhead			
Host mobility			
Path efficiency			
Forwarding table usage			
Tolerance to loop			

Outline

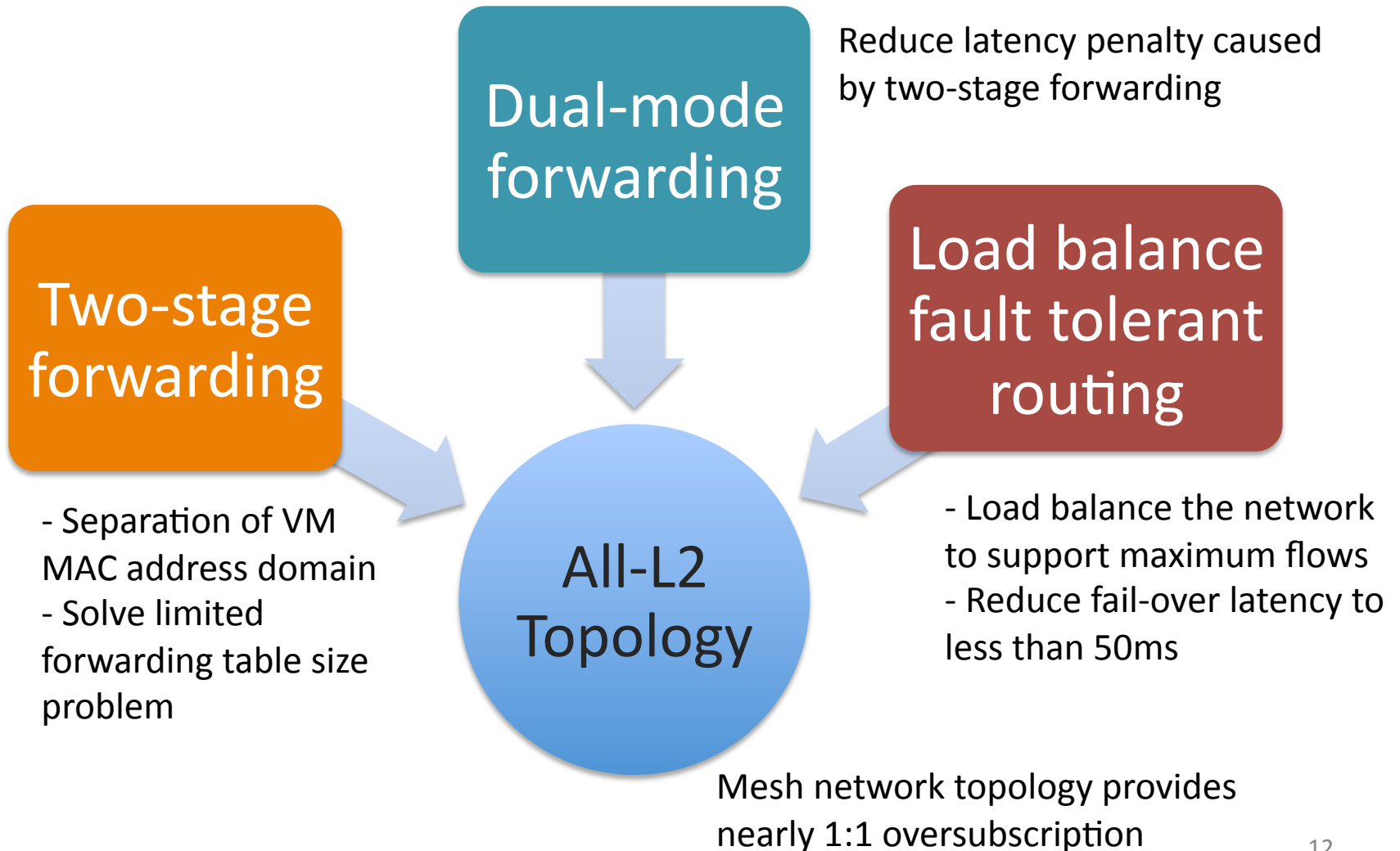
Find where the problem is:

- L3 + L2 Architecture
- Ethernet's scalability problems
- Cost Analysis

Solve the problem:

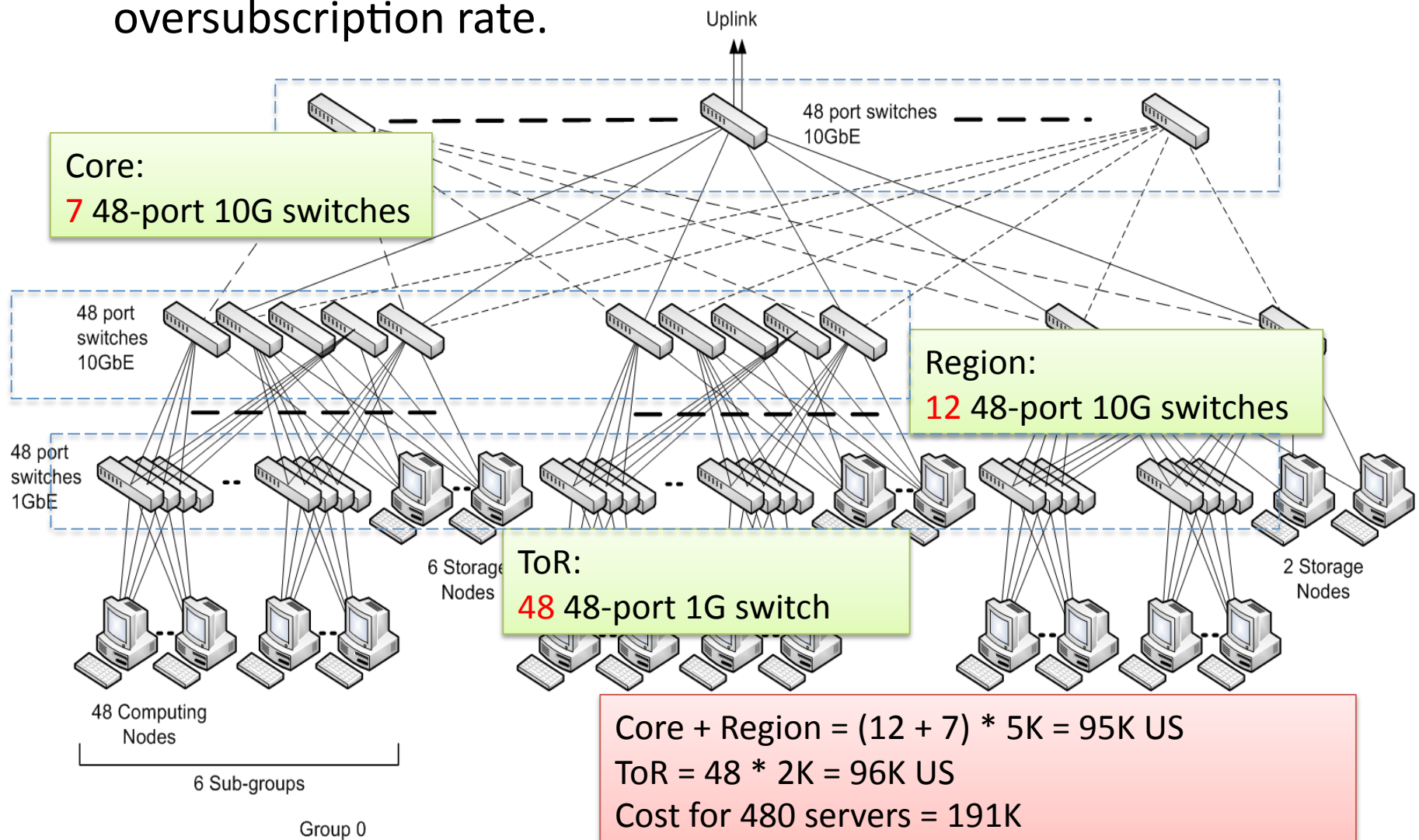
- CCMA All-L2 Network
 - Topology
 - Two-stage dual-mode forwarding
 - Fault-tolerant routing

CCMA All-L2 Design



CCMA All-L2 Network Topology

Container Computer supporting **480** servers with 1:1 oversubscription rate.



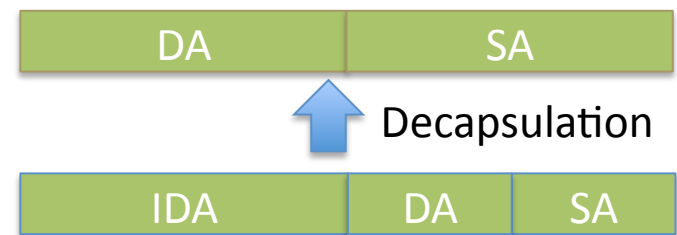
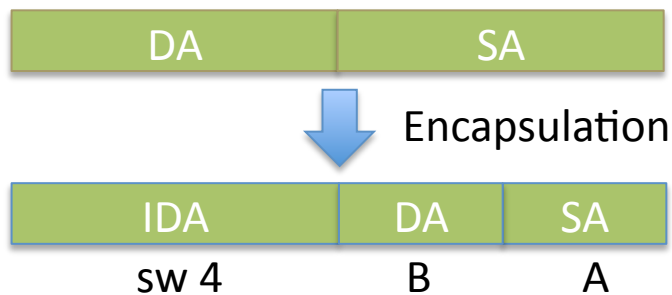
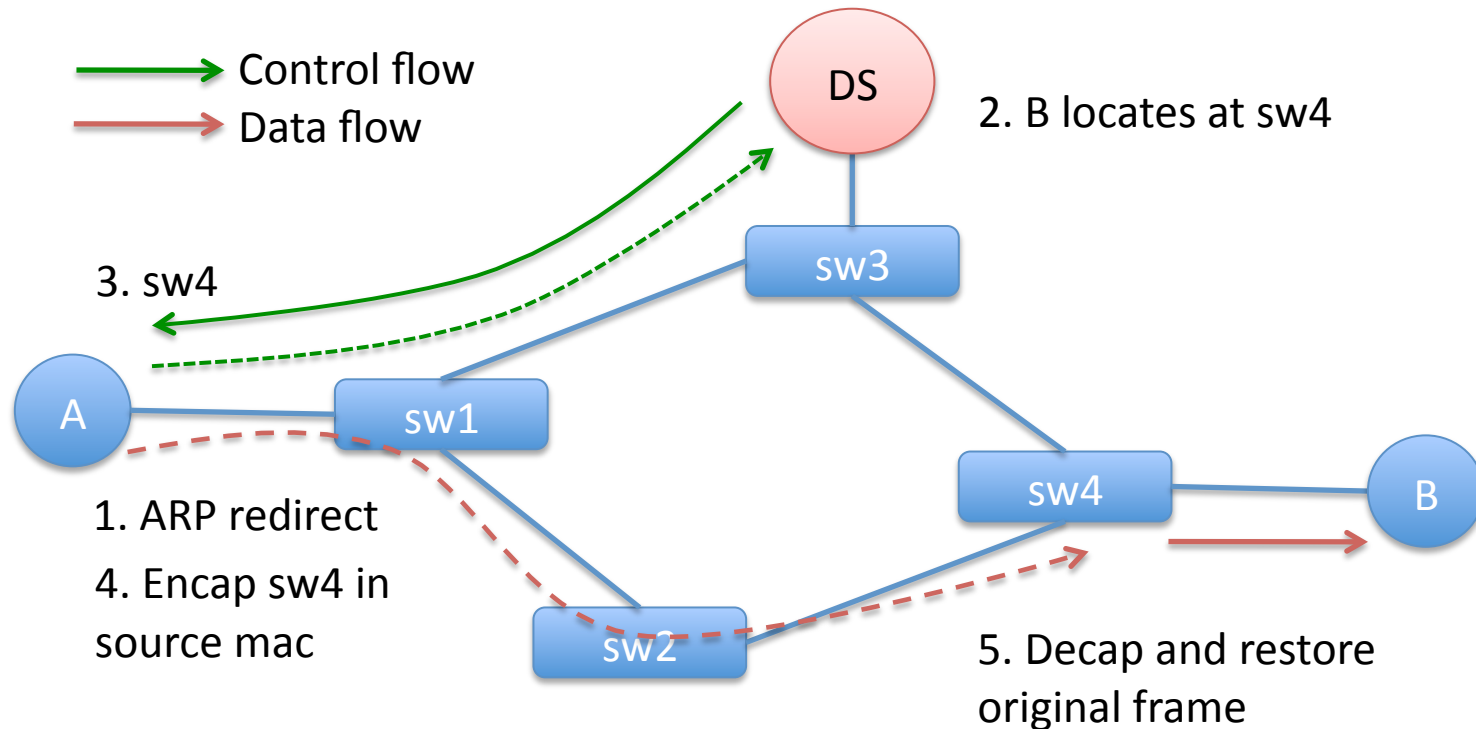
Core:
7 48-port 10G switches

Region:
12 48-port 10G switches

ToR:
48 48-port 1G switch

Core + Region = $(12 + 7) * 5K = 95K$ US
ToR = $48 * 2K = 96K$ US
Cost for 480 servers = 191K
Cost for 9,600 servers = 2400K = **3.82M US**

Two Stage Forwarding



Summary

Benefits

- Switch routes according to IDA, not DA
- VM mac is invisible to Intermediate switch
- Flexible and backward compatible compared with other tagging techniques

Drawbacks:

- Switch has limited computational power
 - Deploy at Dom0

Dual-mode Forwarding

- Problems of two-stage forwarding:
 - Control plane packet processing in commodity switches is too slow
- Solution: **Dual-mode** forwarding
 - **Direct**: source → destination
 - **Indirect**: source → intermediate → destination
- Effect:
 - Performance of direct route and generality of indirect mode
- Direct routing optimization:
 - When two VMs **frequently** talk to each other, create a direct route for them (how?)

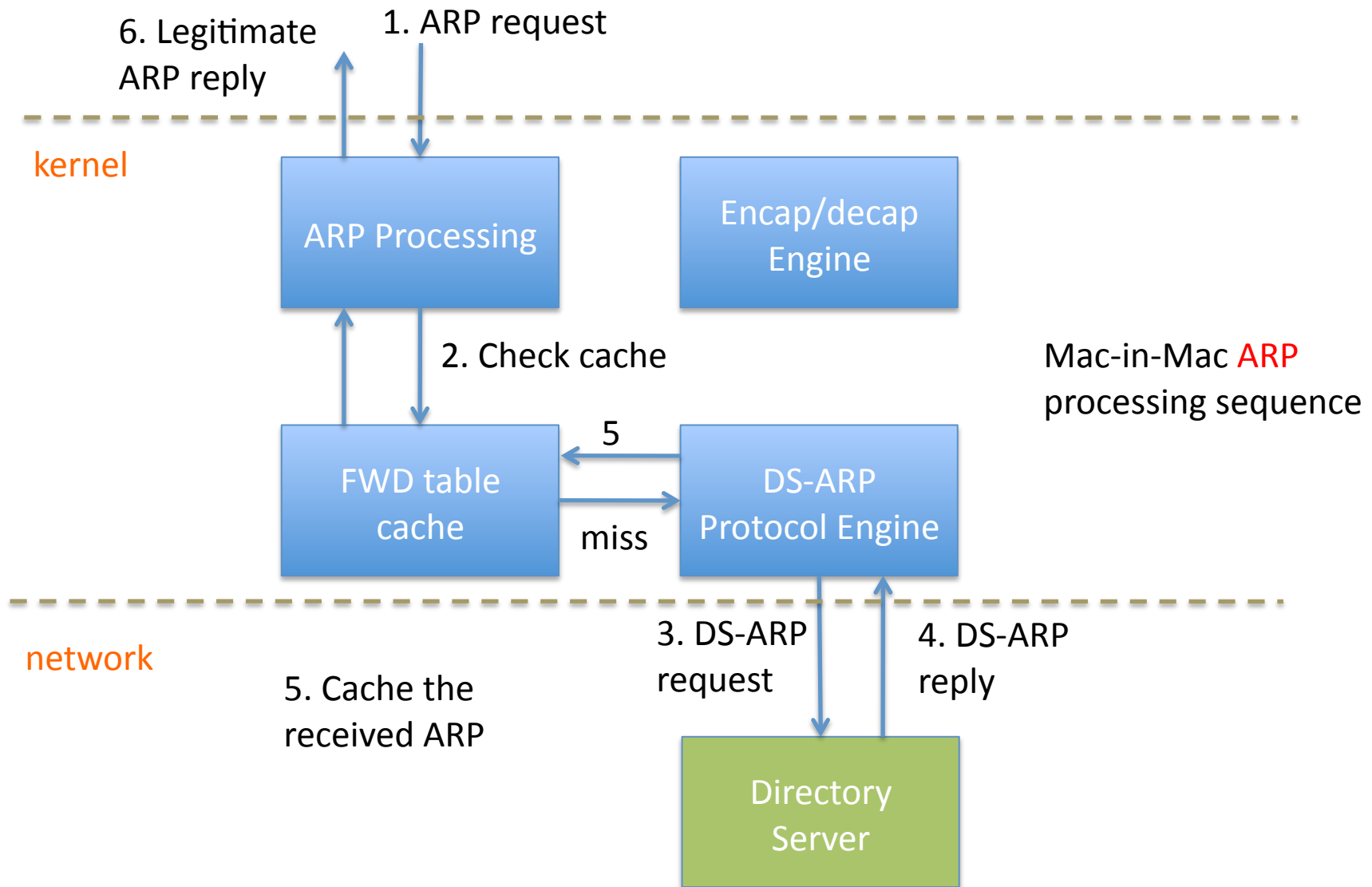
Traffic-based Algorithm to Optimize Forwarding Table Usage

- **Traffic Matrix:** $N \times N$, where N is the number of physical nodes and switches
 - Each entry represents traffic volume from X node to Y node
- 1. Order all traffic matrix entries in decreasing order
- 2. For each $\langle S, D \rangle$ pair, **assign a forwarding table entry in each switch along the path from S to D**
- 3. Continuing on with the remaining traffic matrix entries until they are exhausted or all forwarding table entries are depleted

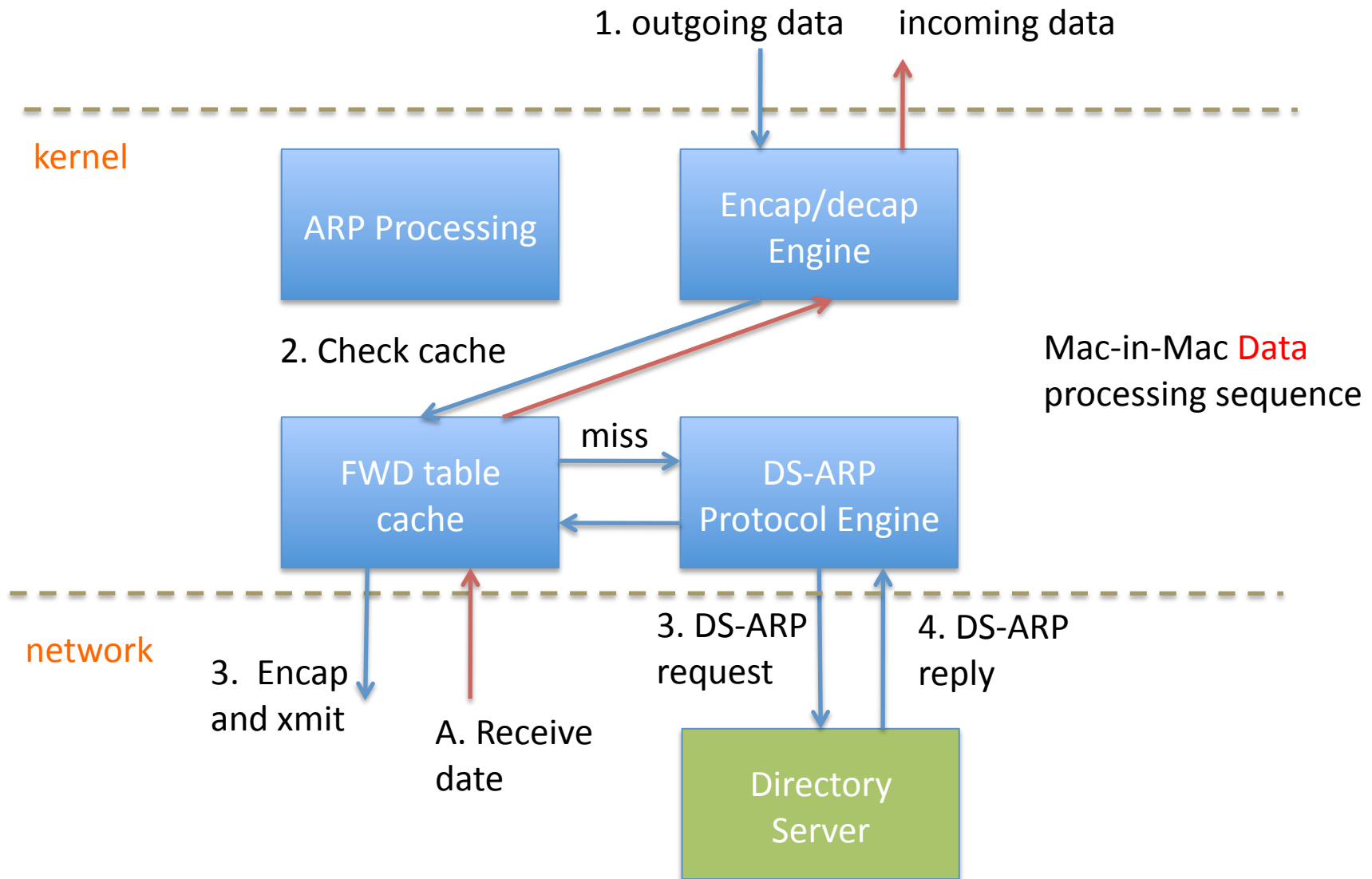
Mac-in-Mac Kernel Module

IMPLEMENTATION

Mac-in-Mac Implementation



Mac-in-Mac Implementation

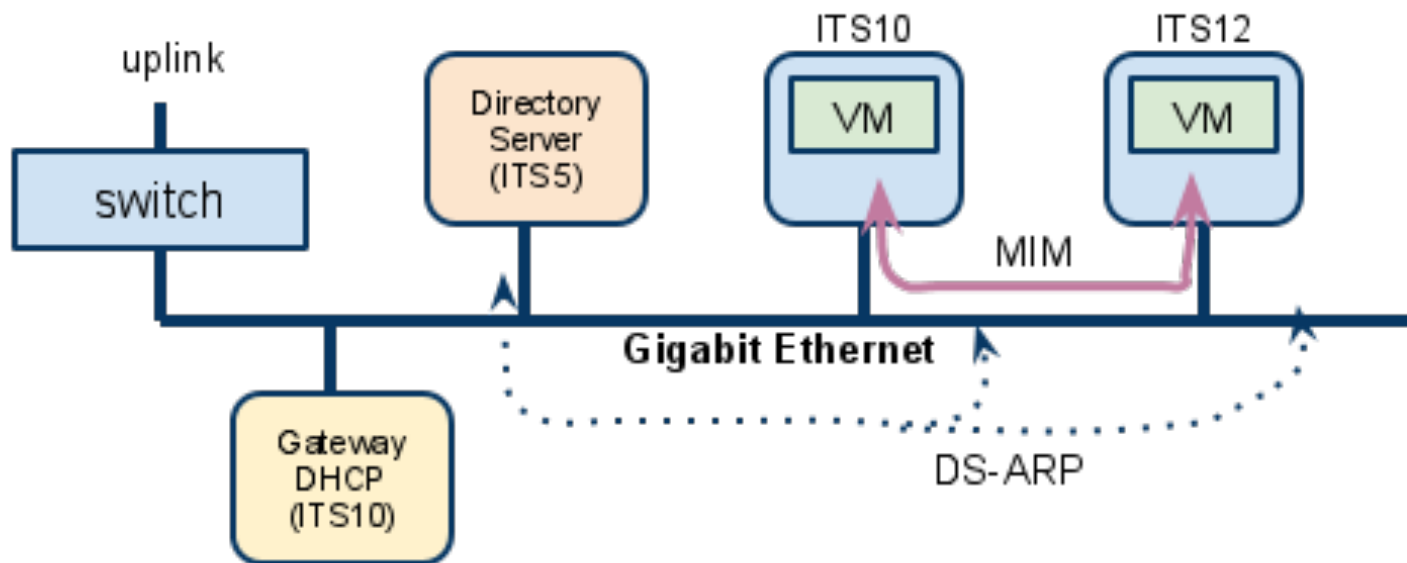


Performance

Testbed:

4 Inventec server

1 Edgecore switch



Performance

Protocol	Number of pkt	Packet Size (byte)	MB/Sec (with MIM)	MB/sec (No MIM)
UDP	1,000,000	1024	128	128
UDP	2,000,000	1024	122	121
UDP	3,000,000	1024	120	119
<i>TCP</i>	1,000,000	1024	128	128
<i>TCP</i>	2,000,000	1024	120	122
<i>TCP</i>	3,000,000	1024	119	118

Outline

Find where the problem is:

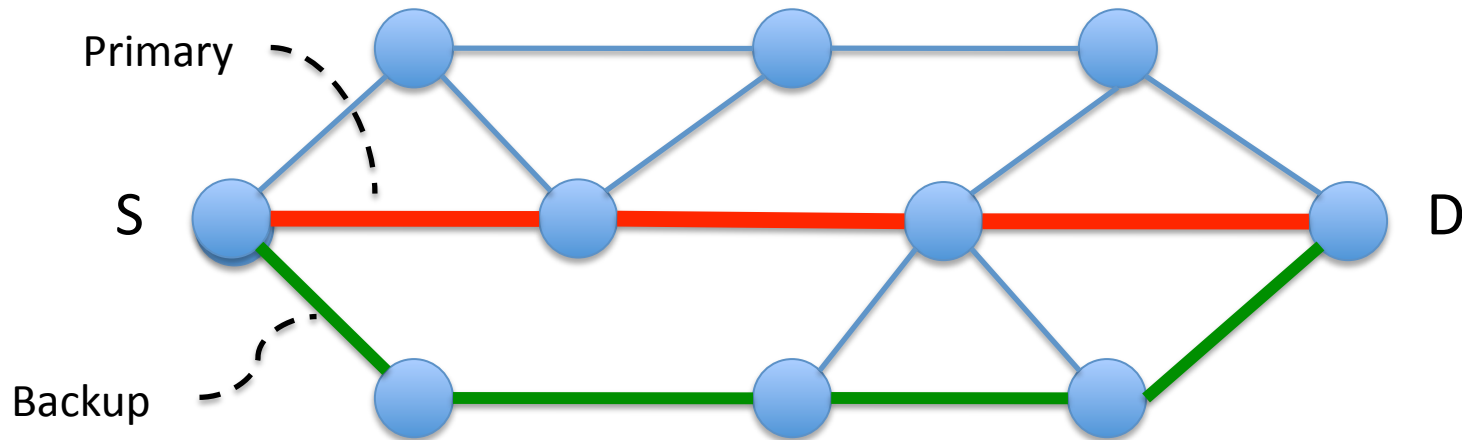
- L3 + L2 Architecture
- Ethernet's scalability problems
- Cost Analysis

Solve the problem:

- CCMA All-L2 Network
 - Topology
 - Two-stage dual-mode forwarding
 - **Load balance routing**

Dynamic Load Balancing Routing Algorithm for Data Center Network

Goal



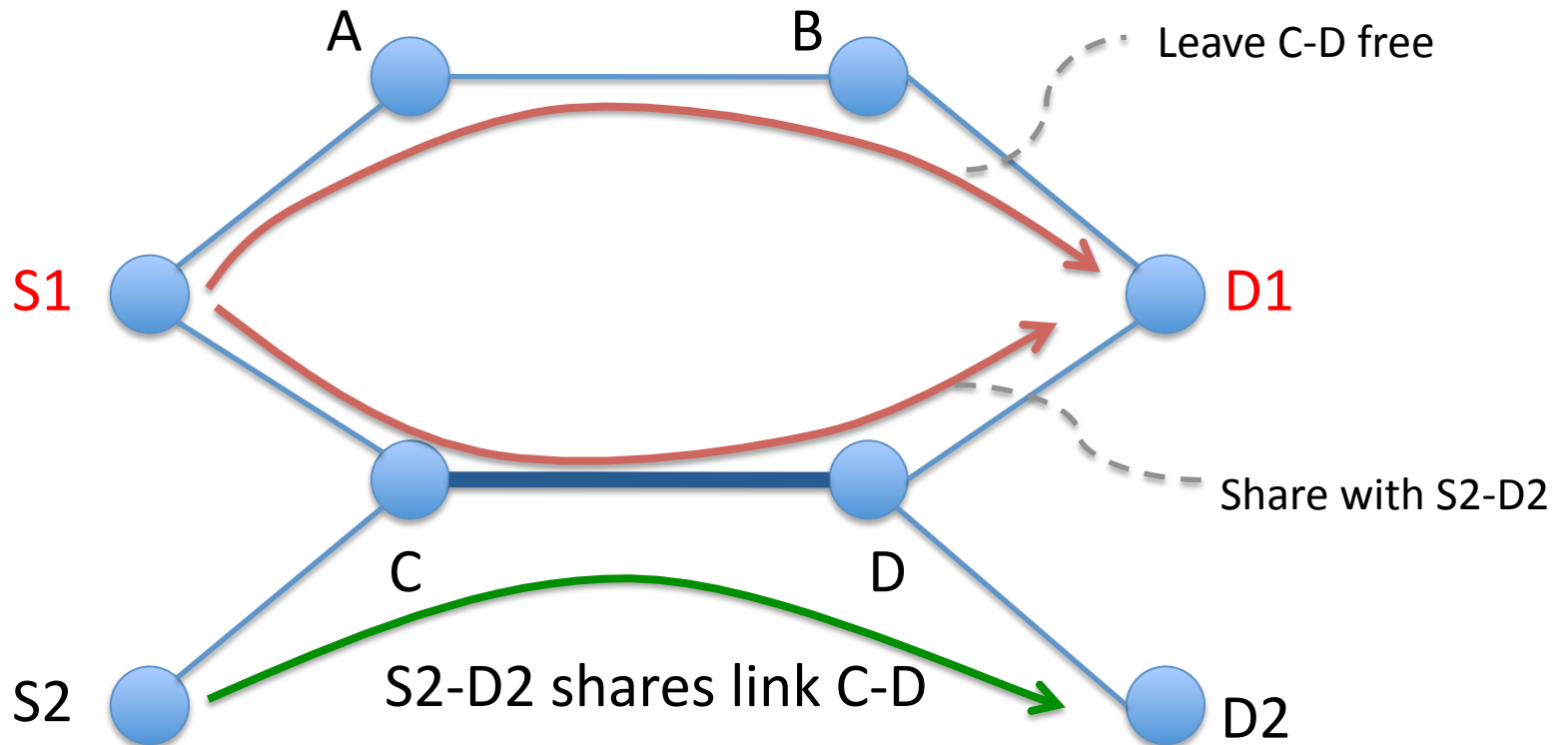
- Given a mesh network and traffic profile
 - Load balance the network resource utilization
 - Prevent congestion by balancing the network load to support as many traffic load as possible
 - Provide fast recovery from failure
 - Provide primary-backup route to minimize recovery time

Factors

- Only hop count
- Hop count and link residual capacity
- Hop count, link residual capacity, and link expected load
- Hop count, link residual capacity, link expected load and additional forwarding table entries required

How to combine them into one number for a particular candidate route?

Route Selection: idea

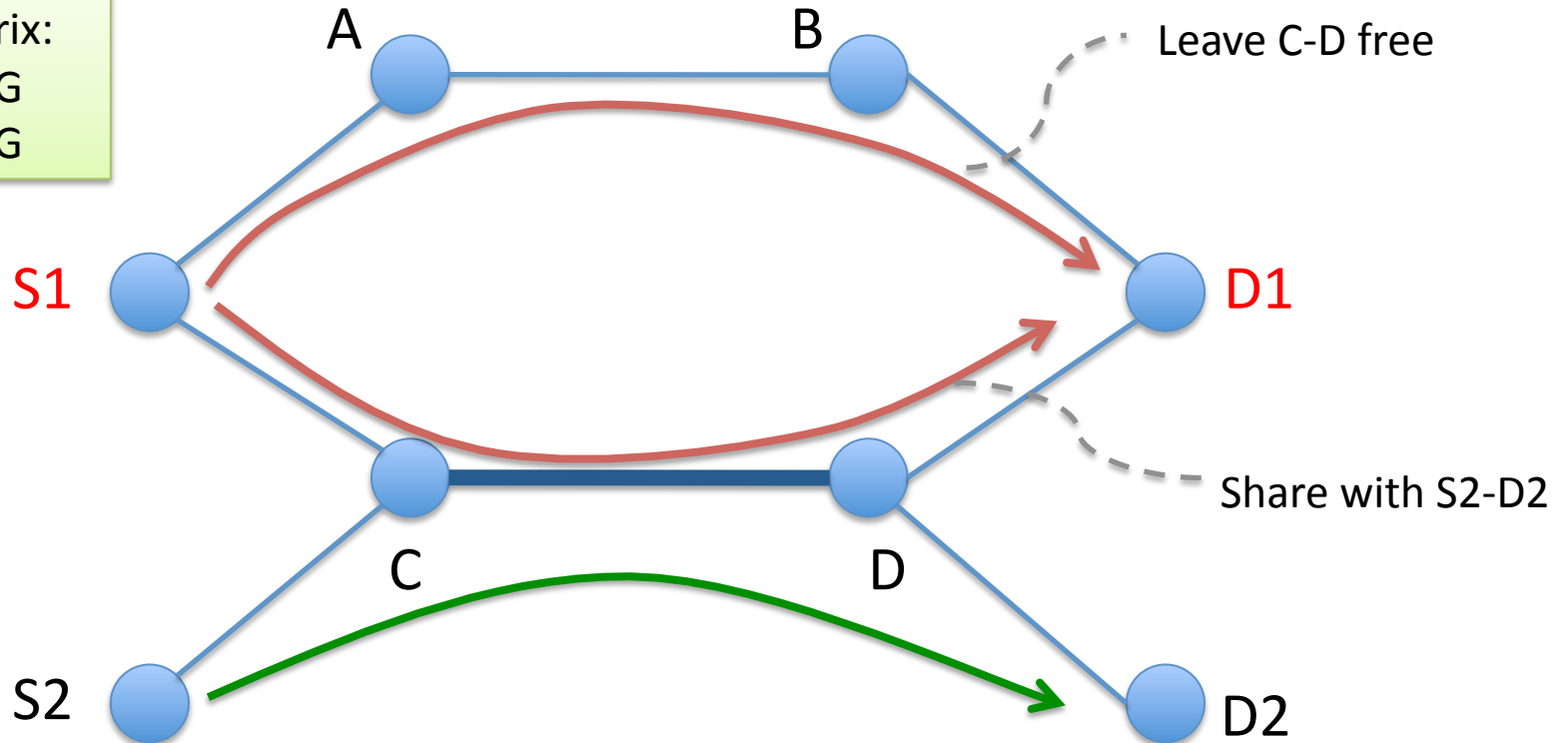


Which route is better from S1 to D1?

Link C-D is more important!
Idea: use it as sparsely as possible

Route Selection: hop count and Residual capacity

Traffic Matrix:
S1 -> D1: 1G
S2 -> D2: 1G



Using Hop count or residual capacity makes no difference!

Determine Criticality of A Link

Determine the importance of a link

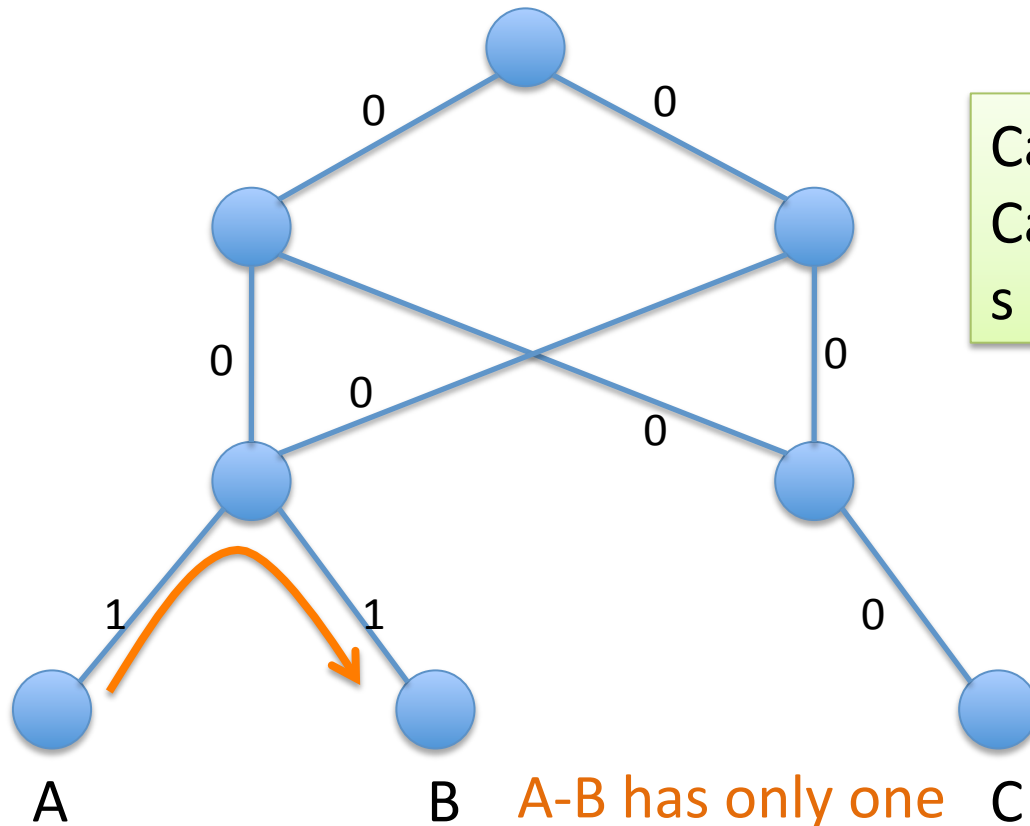
= fraction of all (s, d) routes that pass through link l

Expected load of a link at initial state

= Bandwidth demand matrix for s and d

Criticality Example

Consider a network with three host A, B, and C, each connects to the switch network

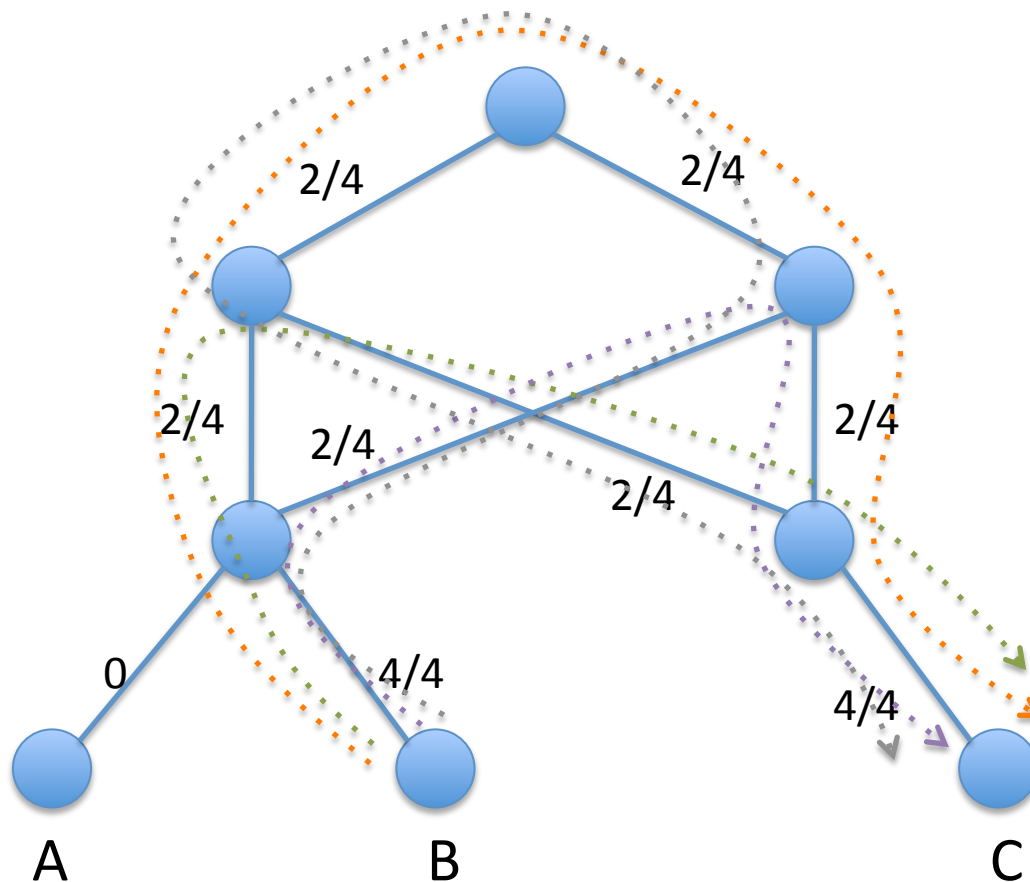


Case1:
Calculate
 $s = A, d = B$

A-B has only one
route

Criticality Example

From **B** to **C** has four possible routes.



Case2:

Calculate

$s = B, d = C$

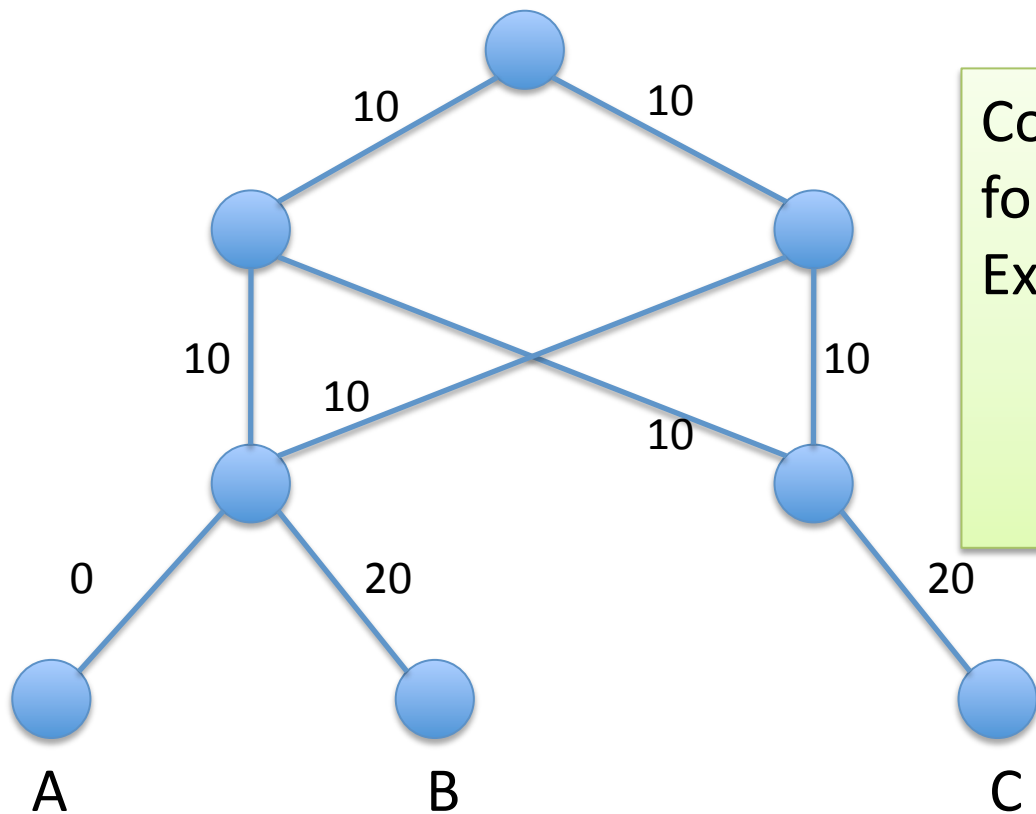
Case3:

$s = A, d = C$ is

similar

Expected Load

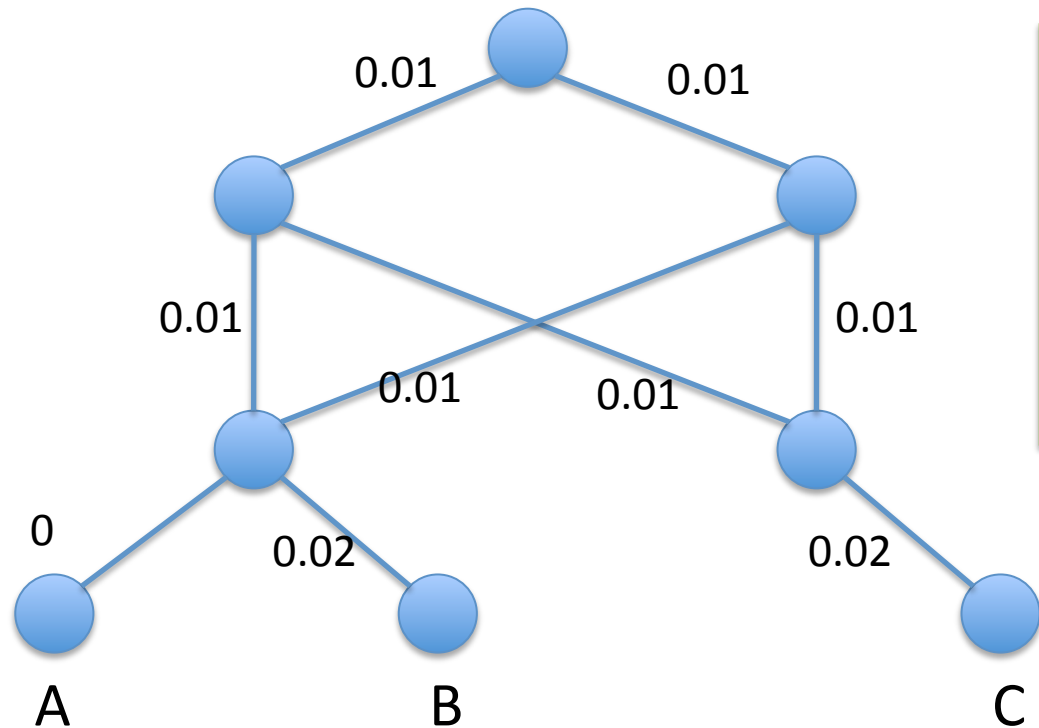
Assumption: load is equally distributed over each possible route between S and D.



Consider bandwidth demand for B-C is 10.
Expected Load:

Cost Metrics

Cost metric represents the expected load per unit of available capacity on the link

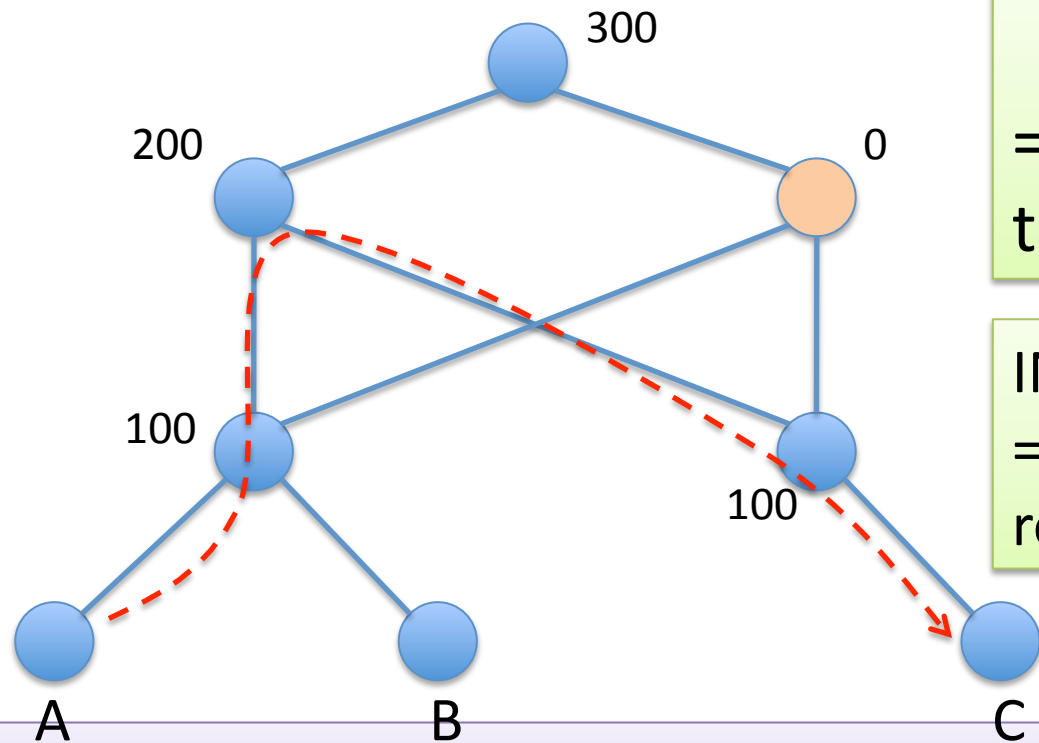


= Expected Load
= Residual Capacity

Idea: pick the link with minimum cost

Forwarding Table Metric

Consider using commodity switch with 16-32k forwarding table size.

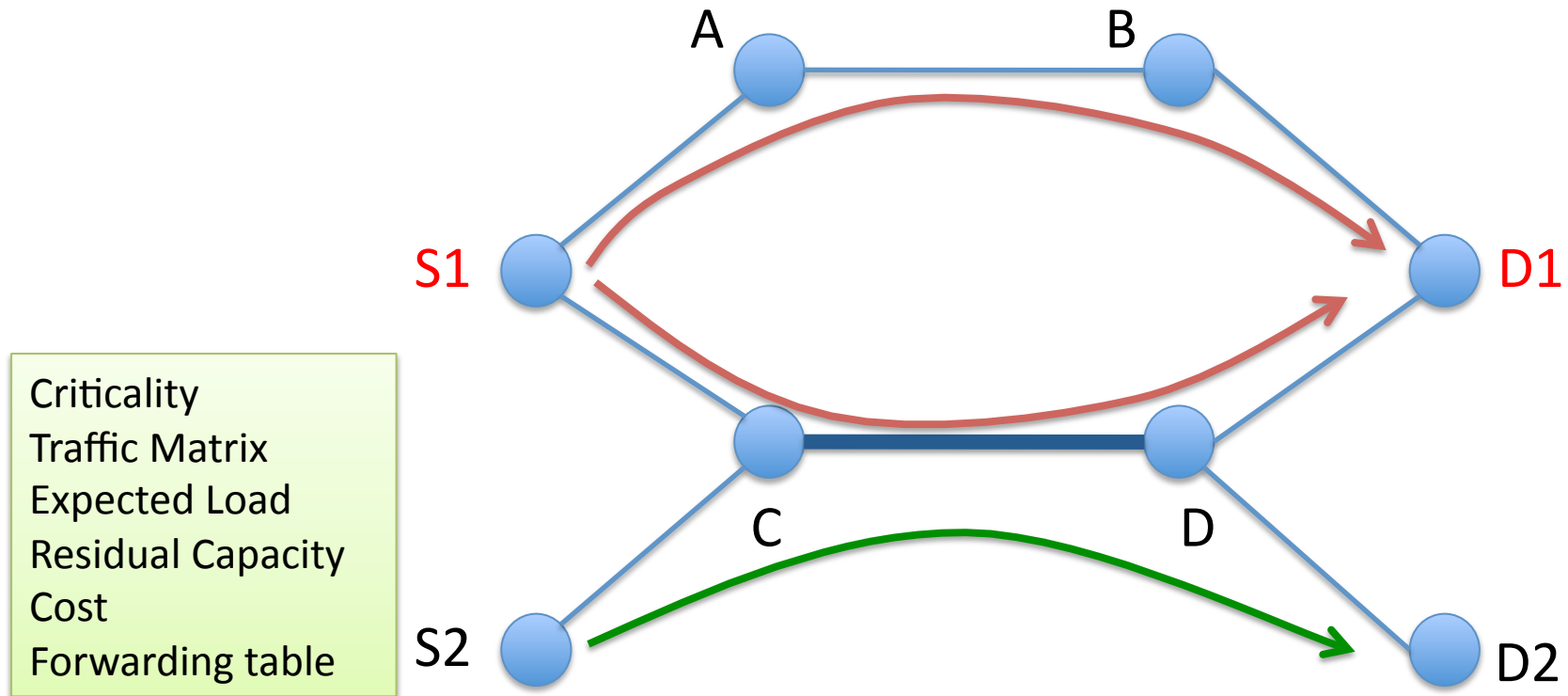


= available forwarding table entries at node n

INC_FWD
= extra entries needed to route A-C

Idea: minimize entry consumption, prevent forwarding table from being exhausted

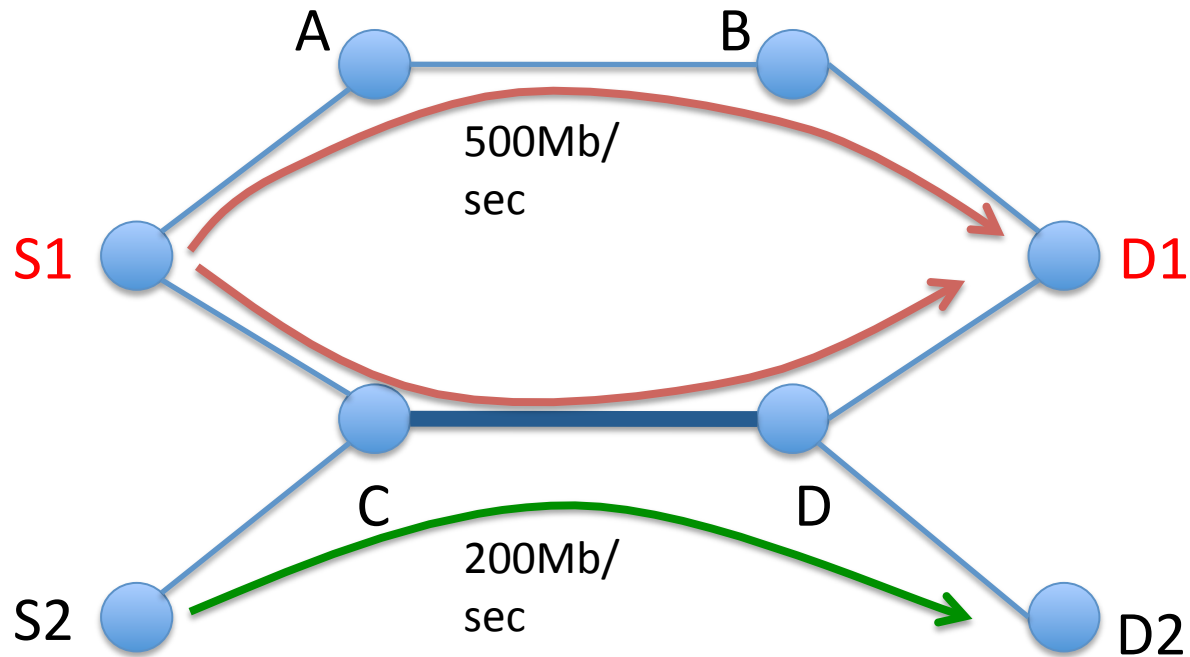
Route Selection: recap



How do we start? S1->D1 first or S2->D2?

Route Selection Order

<S1, D1>
<S2, D2>
<S3, D15>
<S4, D15>
<S5, D19>
.....
<S30, D41>
<S31, D21>
<S18, D71>
.....



1. Order the <S, D>s in decreasing traffic volume order
2. Start with the big guy until a threshold
3. The remaining <S, D>s are routed via a random routing algorithm -> reduce computational overhead!

Route Selection Process

Step1.

Order all traffic matrix entries in decreasing order.

Step2.

For each $\langle S, D \rangle$ pair in the matrix, find all possible routes between $\langle S, D \rangle$.

Step3.

Pick a pair of **disjoint** routes (primary and backup) from it by calculating the the value v .

Step4.

Update expected load on each link and forwarding table entry on each node.

Step5.

Continue on the remaining traffic matrix until they are exhausted.

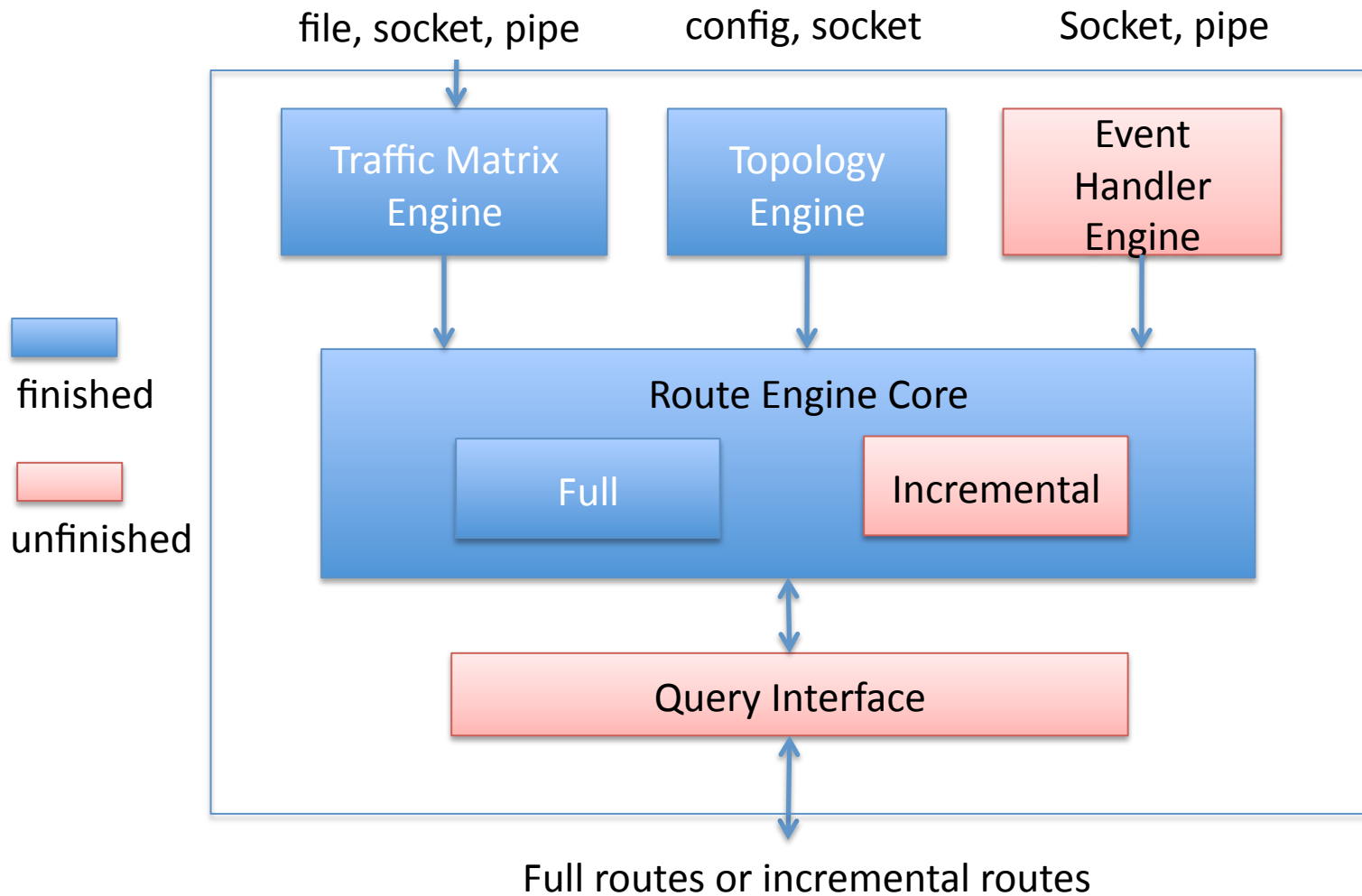
Incremental Routing

- Route Engine trigger incremental route when
 - Link congestion (by SNM)
 - Link/Switch down (by SNMP trap)
 - New VM boot-up or VM Migration (RPM)
- Incremental Routing
 - Figure out the affected pairs
 - Compute new route for them

Route Engine

IMPLEMENTATION

Implementation of Route Engine



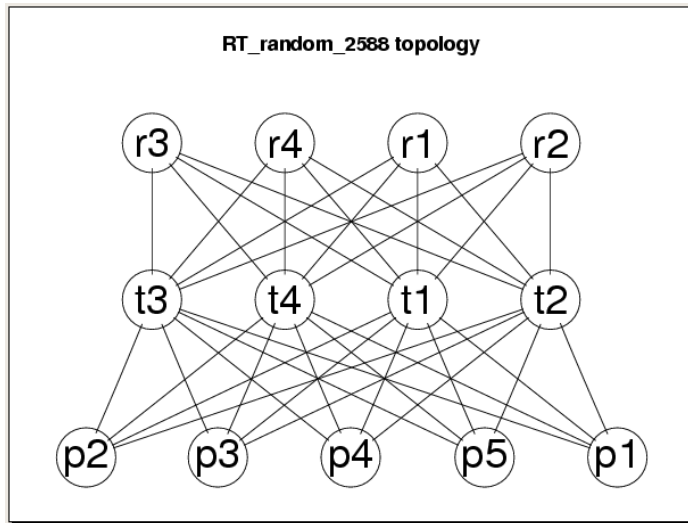
Route Selection Algorithm

- Load balance metric:
 - Load-balanced: the variation of link usage is minimized
 - Take standard deviation of all link usages
 -
- **Algo1**: mini hop
 - Pick the route with minimum hop count
- **Algo2**: WSP
 - pick the route with the maximum of the minimum of all links in each candidate route
- **Algo3**: K shortest path
 - Pick the route with minimum sum of cost, and that requires the smallest number of additional forwarding table entries.

Verification: route

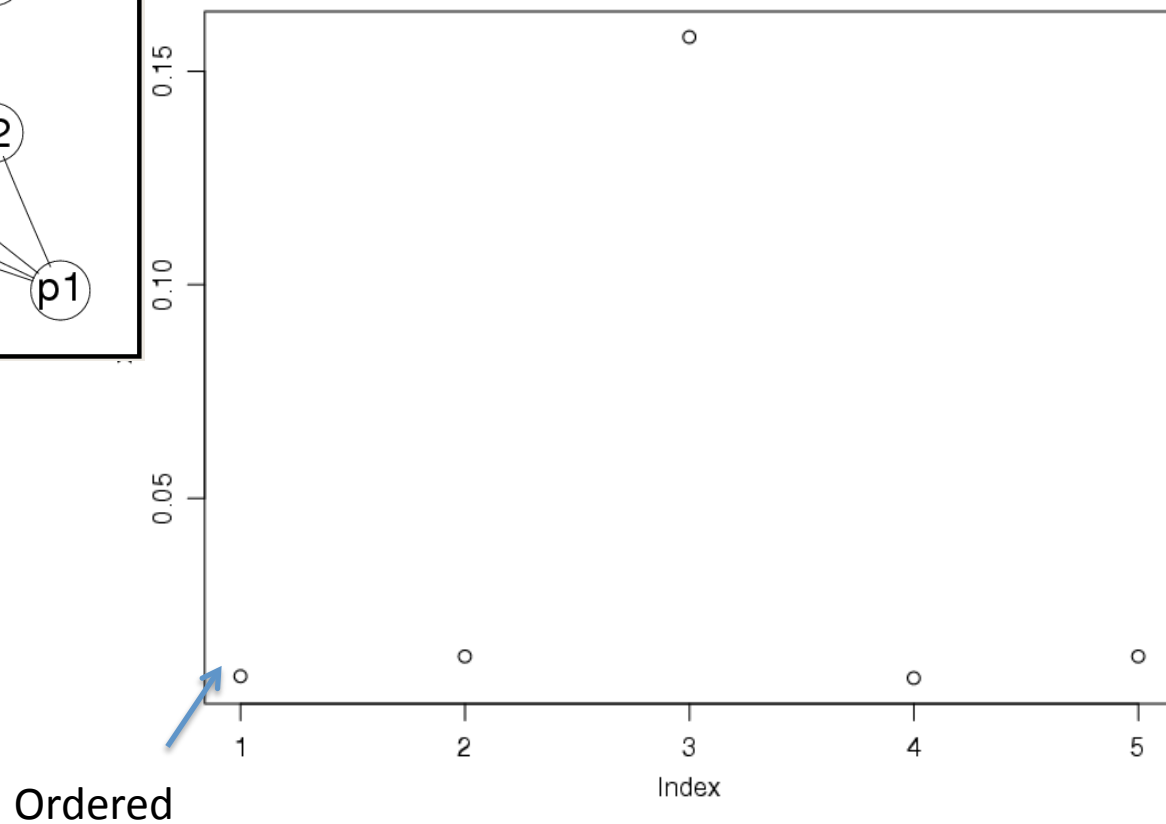


Performance Result of Route Selection Order

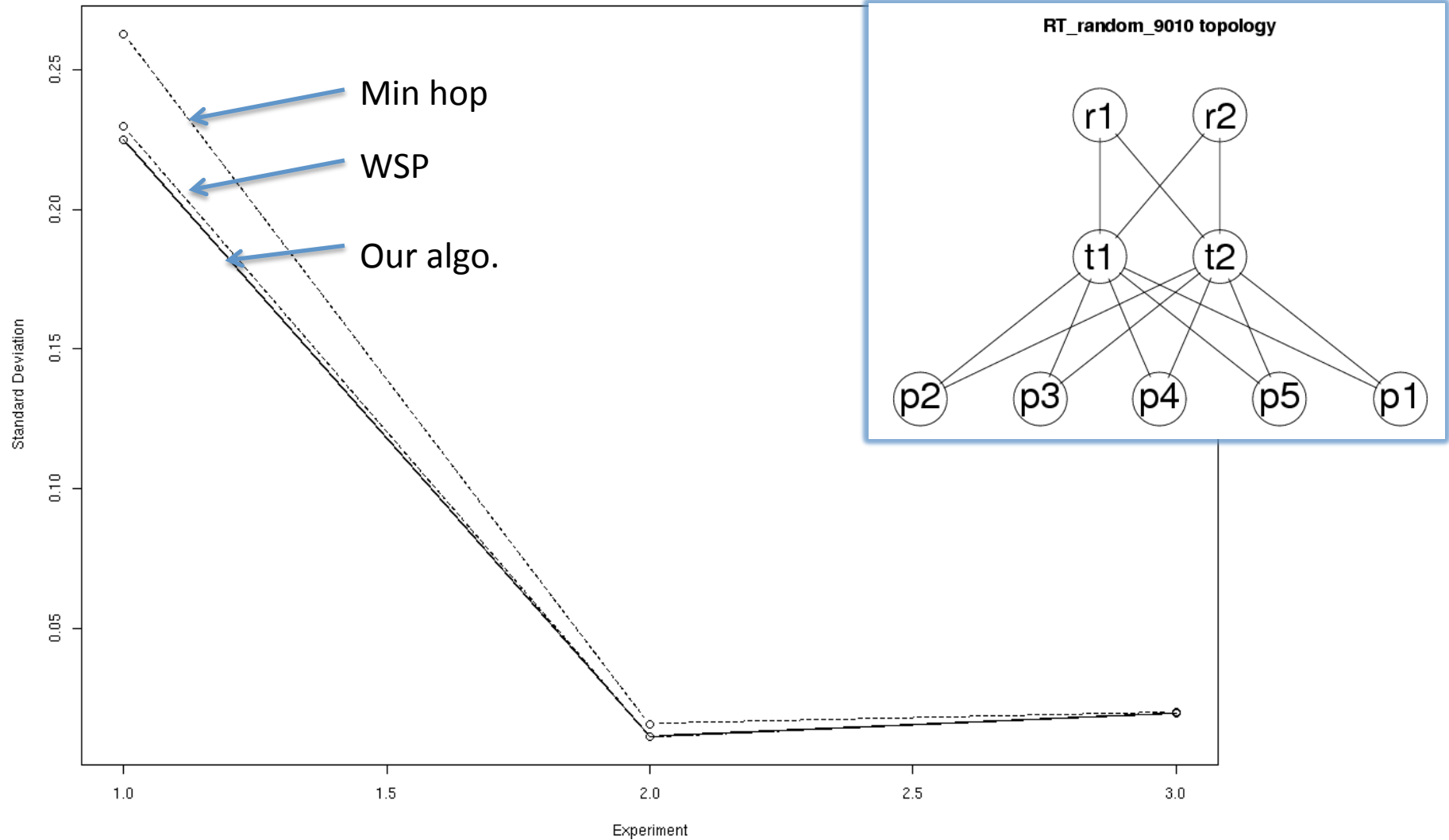


X: 5 experiments
Y: Standard Deviation
10 VM on each PM

Compare random and ordered <S,D> selection



Performance of Algorithms



Future Work

- Apply more topologies and traffic matrix
- Route selection algorithms
 - Pick the route that renders the resulting link expected load the closest to the original
- Tradeoff between computation overhead and load balance degree
 - "intelligently" route X percent of $\langle S, D \rangle$ pairs and achieve 90 percentage of load balance degree

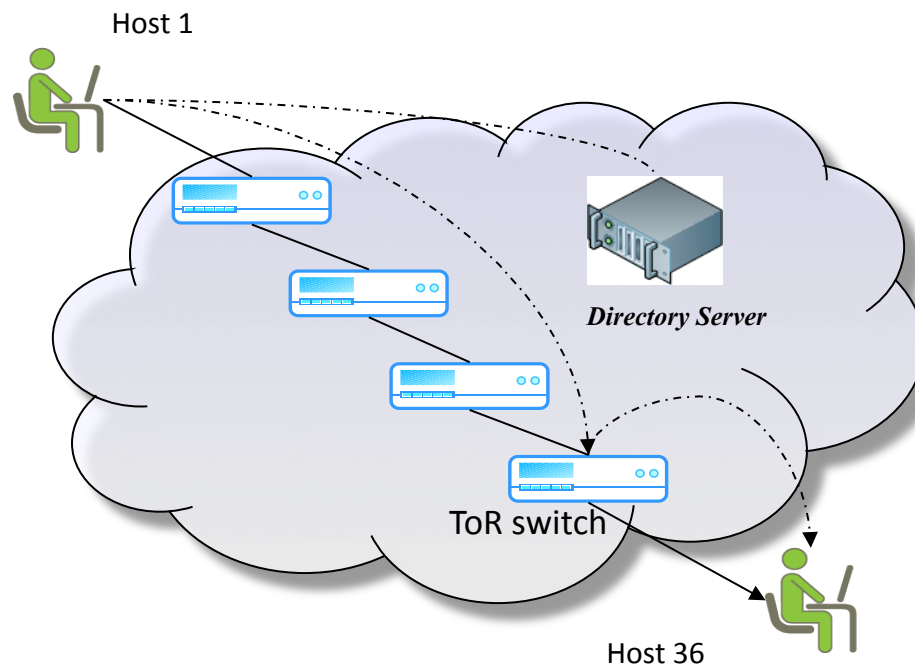
Thank you

END

BACKUP SLIDE

Two-stage Forwarding

- Source → Intermediate → Destination
 - Intermediate: **TOR_Switch(Dest)** or **Physical_Machine (Dest)**
- Every Intermediate knows how to route to every VM in its scope
 - Intermediate needs to be notified when VM leaves or joins its scope
- Directory Server: Host → Intermediate(Host)



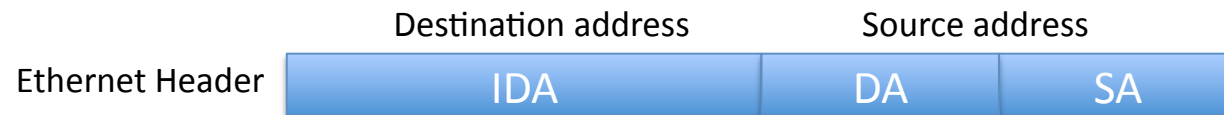
Switch only learns MAC addresses from:

1. TOR switches or
2. Physical Machines

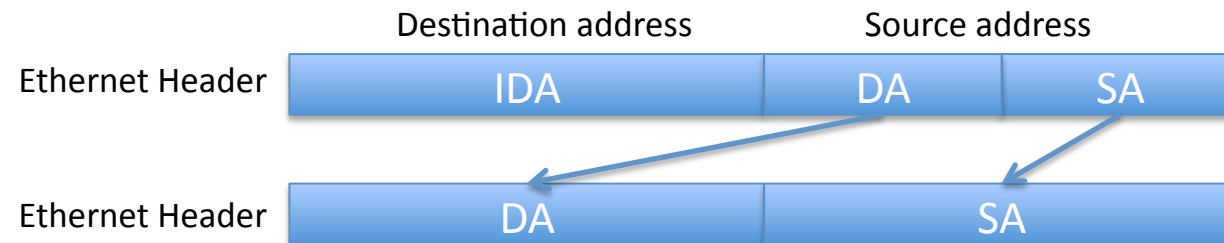
No visibility to VM's MAC!

Encapsulation and Decapsulation

- **Encapsulation:** Place DA and SA in the Ethernet source address field



- **Decapsulation:** extract DA and put it in the Ethernet destination address field



- Each VM's and PM's MAC address is effectively only 3 bytes long (as opposed to 6 bytes long)
- The most significant half of the Ethernet source address field signifies whether it is "encapsulated"

SA: Source address, **DA:** Destination address
IDA: Intermediate Destination address

Direct Routing Optimization

- Idea: When destination VMs are popular, they should be reached via **direct routing**
- Reserve some forwarding table entries for physical machines and switches, and leave the rest for directly routing
- How to allocate forwarding table entries
 - Total destination popularity/load of each VM
 - Load of each <source VM, destination VM> pair
 - Leverage **traffic matrix** information

Related Work

- **Shortest path routing**
 - Select minimum hop paths
 - Unaware of traffic load
 - Some perform ad hoc load balancing and congestion management
- **Hop-by-hop routing schemes (OSPF, BGP)**
 - No explicit control over end-to-end route
- **Valiant-load balancing**
 - Unaware of traffic load

Goals for Data Center Network

Efficient

- Achieve wire-speed
- No oversubscription
- Non-blocking

Manageable

- Ease of configuration
- Ease of management
- Resilient to frequent changes

Reliable

- Failure aware
- Fast recovery

Data Center Network Requirements

- **Host mobility** (VM migration)
 - no IP re-assignment
 - no TCP reconnect
- **Easy administration and configuration**
- **Path efficiency** (No oversubscription)
 - Any end host achieves wire speed
- **No forwarding loops**

802.1ah Mac-in-Mac

- Known as Provider Backbone Bridges (PBB)
- Meant for tunneling and nested VLAN
- Too much control plane set-up
 - Hierarchical VLAN (S-VID, B-VID) ... need to be managed and configured
- Commodity switch rarely supports
- All PBB bridges must support 802.1ah
 - We provide both a hardware (switch) and software(dom0) mac-in-mac solution.

Outline

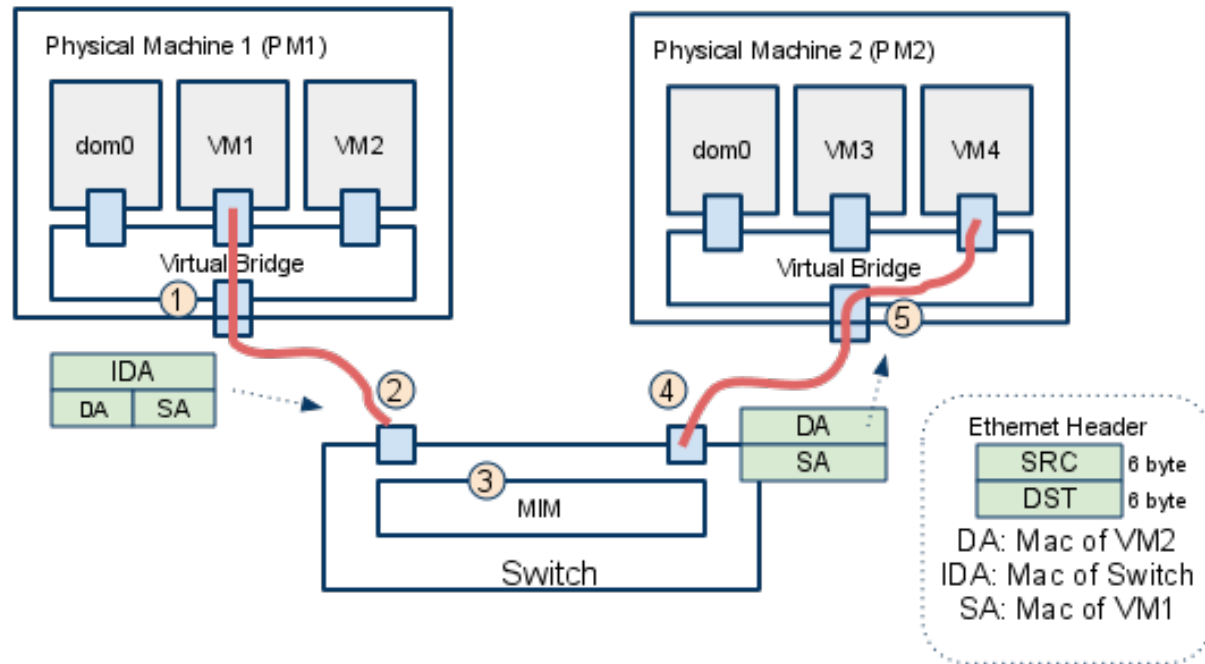
Find where the problem is:

- Ethernet's scalability problems
- L3 + L2 Architecture
- Cost Analysis
- 802.1ah is not a good fit

Solve the problem:

- CCMA L2
 - Topology
 - Two-stage dual-mode forwarding
 - Fault-tolerant routing

Encapsulation and Decapsulation



- (1) Data packets are encapsulated at PM1. (SA=VM1, DA=VM4, IDA=Switch)
- (2) The MIM switch forwards the frame based on IDA.
- (3, 4) The MIM switch detects the mac-in-mac packets and do decap at MIM.
Mac address of **VM4** is now at Ethernet destination header (DA)
- (5) The virtual switch at PM2 forwards this frame to VM4.

Reallocation of Direct Route When VM Migration

- **Problem:** What happen when VM migrates to other physical machine?
 - The forwarding table entries allocated to a destination VM need to be explicitly removed.
- **Solution:**
 - The route server first allocates forwarding table entries for new location.
 - Then the route server removes the invalid forwarding table entries along the old path.

Conclusion

With Two-stage, dual-mode forwarding:

- Scale to 1 million VMs using **commodity switch** with only 16K to 32K entries
- Optimize the switch performance **by direct and indirect** route strategy.

Outline

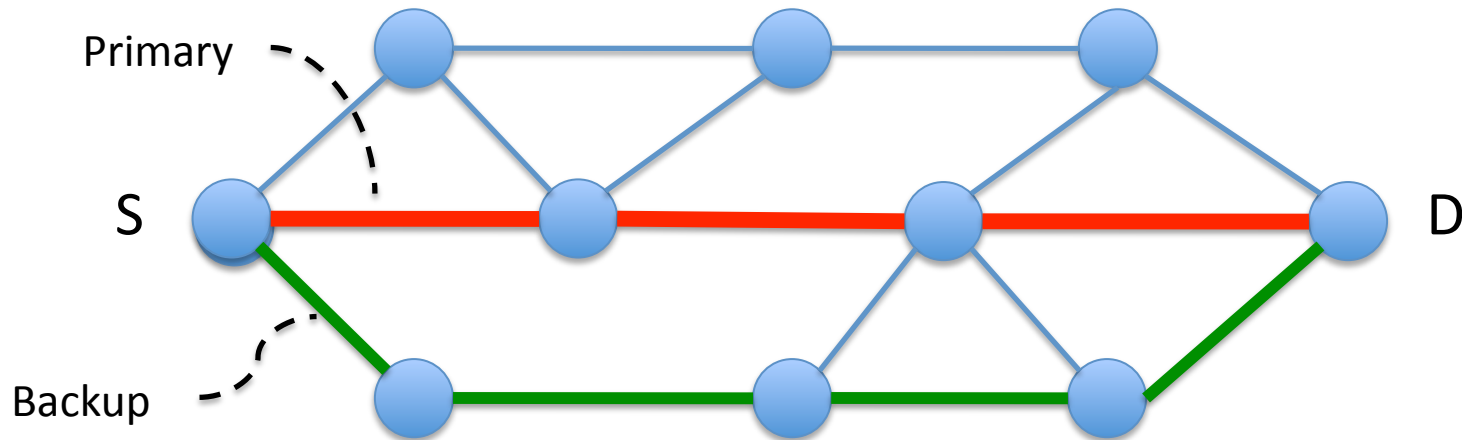
Find where the problem is:

- Ethernet's scalability problems
- L3 + L2 Architecture
- Cost Analysis
- 802.1ah is not a good fit

Solve the problem:

- CCMA L2
 - Topology
 - Two-stage dual-mode forwarding
 - **Dynamic Load Balancing Routing**

Goal



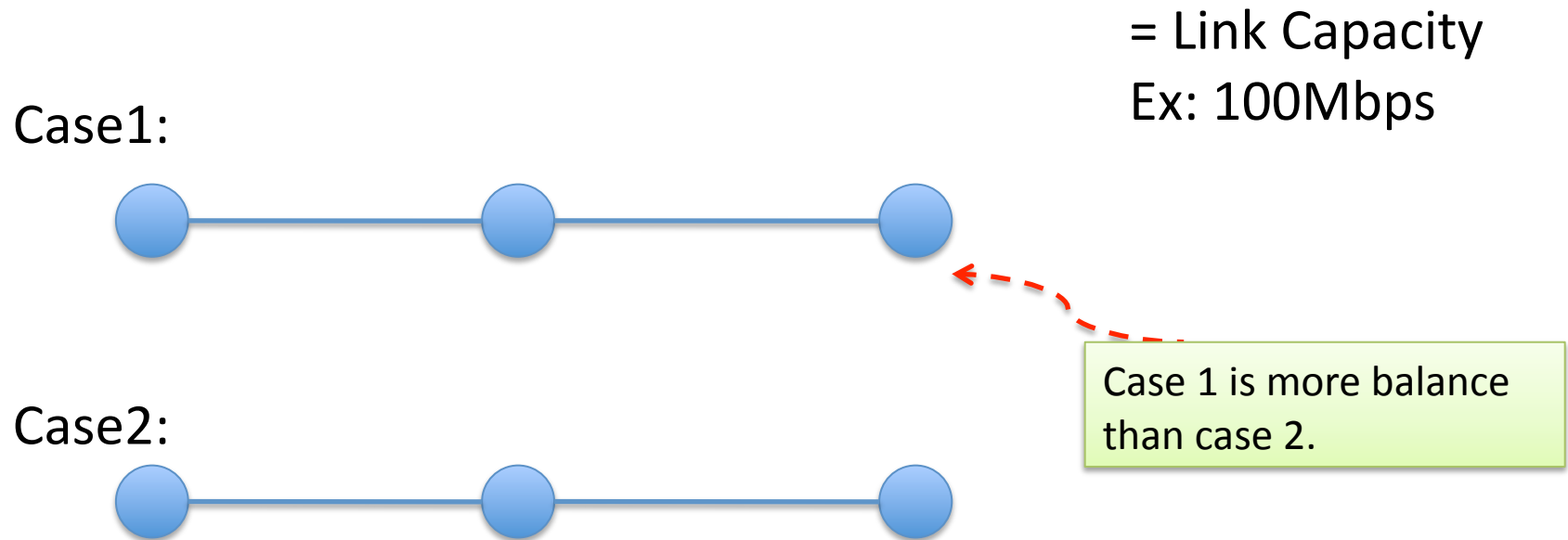
- Given a mesh network and traffic profile
 - Maximize the network resource utilization
 - Prevent congestion by balancing the network load to support as many traffic load as possible
 - Provide fast recovery from failure
 - Provide primary-backup route to minimize recovery time

Related Work

- **Shortest path routing**
 - Select minimum hop paths
 - Unaware of traffic load
 - Some perform ad hoc load balancing and congestion management
- **Hop-by-hop routing schemes (OSPF, BGP)**
 - No explicit control over end-to-end route
- **Valiant-load balancing**
 - Unaware of traffic load

Quantify Load Balance Metric

Network-wide load balancing metric:



Route Selection Algorithm

Given the following information:

- **Traffic matrix**: $N \times N$, where N is the number of physical nodes.
- Take the following metrics:
 - Hop count for each path (M1)
 - Cost of each link (M2)
 - Load balance value (M3)
 - Fwd on each node (M4)
- Find an optimized value that represents best route

Route Selection Process

Step1.

Order all traffic matrix entries in decreasing order.

Step2.

For each $\langle S, D \rangle$ pair in the matrix, find all possible routes between $\langle S, D \rangle$.

Step3.

Pick a pair of **disjoint** routes (primary and backup) from it by calculating the the value v .

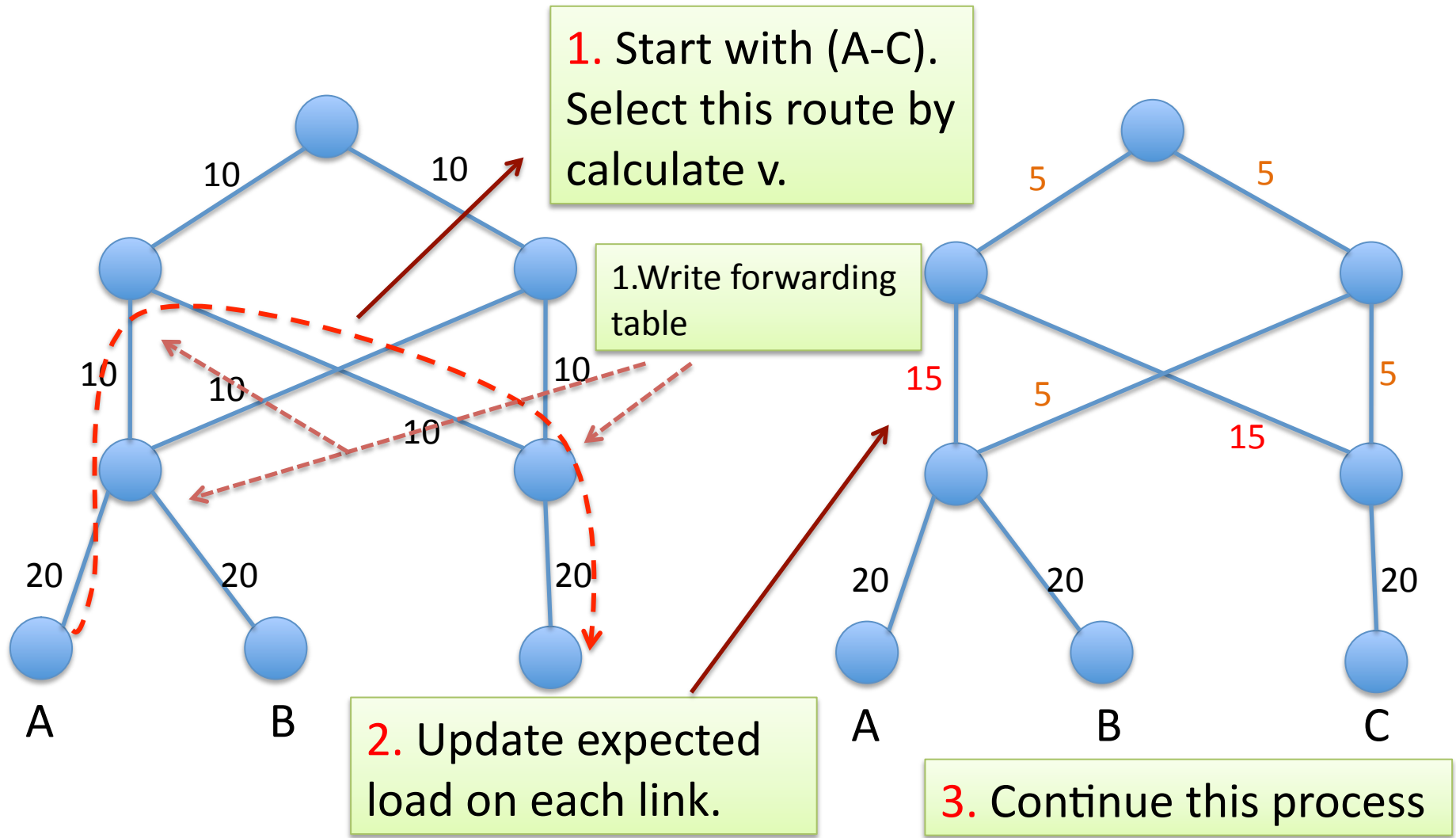
Step4.

Update expected load on each link and forwarding table entry on each node.

Step5.

Continue on the remaining traffic matrix until they are exhausted.

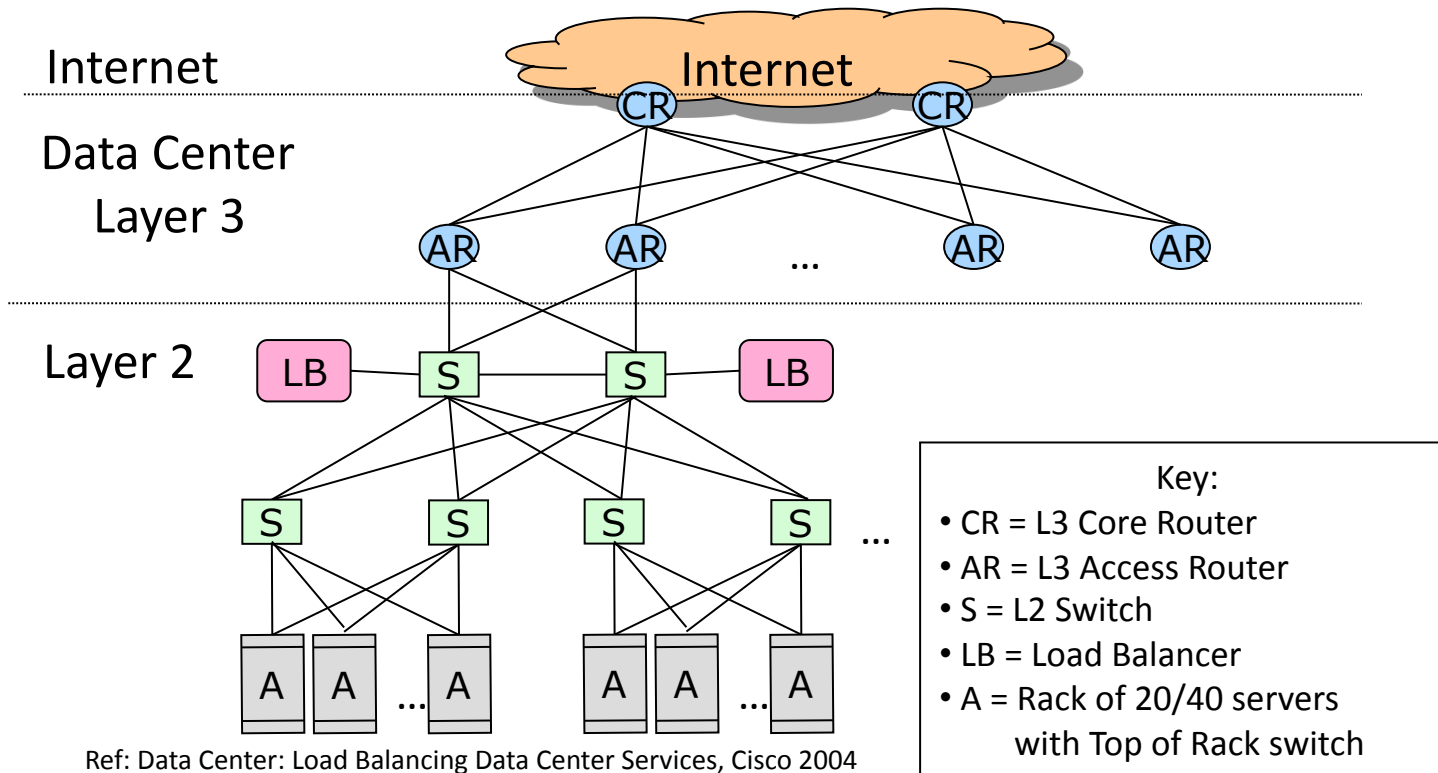
Primary Route Selection Example



Conclusion

- Provide a route selection algorithm to maximize link utilization
- Provide a backup route for fast recovery
- Achieve maximum link utilization by load balancing the requests
- Define a quantitative metric based on
 - Hop count
 - Criticality
 - Load balance
 - Forwarding table size

L2 + L3 Architecture

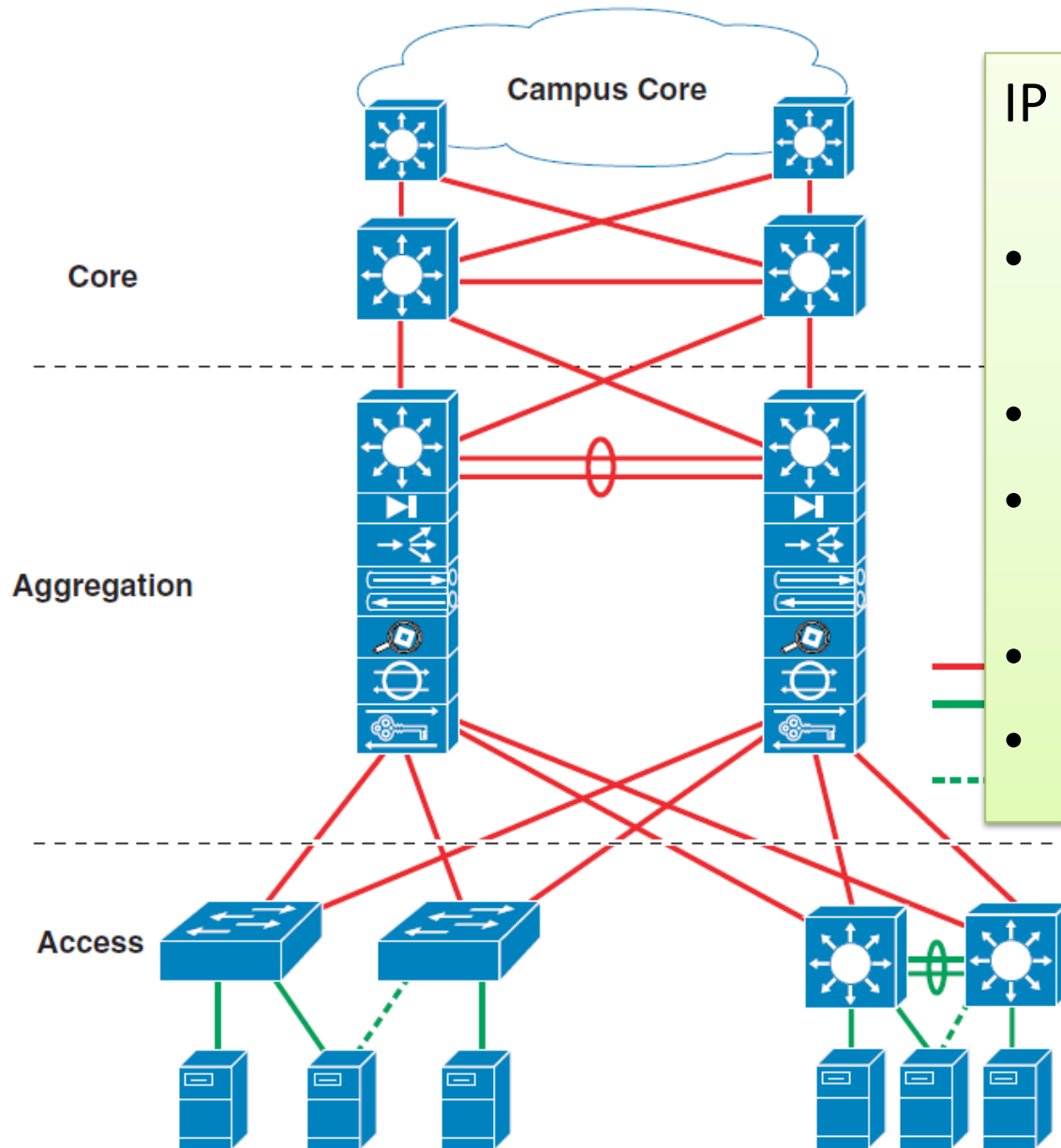


1. Hierarchical network; 1+1 redundancy
2. Equipment higher in the hierarchy cost more and more efforts made at availability

Dynamic Load Balancing Routing Algorithm for Data Center Network

Cheng-Chun Tu

L2 + L3 Architecture



IP network provide scalability

but...

- VMs can't migrate to different subnets
- Routers need to configure
- Address assignment to be managed
- Single point of failure
- Scale up not scale out

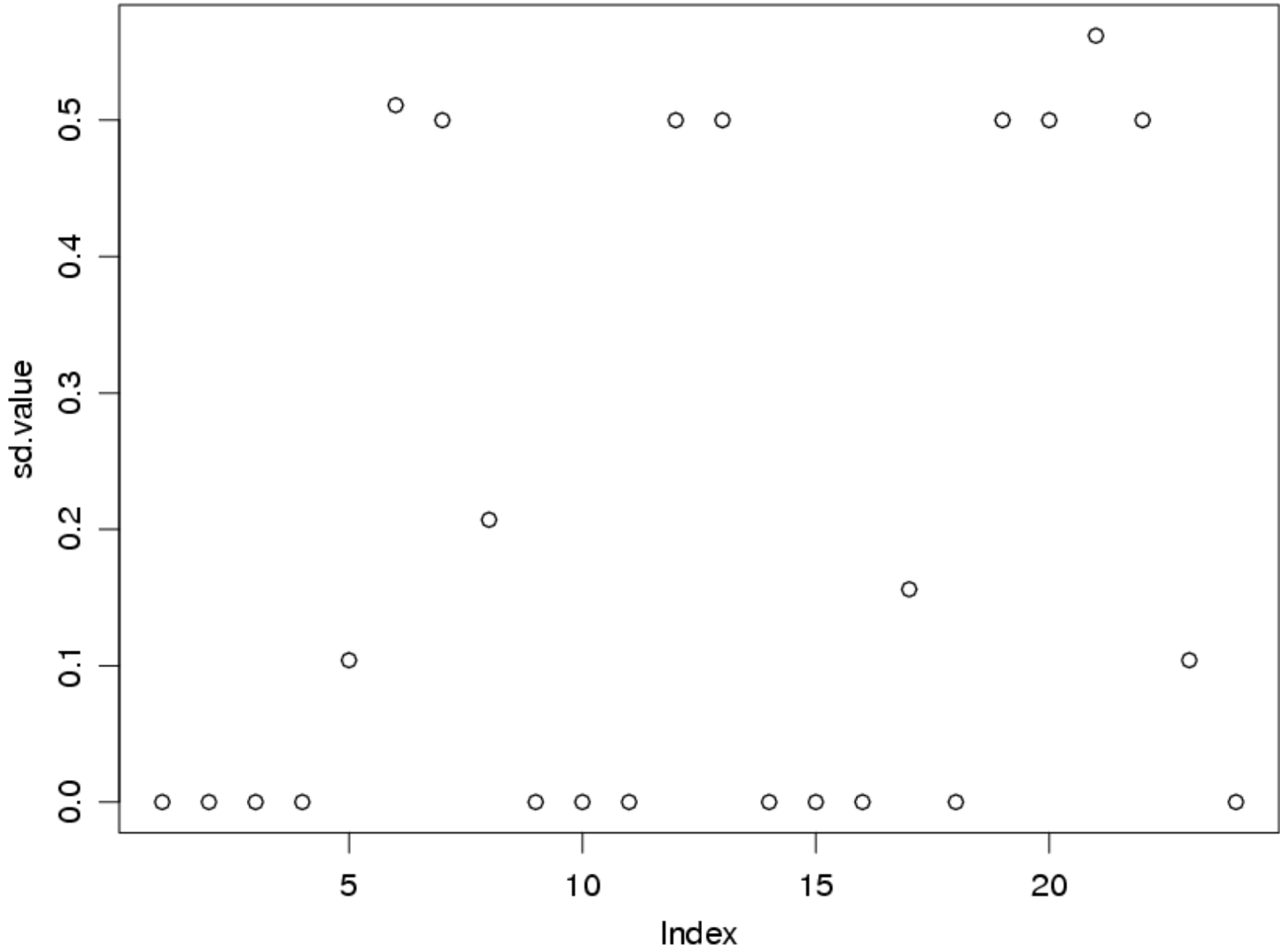
From Cisco's Data Center Architecture 2.5

143440

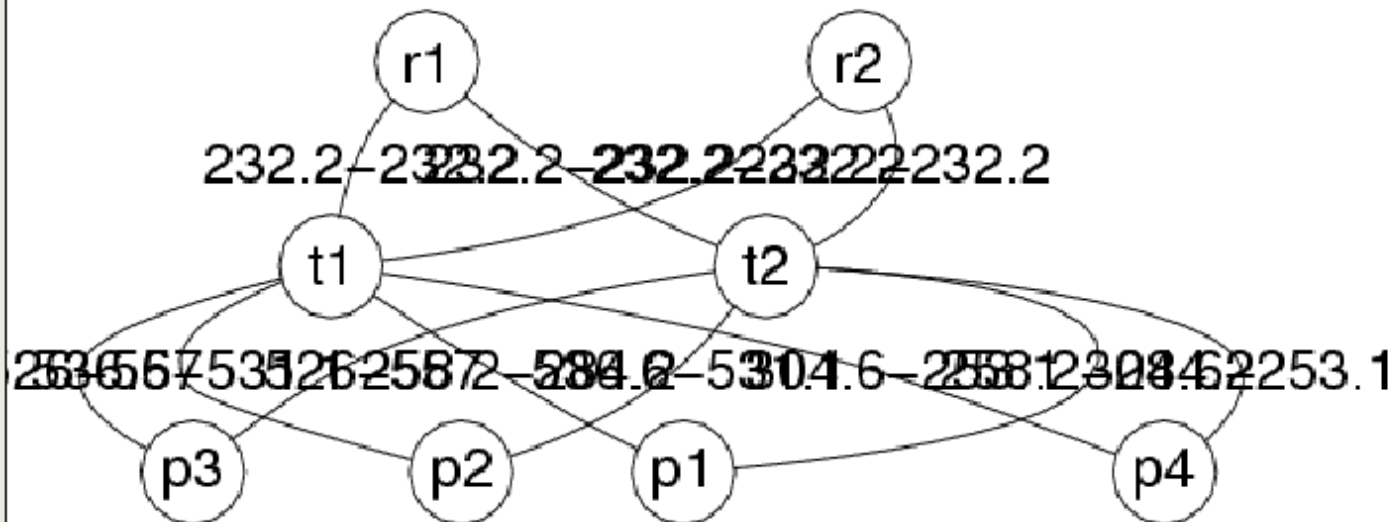
TODO List

- Routing adds node fails
- The network is flat

Link Costs



Expected Load



Residual Capacity

