CS 755 – System and Network Architectures and Implementation

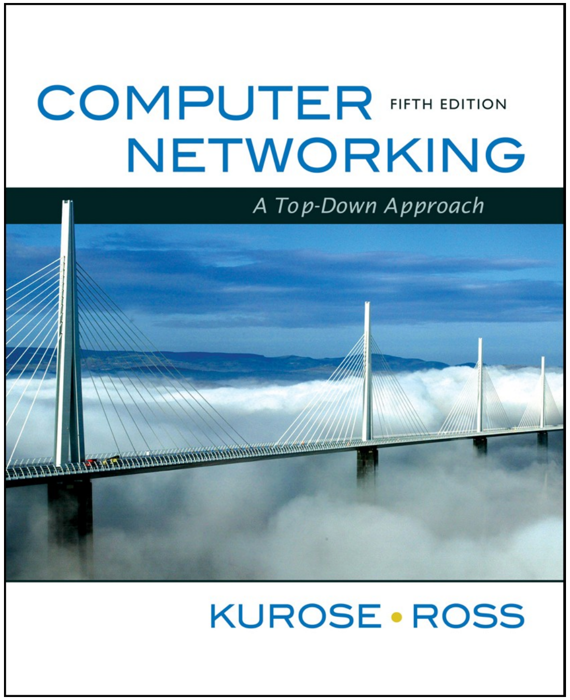
Module 2 - Networks

Martin Karsten

mkarsten@uwaterloo.ca

Notice

Some slides and elements of slides are taken from third-party slide sets. In this module, parts are taken from the Kurose/Ross slide set. See detailed statement on next slide...



A note on the use of these ppt slides:

Computer Networking: A Top Down Approach
5th edition.
Jim Kurose, Keith Ross
Addison-Wesley, April 2009.

Overview

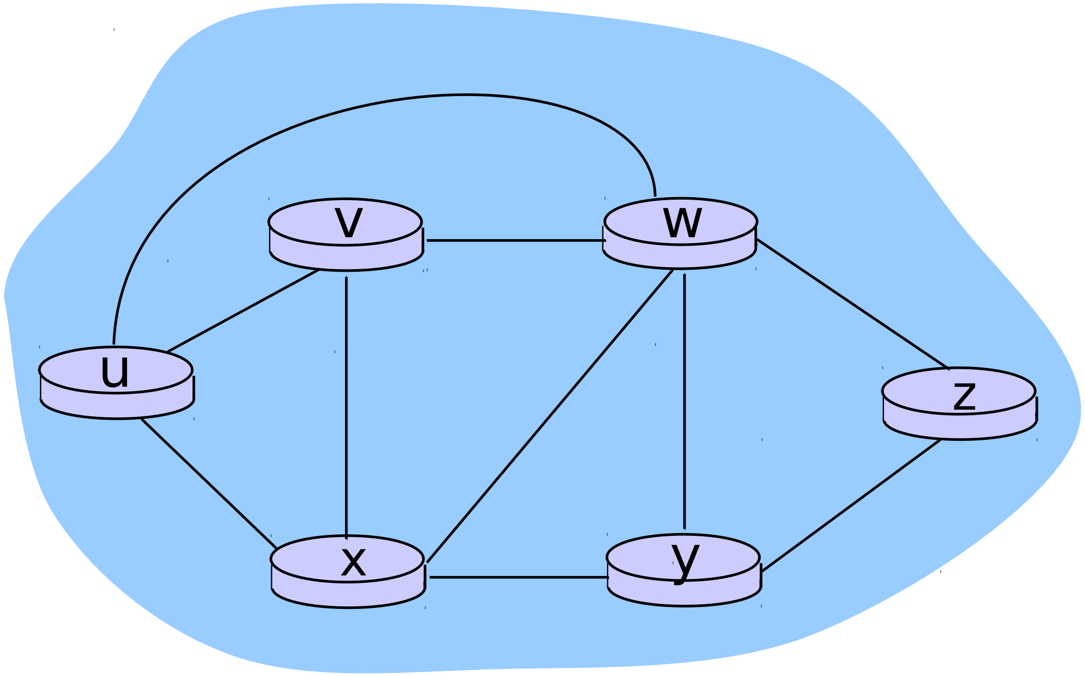
- graph abstraction
- routing and forwarding
- scalability: hierarchy and aggregation
- virtualization

Channels – Review

- multiple stations share channel
- assumption: every station can reach all others
    - not entirely true for radio channels...
- main concerns
    - transmission of meaningful units, error control
    - medium access control
- labelling? yes, for filtering (not reachability)
    - add sender/receiver labels to message
        - not strictly needed for point-to-point links

Network

- consider network as partially connected graph



- no direct reachibility between all pairs of nodes

Node Labelling

- assign global label to each node – *address*
- compare with postal address
  - hierarchical
    - uniqueness
  - location-dependent
    - implicitly hierarchical

- network address – may or may not be hierarchical or location-dependent

Interface Labelling

- assign label to each interface at each node
  - global vs. local ('eth0')
- with node labelling
  - need/want at least neighbour-to-interface mapping
  - e.g. at Node U (#Slide 6)

    V -> eth0

    W -> eth1

    X -> eth2

Terminology

- communication session
    - *unicast* (1-to-1) – in focus here
    - *multicast* (1-to-many)
    - *broadcast* (1-to-all) – what does 'all' mean? (*scope*)
- end system: *host*
    - *sender* or *source*
    - *receiver* or *destination*

Terminology (cont'd)

- intermediate system: *router*
  - vs. *hub* vs. *switch* – details later
- *routing*
  - dissemination of topology information
  - *path computation*
- *forwarding*
  - *path selection*
  - move messages from input link to output link

Return Path Announcement

- assume forward path exists and is used
- assume symmetric return path
- record return path
  - in message
  - in routers (vs. switch)

- assume previously shown graph (#Slide 6)

Source Routing

- message from U to Z travels via
  - U/eth2 -> X/eth0
  - X/eth3 -> W/eth0
  - W/eth1-> Z/eth1
- record eth0, eth0, eth1 -> can reverse and use!

Source Routing (alt version)

- assume globally unique names
- message from U to Z travels via X and W
  - record path in message: U, X, W, Z
  - reverse path at receiver: Z, W, X, U
- use reverse path to reach U from Z
- use local neighbour table to find interface

Self Learning

- message from U to Z travels via X and W
- assume neighbour table, then
  - record at W: U -> X
  - record at Z: U -> W
- can send message to Z 'directly'
  - without including path
  - at each router: look up table entries

Bootstrapping

- first message announces reachability

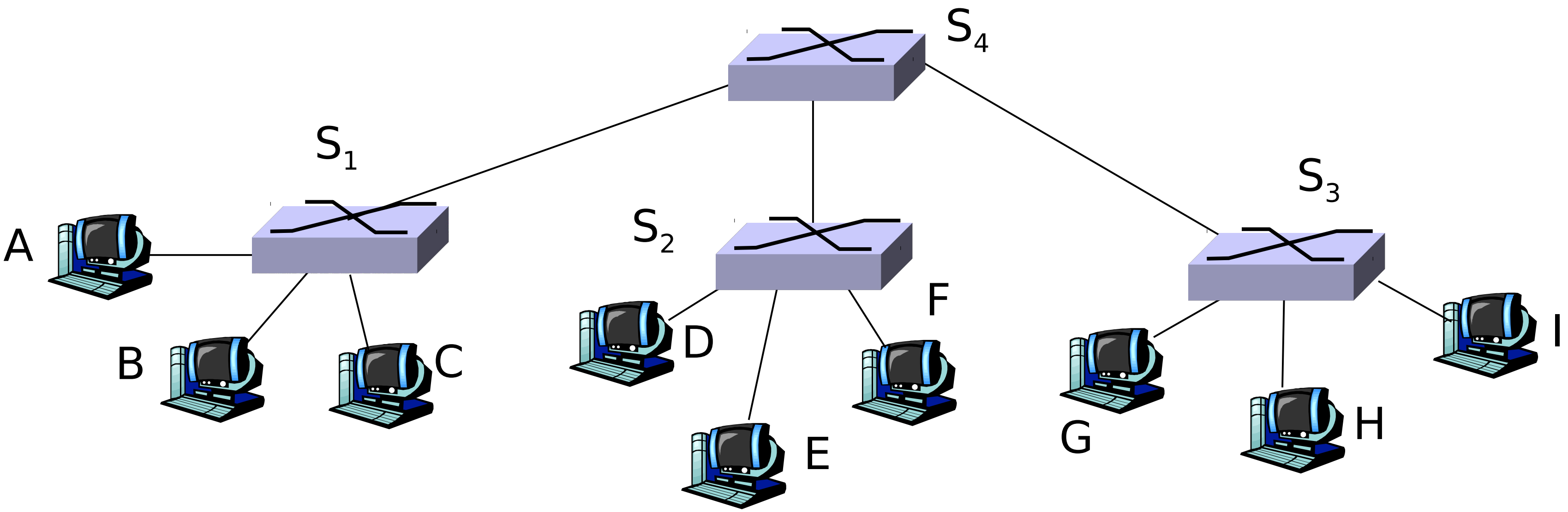
- transmission of first message?
  - broadcast – e.g., Ethernet switching
  - unicast – using some other method

Switched Ethernet

- globally unique MAC addresses
  - admin hierarchy through IEEE
- switch records information from arriving frame
  - store source address -> interface in switch table
- switch looks up destination address
  - found -> forward via interface
  - not found -> broadcast to all interfaces

Ethernet – Hierarchical Topology

- works just fine



- self-learning algorithm adapts automatically
- but: broadcast overhead?

Ethernet – History

- initial version: bus/cable
  - signal transmission limitations
  - cabling structure? cable break?
- next version: star topology
  - repeater – extend signal reach
  - hub (multiple interfaces) – permit structured cabling
- current version: switched
  - reduce broadcast effects / isolate collision domains
  - intelligence: self-learning & buffering

Virtual Circuit

- similar to self-learning:
  return path announcement
- use local labels, instead of addresses (#Slide 6)
  - at U: store a -> application, announce U/a
  - at X: store b -> U/a, announce X/b
  - at W: store c -> X/b, announce W/c
  - at Z: return label is W/c
- need neighbour tables (or use interface labels)
  - forwarding: replace label and forward message

Virtual Circuit

- rationale
  - can set up circuit per session (management)
  - number of sessions << number of end systems
  - use (and reuse) limited range of local labels

  => compact table, fast lookup
  - array vs. tree
- *but*: circuit management overhead

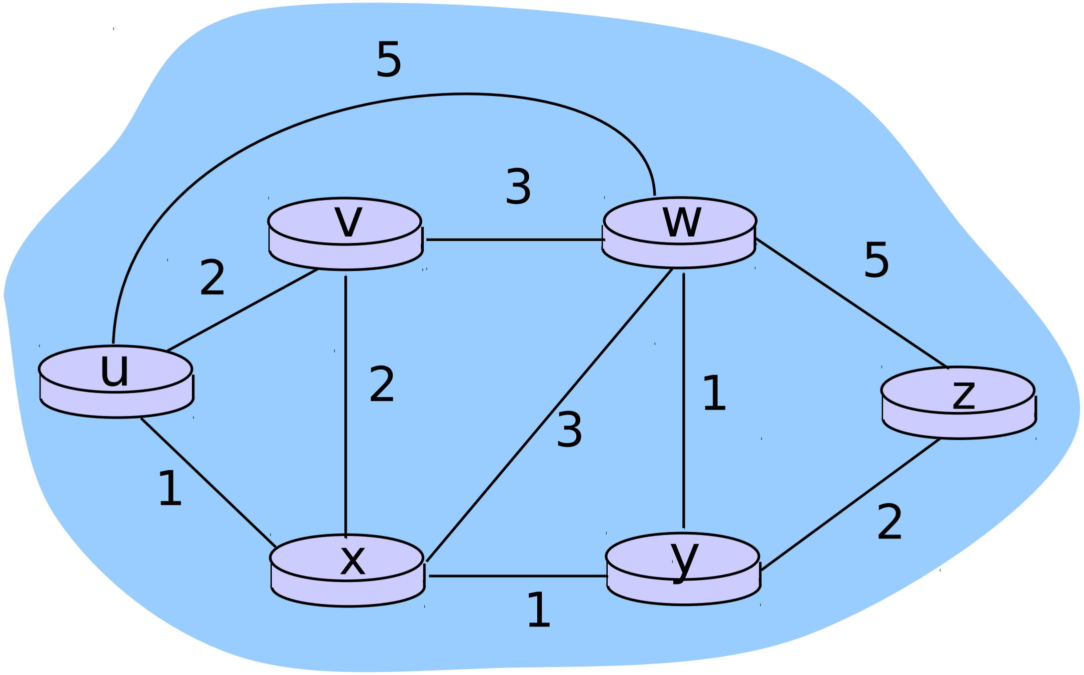
- home exercise – verify: NAT uses VC principles

Routing

- asynchronous topology discovery
  - decoupled from message transfer
- goals
  - discover available paths and characteristics
  - choose between paths
    - lowest cost, best service
    - get rid of packet asap
    - do not send via provider X
    - but also: maintain system consistency and stability

Graph with Link Costs

- cost: money, delay, load, etc.
- algorithms: cost must be positive and additive



Dijkstra's Algorithm

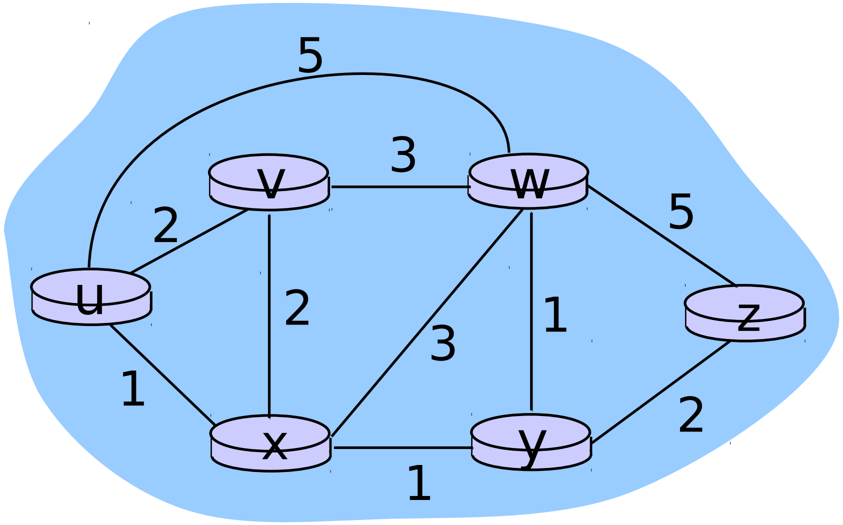
- global information: cost of all links in network
- Notation (at one node)
    - c(x,y): link cost from node x to y
    - D(v): current of cost of path to v
    - p(v): last predecessor on path to v
    - N': set of nodes whose least cost path is known
- iterative algorithm:
  after k iterations, algorithm has computed k least-cost paths to k nearest destinations

Dijkstra's Algorithm

```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
      then $D(v) = c(u,v)$
    else $D(v) = \infty$
Loop
  find w not in N' such that D(w) is a minimum
  add w to N'
  update D(v) for all v adjacent to w and not in N' :
      $D(v) = \min( D(v), D(w) + c(w,v) )$
  /* new cost to v is either old cost to v or known
   shortest path cost to w plus cost from w to v */
until all nodes in N'
```

Dijkstra's Algorithm – Example

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Link State Routing

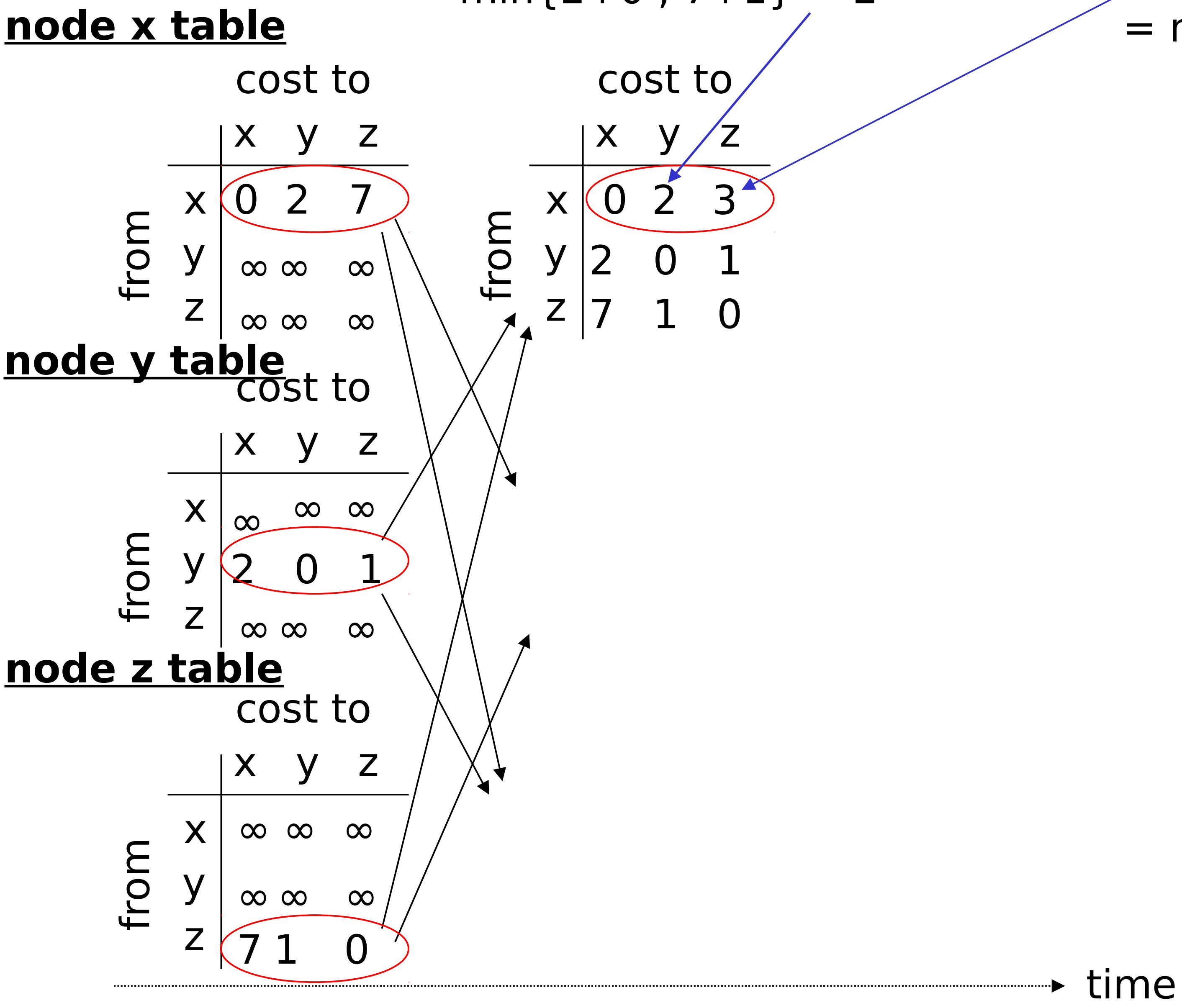
- routing protocols, e.g., OSPF
- establish scope, then
  disseminate link information “globally”
- update periodically and when link changes
- run Dijkstra's algorithm at each router
  - convergence phase during updates
- O(n^2) runtime, broadcast updates, scalability?

Distance Vector Algorithm

- local information:
  - cost of links to all neighbours
  - neighbours' current costs to all known destinations
- Notation
  - c(x,y): link cost from node x to y
  - d(x,y): cost of known least-cost path from x to y
- Then: $d(x,y) = \min_v\{ c(x,v) + d(v,y) \}$
  - repeated iterative application converges to least-cost of paths and known next hop (Bellman-Ford)

$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$$
$$= \min\{2+0 , 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$$
$$= \min\{2+1 , 7+0\} = 3$$



node x table

cost to

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

cost to

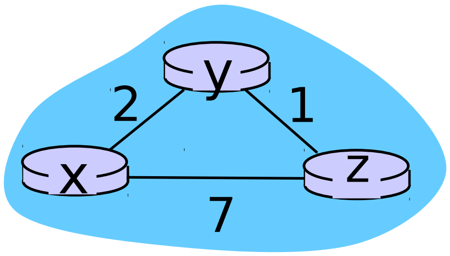
from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

cost to

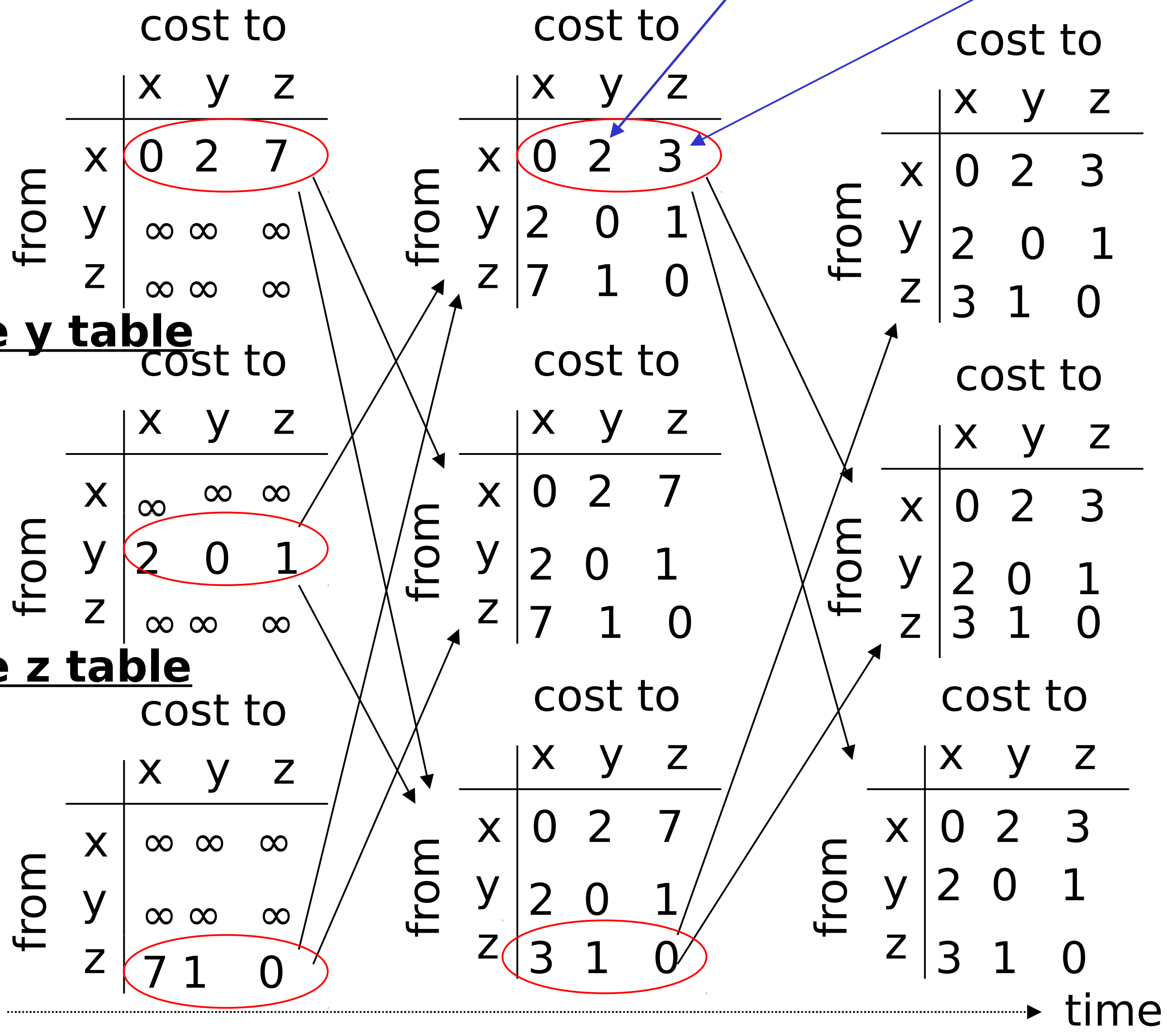
from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

time



$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$$
$$= \min\{2+0 , 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$$
$$= \min\{2+1 , 7+0\} = 3$$



node x table

		cost to		
		x	y	z
from	x	0	2	7
	y	∞	∞	∞
	z	∞	∞	∞

		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	7	1	0

		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	3	1	0

node y table

		cost to		
		x	y	z
from	x	∞	∞	∞
	y	2	0	1
	z	∞	∞	∞

		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	7	1	0

		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	3	1	0

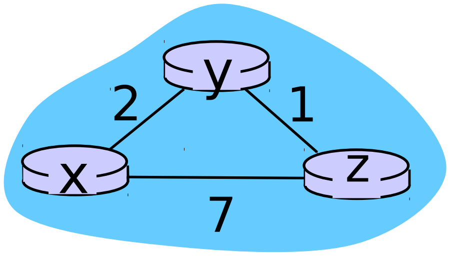
node z table

		cost to		
		x	y	z
from	x	∞	∞	∞
	y	∞	∞	∞
	z	7	1	0

		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	3	1	0

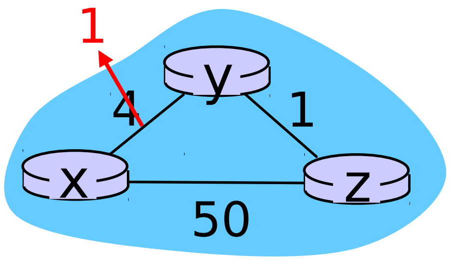
		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	3	1	0

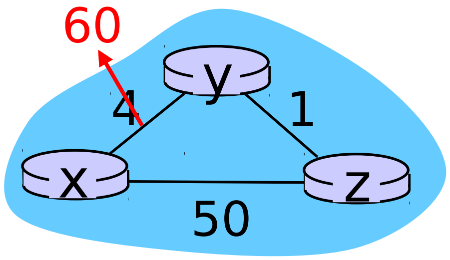
time



Distance Vector – Challenges

- link cost changes
  - node detects local change
  - updates local table
  - if necessary, send updates
- “good news travels fast”





- “bad news travels slow”
- “count to infinity” problem

Distance Vector – Options

- convergence time during updates might be long
  - transient routing loops are problematic
- approaches
  - poisoned reverse: don't send route to next hop
    - only avoids small 3-hop loops
  - path vector: keep and transmit full path
    - avoids loops, but overhead and transparency?
  - synchronous updates -> see literature

Distance Vector Routing

- routing protocols, e.g., RIP, BGP
- disseminate local routing table to neighbours
- update periodically and when table changes
- update local table at each router
  - convergence phase during updates
- O(n) runtime, local updates
- potentially slow convergence, transient loops

Characteristics for Comparison

- message overhead
  - message number vs. transmission scope
- computational overhead
  - vs. frequency of updates
- robustness
  - impact of failures
- policy support
  - transparency might be a good or a bad thing

Other Aspects

- adjust routing dynamically to load changes?
  - might be unstable
- policy routing, BGP local preference
  - might result in inconsistent routing
- route information called *advertisement*
  - advertise reachability via gateway
  - somewhat similar to return path announcement

Scalability

- destination-based routing and forwarding vs. billions of nodes?

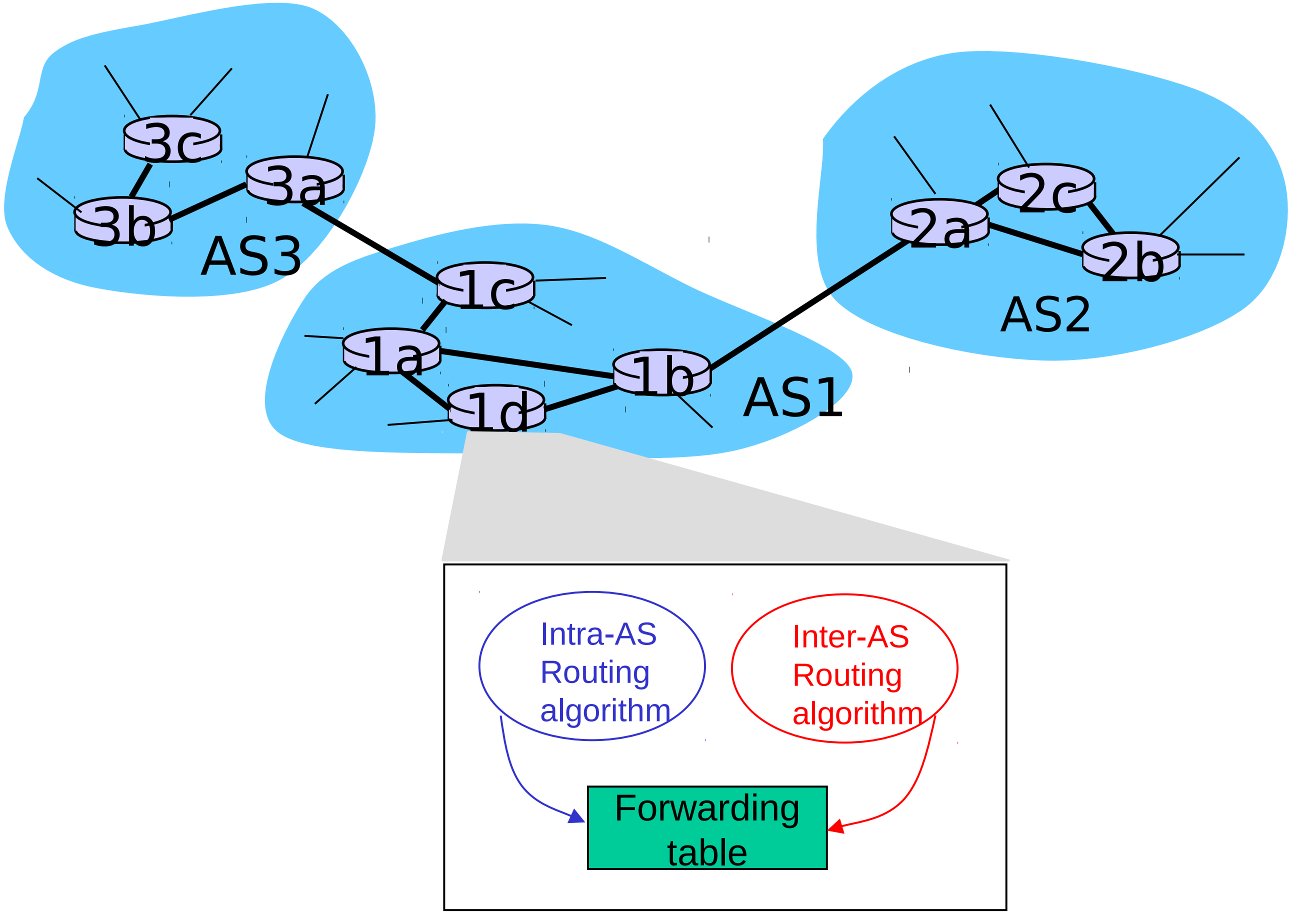
  => hierarchical addressing and routing

  - administrative autonomy for networks
  - business relationships between networks
- Internet = network of networks

- terminology: *autonomous system* (AS)
  - network – administrative unit

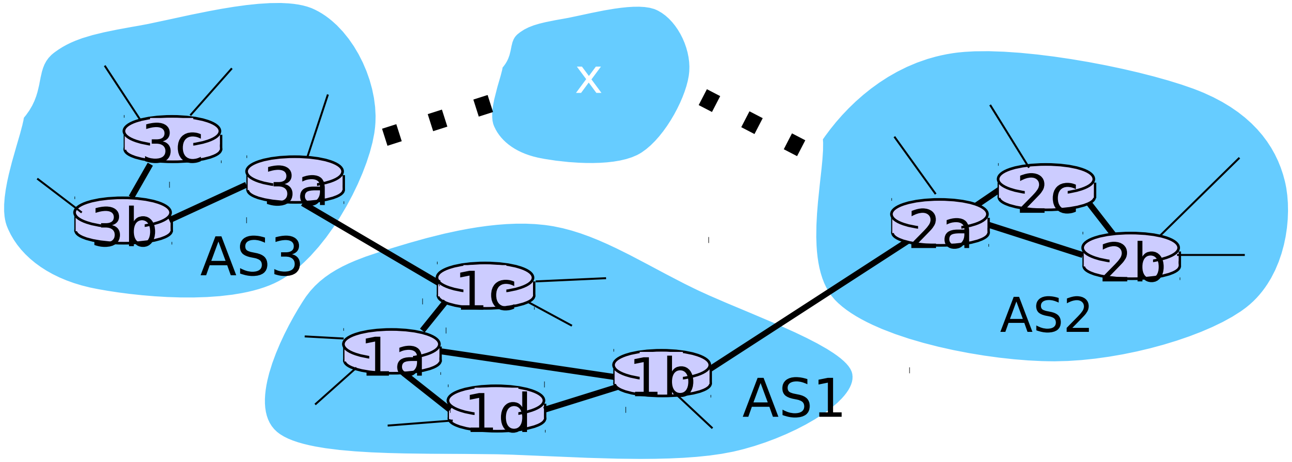
Hierarchical Routing

- interconnected ASes



Hierarchical Routing

- suppose X reachable from AS1 via AS2 or AS3
- configure forwarding table in router 1d
  - inter-domain routing
  - local (cost between routers) vs. global (cost between AS)es concerns?

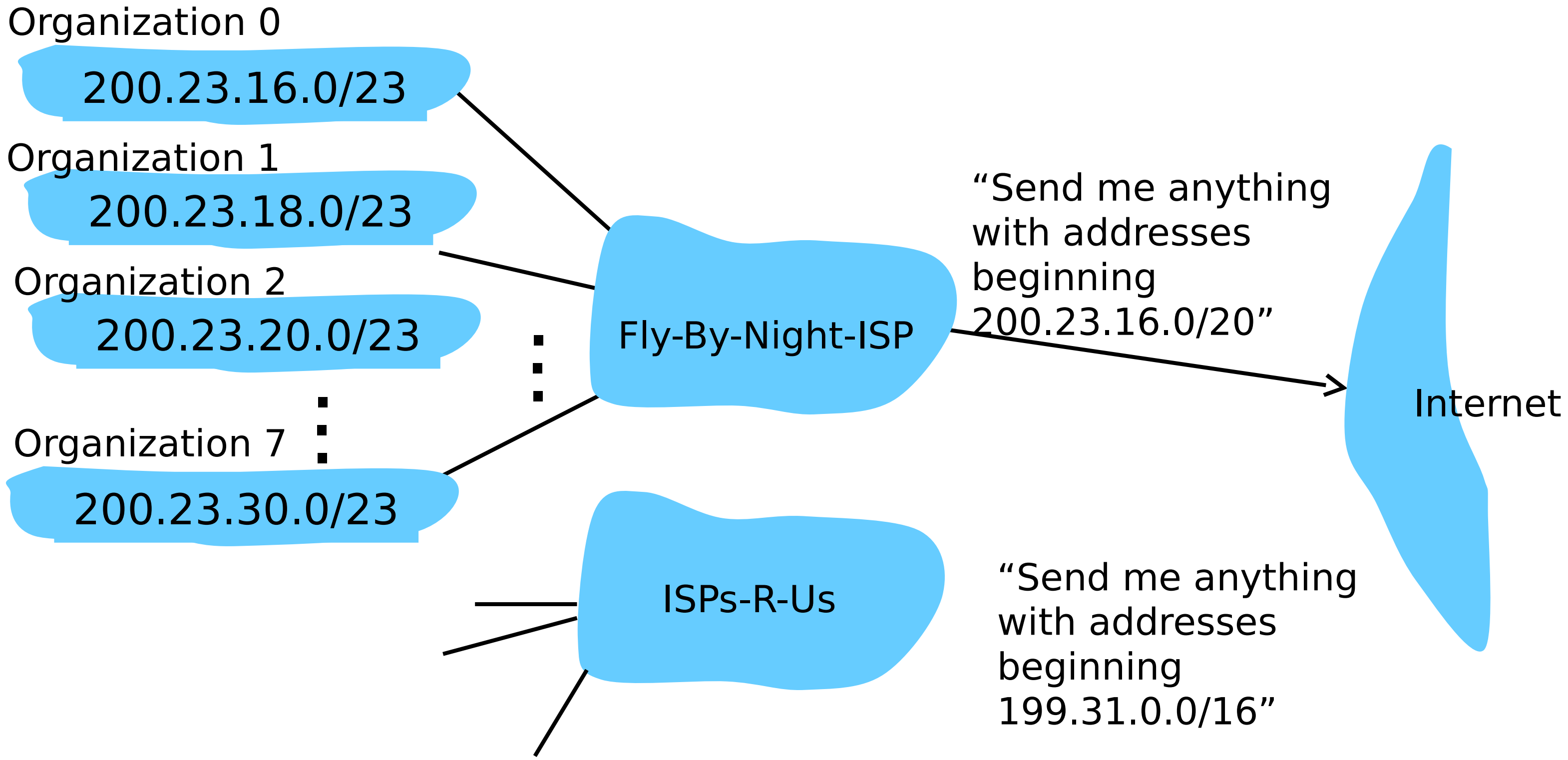


Hierarchical Addressing

- assign contiguous addresses to subnets
  - identified by address *prefix*
- portion of provider's address space
- provider advertises aggregated prefix

ISP's block	11001000 00010111 00010000	00000000	200.23.16.0/20
Organization 0	11001000 00010111 00010000	00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010	00000000	200.23.18.0/23
Organization 2	11001000 00010111 00010100	00000000	200.23.20.0/23
...	.....	....	....
Organization 7	11001000 00010111 00011110	00000000	200.23.30.0/23

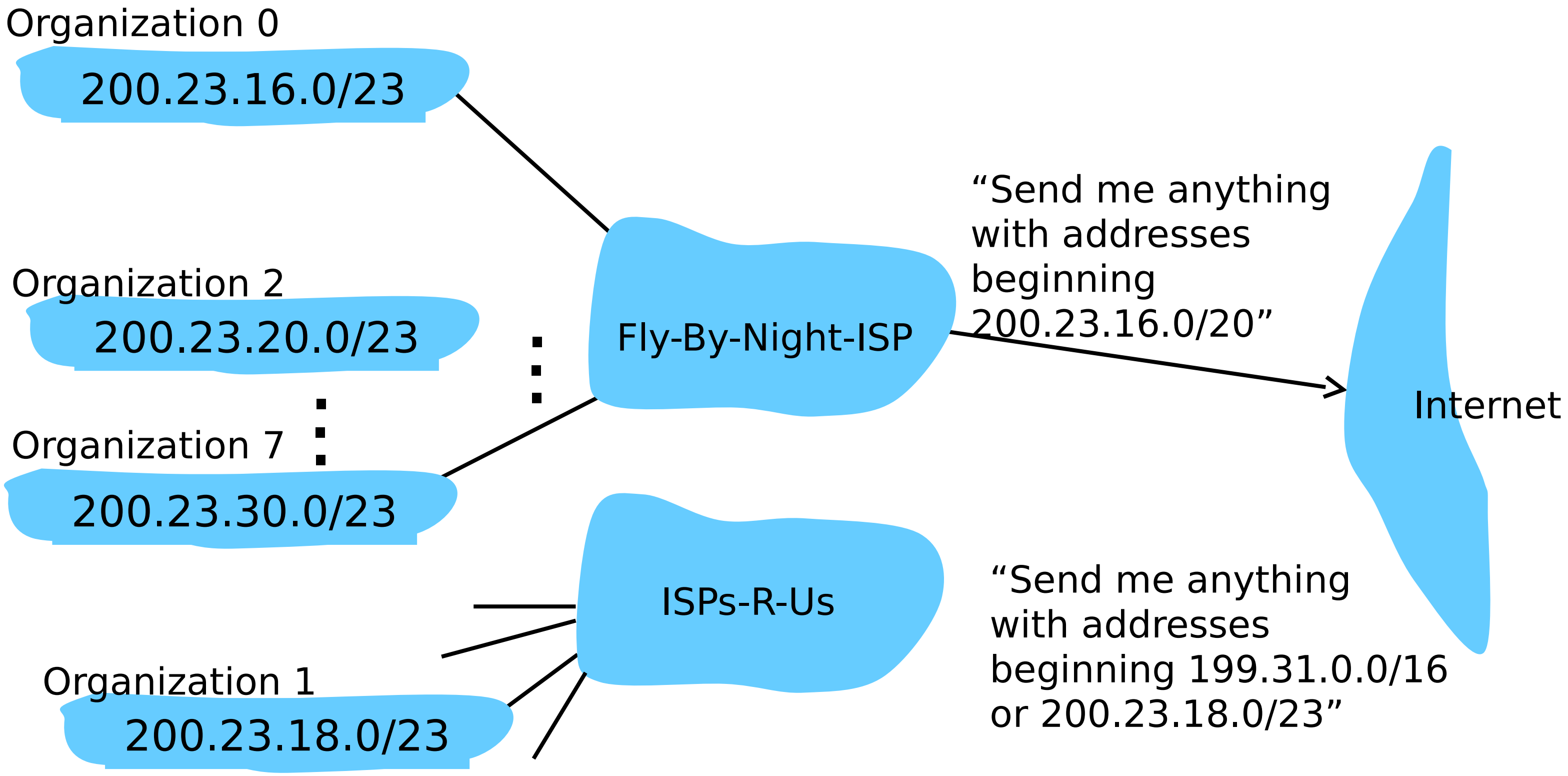
Hierarchical Addressing



- fundamentally: tree vs. graph

Hierarchical Addressing

- deaggregation when network moves
- also: multi-homing



"Send me anything with addresses beginning 200.23.16.0/20"

"Send me anything with addresses beginning 199.31.0.0/16 or 200.23.18.0/23"

Location and Addressing

- location-independent MAC addresses
  - hard-coded in firmware, globally unique
  - Ethernet self-learning algorithm: plug-and-play
  - scaling limitations
- topological IP addresses
  - configured, must be globally unique for responders
    - otherwise NAT is an option
  - scalable, but: network is more densely connected
  - use graph features (redundancy) -> deaggregation

Initiator vs. Responder

- who needs globally routable address?
  - initiator: party to initiate conversation
  - responder: party that accepts conversations
- only responders need globally routable address
  - e.g., initiators work well begin NAT
  - service directory (e.g. VoIP)
    => maintain initiator role for responder functionality
  - service directory itself is responder
- is your laptop a responder?

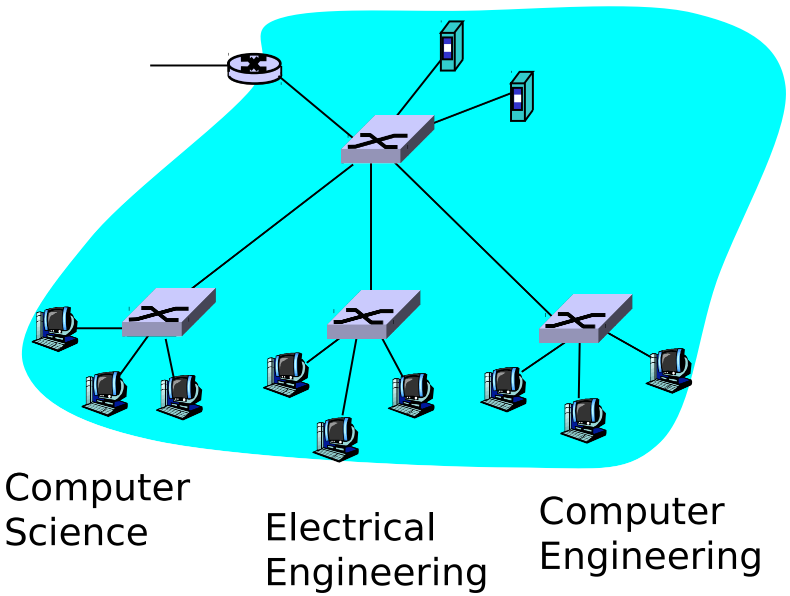
Other Protocols

- Address Resolution Protocol (ARP)
  - request MAC address using broadcast
  - “who knows 10.2.57.10?” -> that node responds
  - broadcast overlaps nicely with Eth self-learning
- Dynamic Host Configuration Protocol (DHCP)
  - server manages pool of IP addresses
  - station asks for IP address during bootstrap
  - MAC broadcast request -> server responds
  - broadcast response -> coordinate multiple servers

Virtualization

- build virtual network graphs on top of networks
- use encapsulation and layering
- examples
    - IP over Ethernet
    - Virtual LANs
    - IP over IP
    - etc...

Virtual LAN (VLAN)

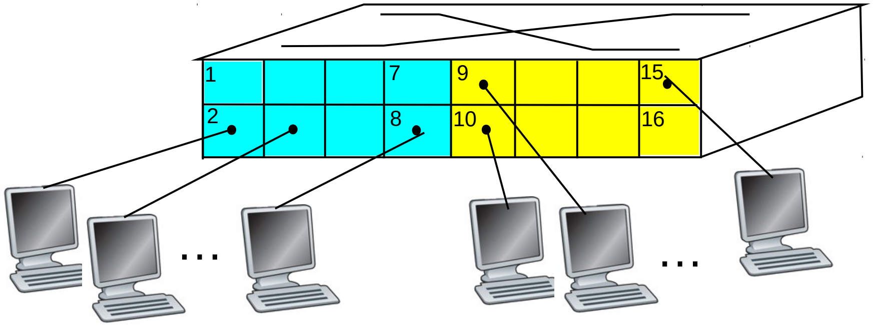


- what if CS user moves office to EE floor?
- single broadcast domain (ARP, DHCP) – security/privacy?
- switches not well utilized

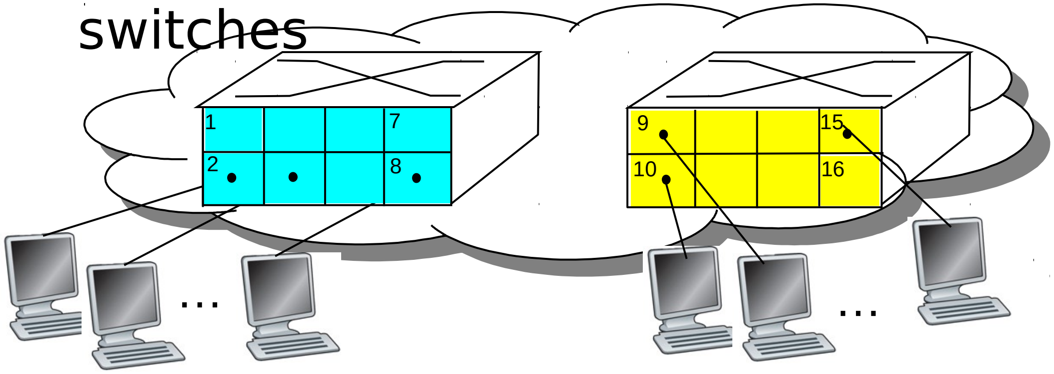
VLAN

- switch can be configured to define multiple <u>virtual</u> LANs over single physical infrastructure

Port-based VLAN: switch ports grouped (by switch management software) so that *single* physical switch ……

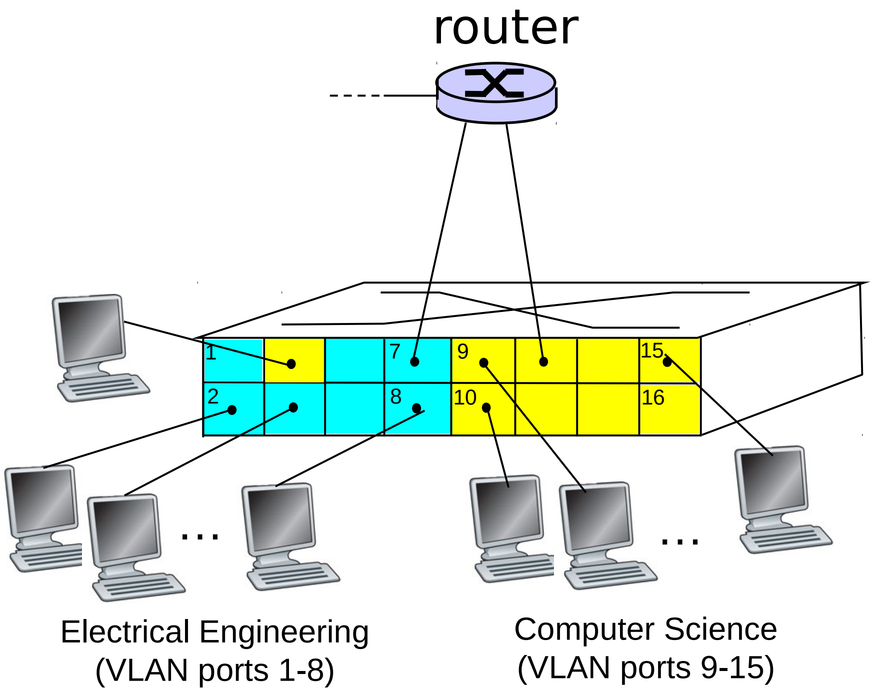


... operates as *multiple* virtual switches

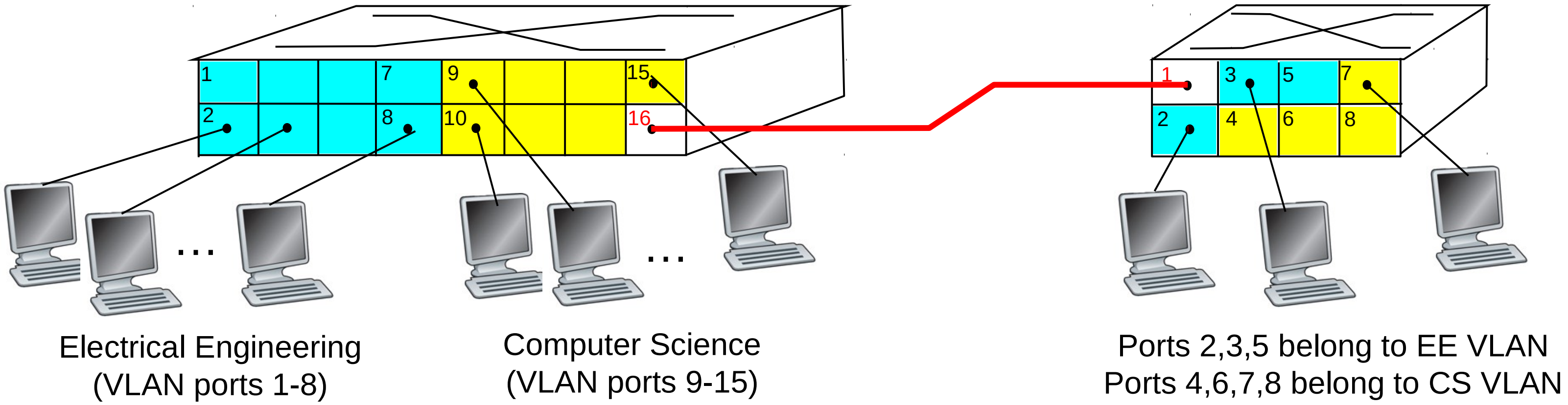


Port-based VLAN

- traffic isolation: broadcast restricted to VLAN
- membership: based on port or MAC address
- forwarding between VLANs: routing



Multi-Switch VLAN



- trunk port: connect switches
- frames forwarded between switches must carry VLAN identifies -> extended protocol format
- IEEE 802.1q defines extra header fields

IP over IP

- example: IPV6 over IPv4
- take arbitrary subset of connected IPv4 nodes
- add IPv6 capability to those nodes
- treat IPv4 as virtual links between IPv6 nodes

  => virtual network

- IP was designed to form overlay network

IP Tunneling