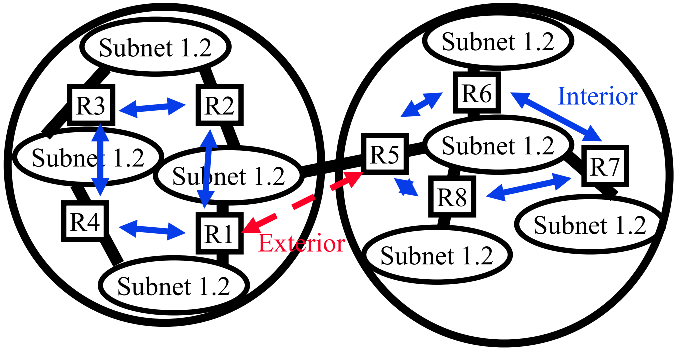
The Network Layer: Control Plane



Raj Jain
Washington University in Saint Louis
Saint Louis, MO 63130
Jain@wustl.edu

Audio/Video recordings of this lecture are available on-line at:

http://www.cse.wustl.edu/~jain/cse473-20/

Overview

1. Routing Algorithms: Link-State, Distance Vector
   Dijkstra's algorithm, Bellman-Ford Algorithm
2. Routing Protocols: OSPF, BGP
3. SDN Control Plane
4. ICMP
5. SNMP

Note: This class lecture is based on Chapter 5 of the textbook (Kurose and Ross) and the figures provided by the authors.

Network Layer Functions

- Forwarding: Deciding what to do with a packet using a routing table ⇒ Data plane
- Routing: Making the routing table ⇒ Control Plane

Routing Algorithms

1. Graph abstraction
2. Distance Vector vs. Link State
3. Dijkstra's Algorithm
4. Bellman-Ford Algorithm

Rooting or Routing

- *Rooting* is what fans do at football games, what pigs do for truffles under oak trees in the Vaucluse, and what nursery workers intent on propagation do to cuttings from plants.
- *Routing* is how one creates a beveled edge on a table top or sends a corps of infantrymen into full scale, disorganized retreat

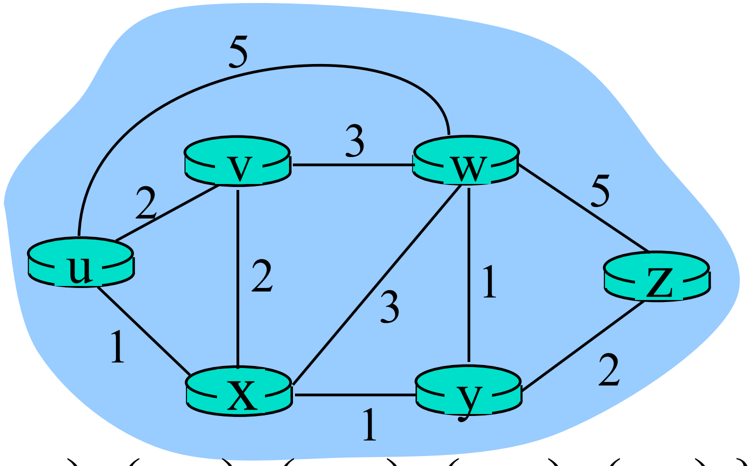
Ref: Piscitello and Chapin, "Open Systems Networking: TCP/IP and OSI," Adison-Wesley, 1993, p413

Routeing or Routing

- Routeing: British
- Routing: American
- Since Oxford English Dictionary is much heavier than any other dictionary of American English, British English generally prevails in the documents produced by ISO and CCITT; wherefore, most of the international standards for routing standards use the routeing spelling.

Ref: Piscitello and Chapin, "Open Systems Networking: TCP/IP and OSI," Adison-Wesley, 1993, p413

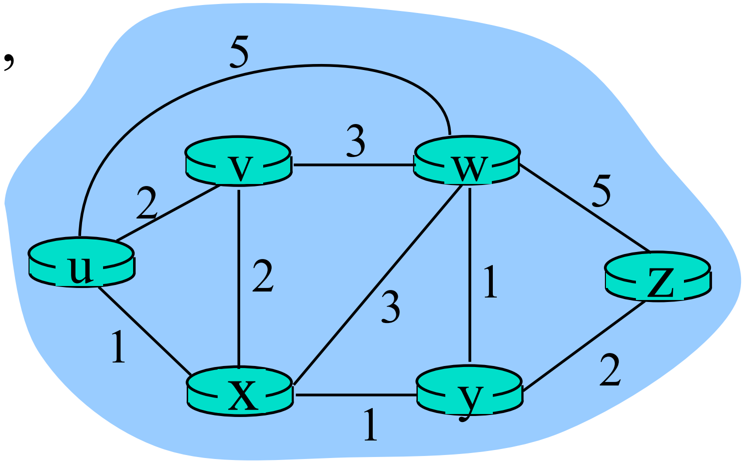
Graph abstraction



- ❑ Graph: G = (N,E)
- ❑ N = Set of routers
  = { u, v, w, x, y, z }
- ❑ E = Set of links
  ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }
- ❑ Each link has a cost, e.g., c(w,z) = 5
- ❑ Cost of path $(x_1, x_2,\ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$
- ❑ Routing Algorithms find the least cost path
- ❑ We limit to "Undirected" graphs, i.e., cost is same in both directions

Distance Vector vs. Link State

Distance Vector:



- ❑ Vector of distances to all nodes, e.g., u: {u:0, v:2, w:5, x:1, y:2, z:4}
- ❑ Sent to neighbors, e.g., u will send to v, w, x
- ❑ Large vectors to small # of nodes Tell about the world to neighbors
- ❑ Older method. Used in RIP.

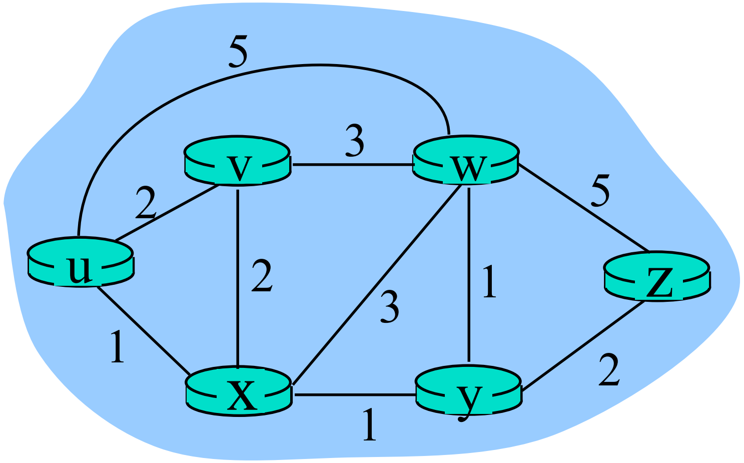
Link State:

- ❑ Vector of link cost to neighbors, e.g, u: {v:2, w:5, x:1}
- ❑ Sent to all nodes, e.g., u will send to v, w, x, y, z
- ❑ Small vectors to large # of nodes Tell about the neighbors to the world
- ❑ Newer method. Used in OSPF.

Dijkstra's Algorithm

- Goal: Find the least cost paths from a given node to all other nodes in the network
- Notation:
  c(i,j) = Link cost from i to j if i and j are connected
  D(k) = Total path cost from s to k
  N' = Set of nodes so far for which the least cost path is known
- Method:
  - Initialize: N'={u}, D(v) = c(u,v) for all neighbors of u
  - Repeat until N includes all nodes:
    - Find node *w* $\notin$ N', whose D(w) is minimum
    - Add w to N'

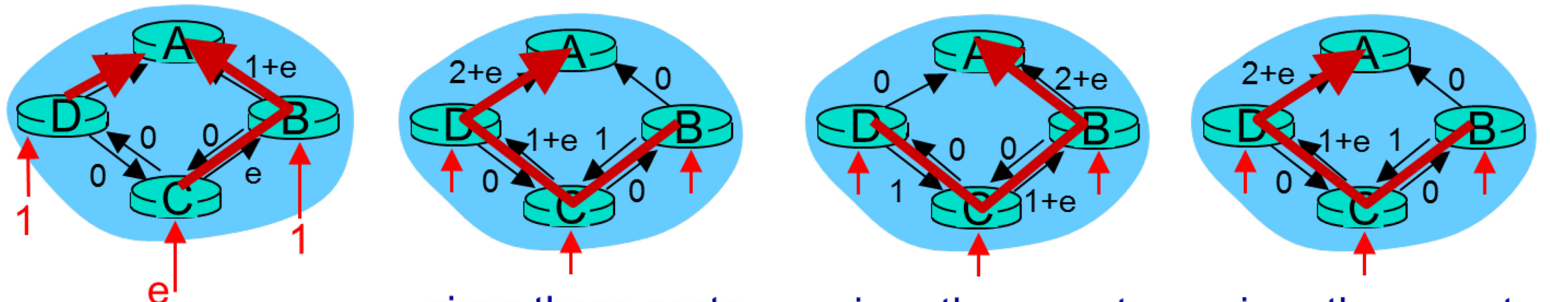
Dijkstra's Algorithm: Example



	N'	D(v)	Path	D(w)	Path	D(x)	Path	D(y)	Path	D(z)	Path
0	{u}	2	u-v	5	u-w	1	u-x	∞	-	∞	-
1	{u, x}	2	u-v	4	u-x-w			2	u-x-y	∞	-
2	{u, x, y}	2	u-v	3	u-x-y-w					4	u-x-y-z
3	{u, x, y, v}			3	u-x-y-w					4	u-x-y-z
4	{u, x, y, v, w}									4	u-x-y-z
5	{u, x, y, v, w, z}										

Complexity and Oscillations

- ❑ *Algorithm complexity:* n nodes
  - Each iteration: need to check all nodes, w, not in N
  - n(n+1)/2 comparisons: $O(n^2)$
  - More efficient implementations possible: O(n log n)
- ❑ *Oscillations Possible:* e.g., support link cost equals amount of carried traffic



initially

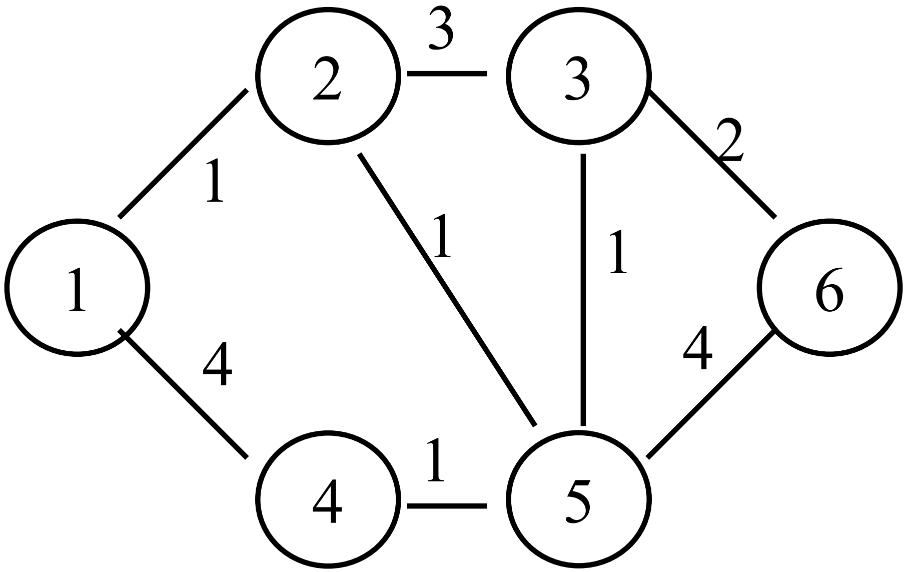
given these costs, find new routing…. resulting in new costs

given these costs, find new routing…. resulting in new costs

given these costs, find new routing…. resulting in new costs

Homework 5A

[12 points] Prepare the routing calculation ***table*** for node 1 in the following network using Dijkstra's algorithm. Explain how you computed new entries in each row.



Bellman-Ford Algorithm

- Notation:

  u = Source node

  $c(i,j)$ = link cost from i to j

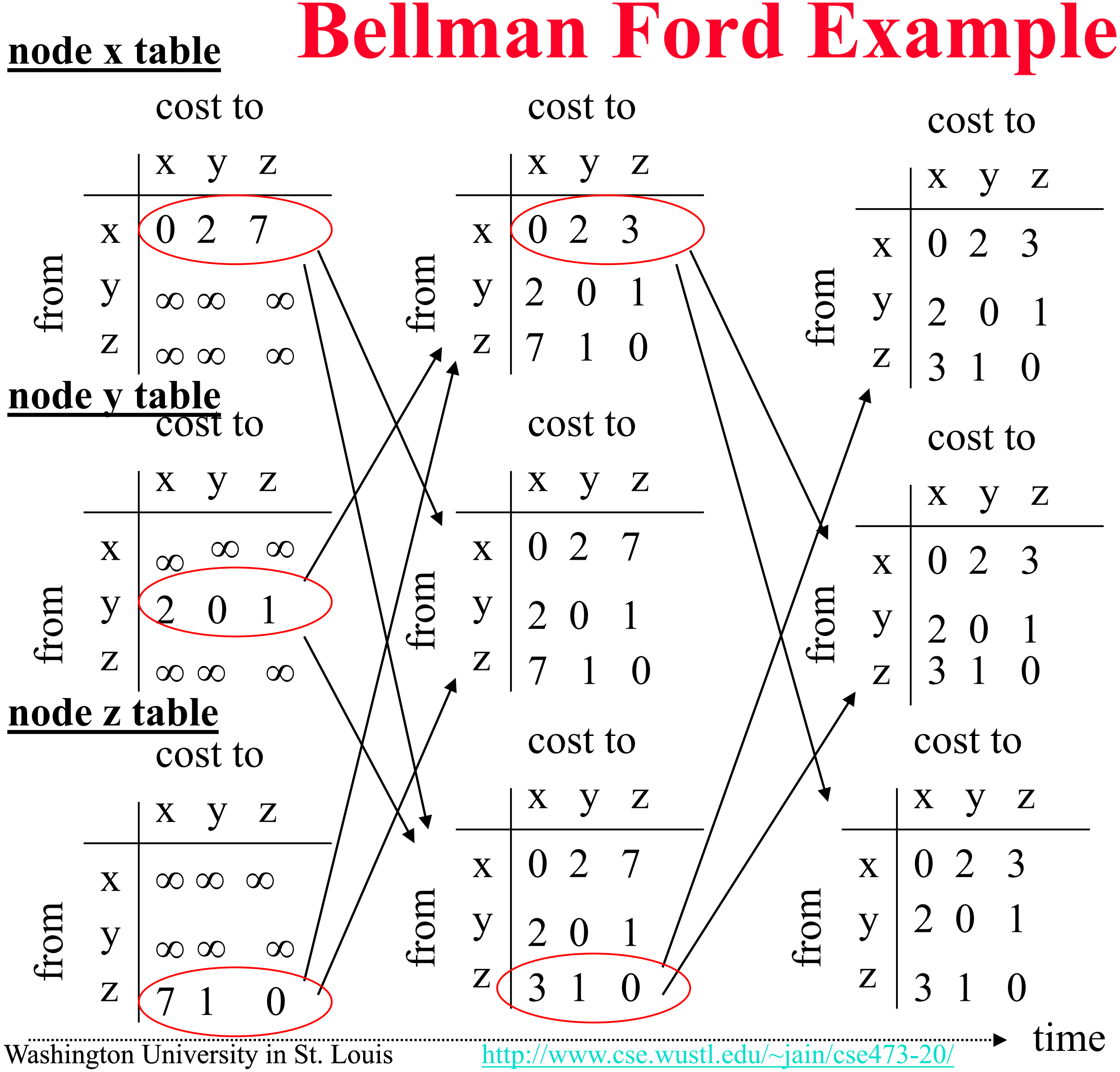
  h = Number of hops being considered

  $D_u(n)$ = Cost of h-hop path from u to n

- Method:

  1. Initialize: $D_u(n) = \infty$ for all $n \neq u$; $D_u(u) = 0$
  2. For each node: $D_u(n) = \min_j [D_u(j) + c(j,n)]$
  3. If any costs change, repeat step 2

Bellman Ford Example 1



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

cost to

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

node y table

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

cost to

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

cost to

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

node z table

cost to

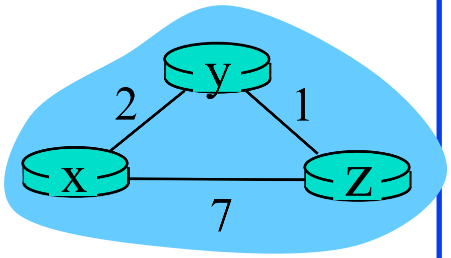
from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

cost to

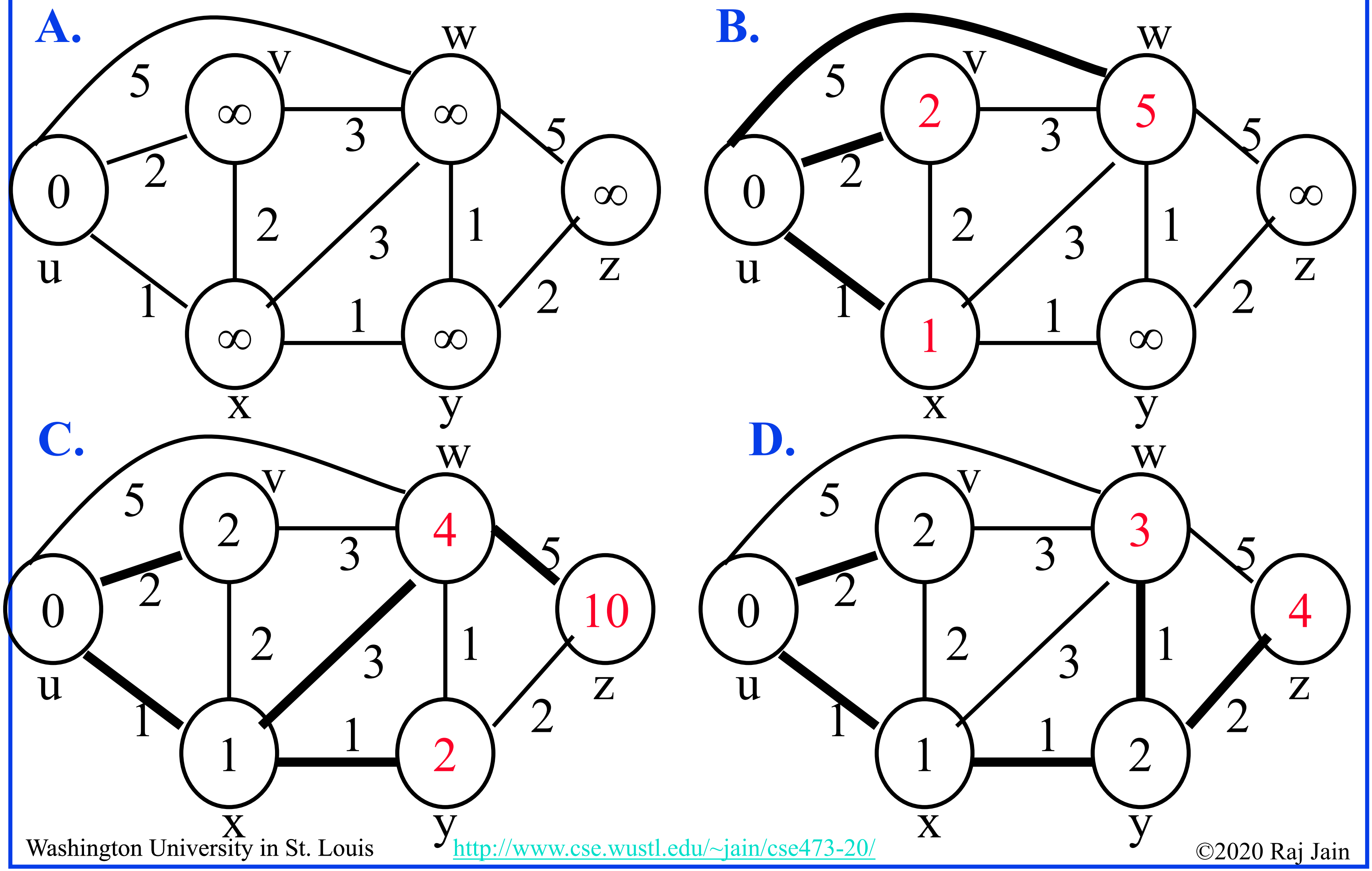
from	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

cost to

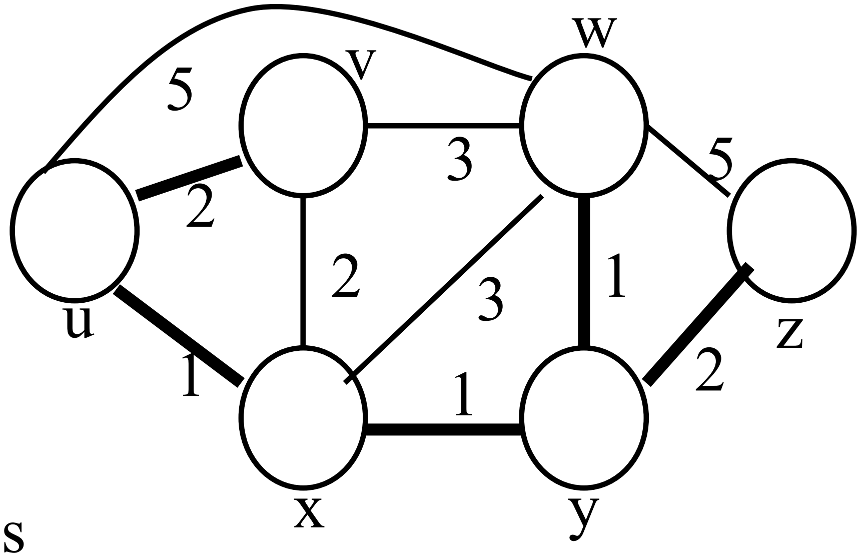
from	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0



Bellman-Ford Example 2



Bellman-Ford: Tabular Method

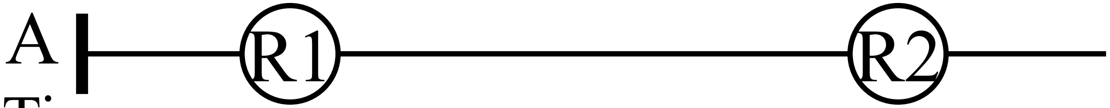


If cost changes

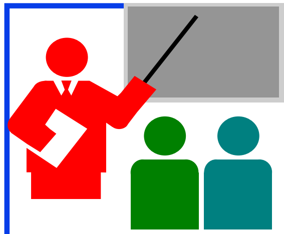
⇒ Recompute the costs to all neighbors

h	D(v)	Path	D(w)	Path	D(x)	Path	D(y)	Path	D(z)	Path
0	∞	-	∞	-	∞	-	∞	-	∞	-
1	2	u-v	5	u-w	1	u-x	∞	-	∞	-
2	2	u-v	4	u-x-w	1	u-x	2	u-x-y	10	u-w-z
3	2	u-v	3	u-x-y-w	1	u-x	2	u-x-y	4	u-x-y-z
4	2	u-v	3	u-x-y-w	1	u-x	2	u-x-y	4	u-x-y-z

Counting to Infinity Problem



	Time						
	0	A	1	R1			
	1				A	2	R1
R1 loses A	1.5	A	16	R1			
R1 hears from R2 →	2	A	3	R2	A	2	R1
(Before it tells R2)	3				A	4	R1
	4	A	5	R2			
	5				A	6	R1
	6	A	7	R2			
			…				
			…				
	14	A	15	R2			
	15				A	16	R1
	16	A	16	R2			



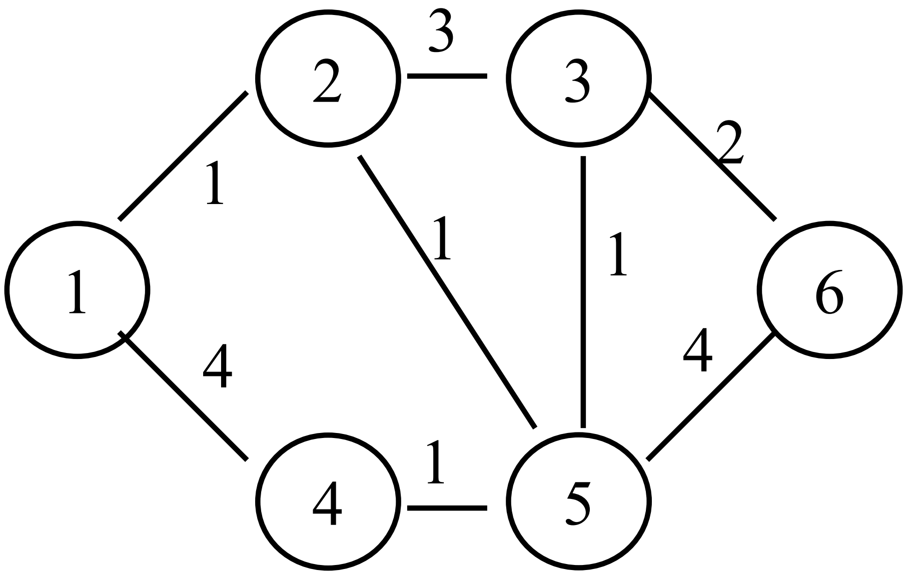
Routing Algorithms: Summary

1. Distance Vectors: Distance to all nodes in the network sent to neighbors. Small # of large messages.
2. Link State: Cost of link to neighbors sent to entire network. Large # of small messages.
3. Dijkstra’s algorithm is used to compute shortest path using link state
4. Bellman Ford’s algorithm is used to compute shortest paths using distance vectors
5. Distance Vector algorithms suffer from the count-to-infinity problem

Ref: Read Section 5.2 of the textbook and try review questions R3-R6.

Homework 5B

[10 points] Prepare the routing calculation ***table*** for node 1 in the following network using the Bellman-Ford Algorithm. Explain how you computed new entries in each row.



Routing Protocols

1. Autonomous Systems (AS)
2. Open Shortest Path First (OSPF)
   - OSPF Areas
3. Border Gateway Protocol (BGP)

Autonomous Systems

- An internet connected by homogeneous routers under the administrative control of a single entity

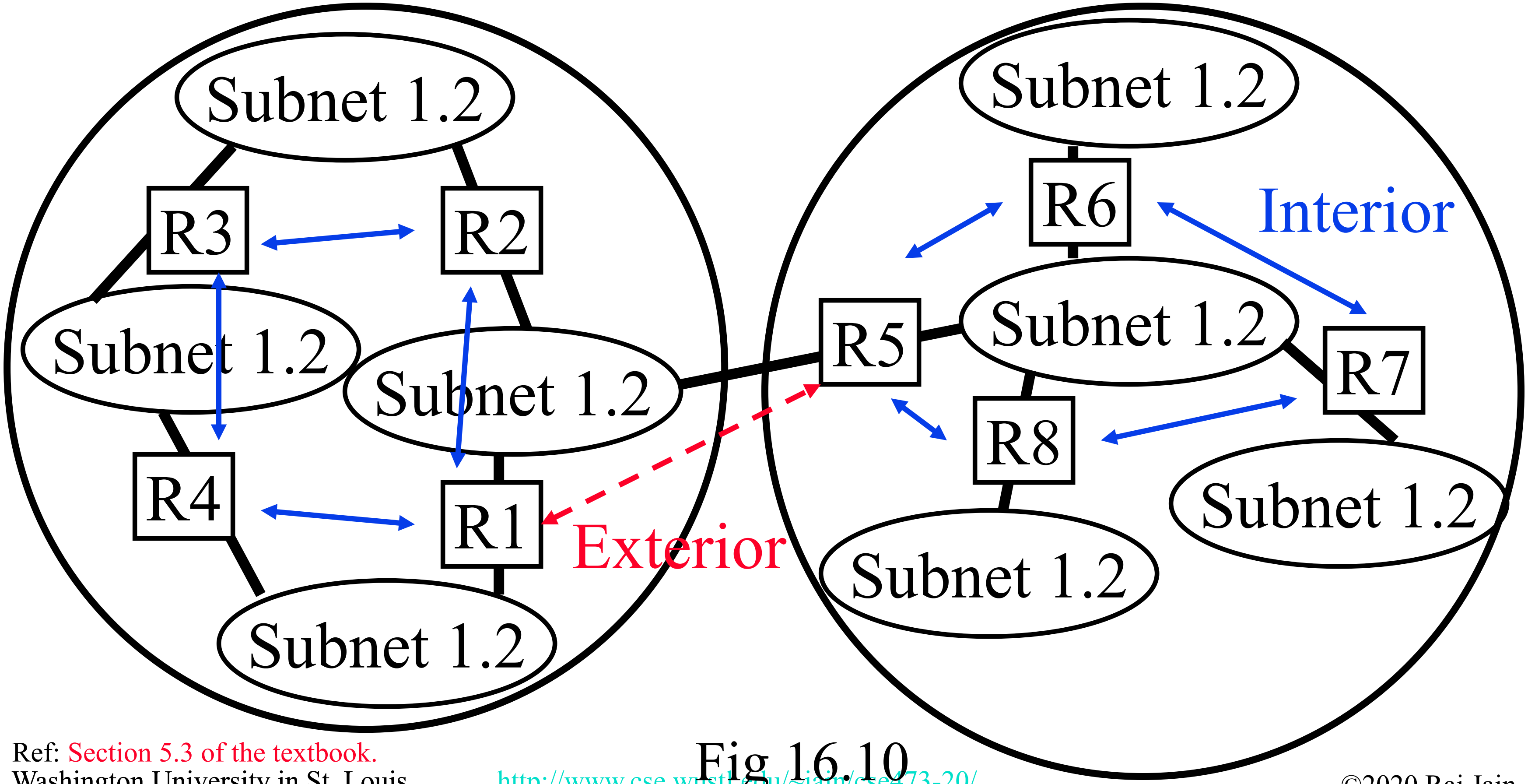


Fig 16.10

Ref: Section 5.3 of the textbook.

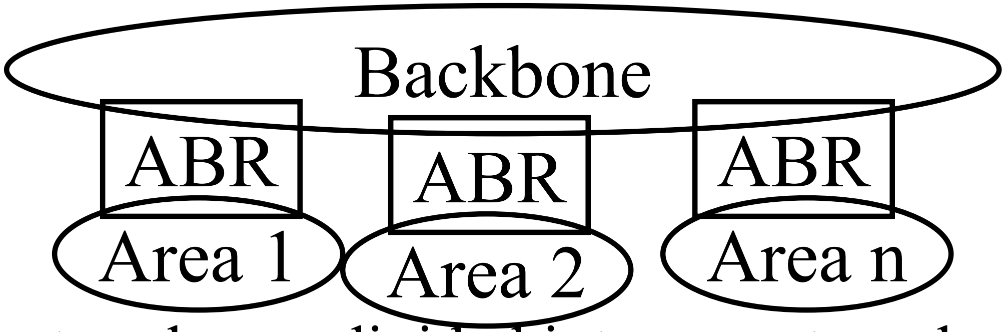
Routing Protocols

- Interior Router Protocol (IRP): Used for passing routing information among routers internal to an autonomous system. Also known as IGP.
  - Examples: RIP, OSPF, IGRP
- Exterior Router Protocol (ERP): Used for passing routing information among routers between autonomous systems. Also known as EGP.
  - Examples: EGP, BGP, IDRP
    Note: EGP is a class as well as an instance in that class.

Open Shortest Path First (OSPF)

- Uses true metrics (not just hop count)
- Uses subnet masks
- Allows load balancing across equal-cost paths
- Supports type of service (ToS)
- Allows external routes (routes learnt from other autonomous systems)
- Authenticates route exchanges
- Quick convergence
- Direct support for multicast
- Link state routing ⇒ Each router broadcasts its connectivity with neighbors to entire network

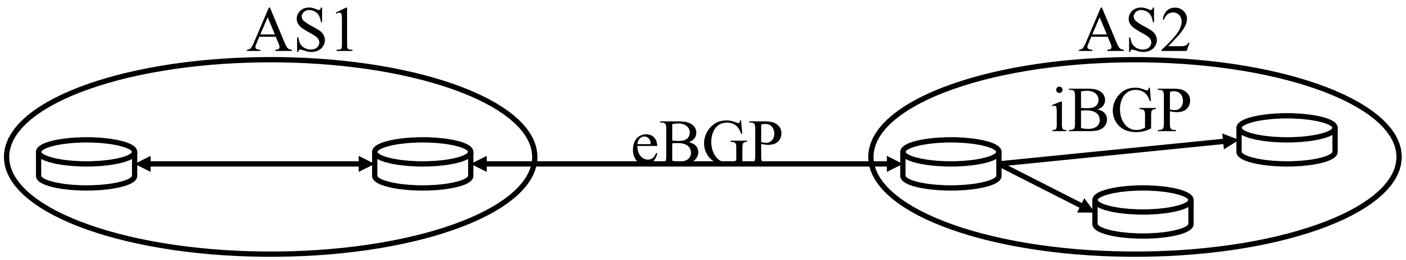
OSPF Areas



- Large networks are divided into areas to reduce routing traffic.
- LSAs are flooded throughout the area
- Area border routers (ABRs) summarize the topology and transmit it to the backbone area
- Backbone routers forward it to other areas
- ABRs connect an area with the backbone area.
  ABRs contain OSPF data for two areas.
  ABRs run OSPF algorithms for the two areas.
- If there is only one area in the AS, there is no backbone area and there are no ABRs.

Border Gateway Protocol

- Inter-autonomous system protocol [RFC 1267]
- Used since 1989 but not extensively until recently
- Runs on TCP (segmentation, reliable transmission)
- Advertises all transit ASs on the path to a destination address
- A router may receive multiple paths to a destination ⇒ Can choose the best path
- iBGP used to forward paths inside the AS.
  eBGP used to exchange paths between ASs.

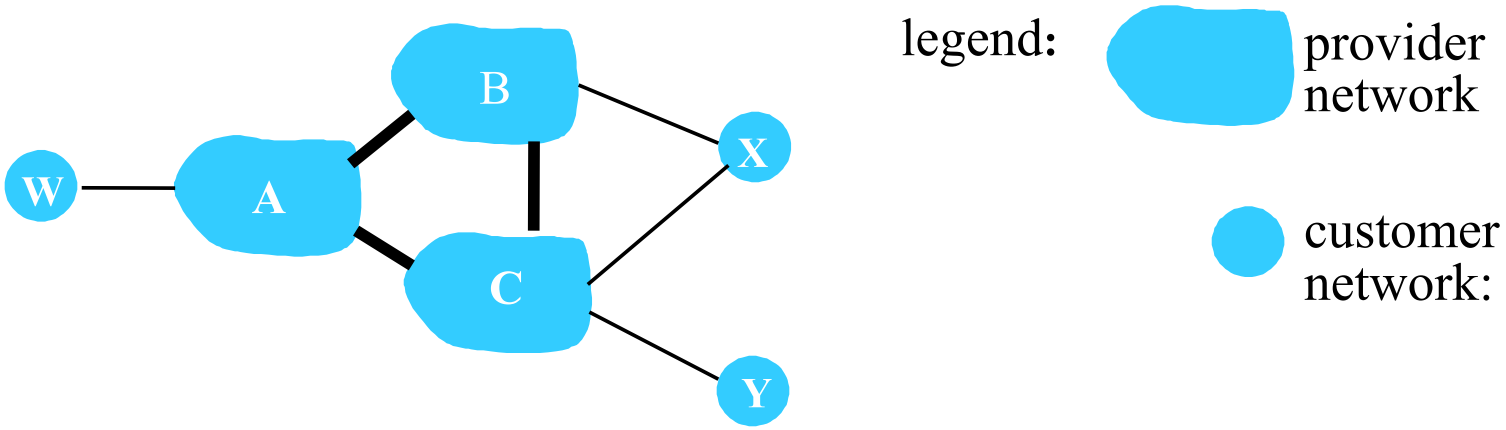


Ref: Section 5.4 of the textbook.

BGP Operations

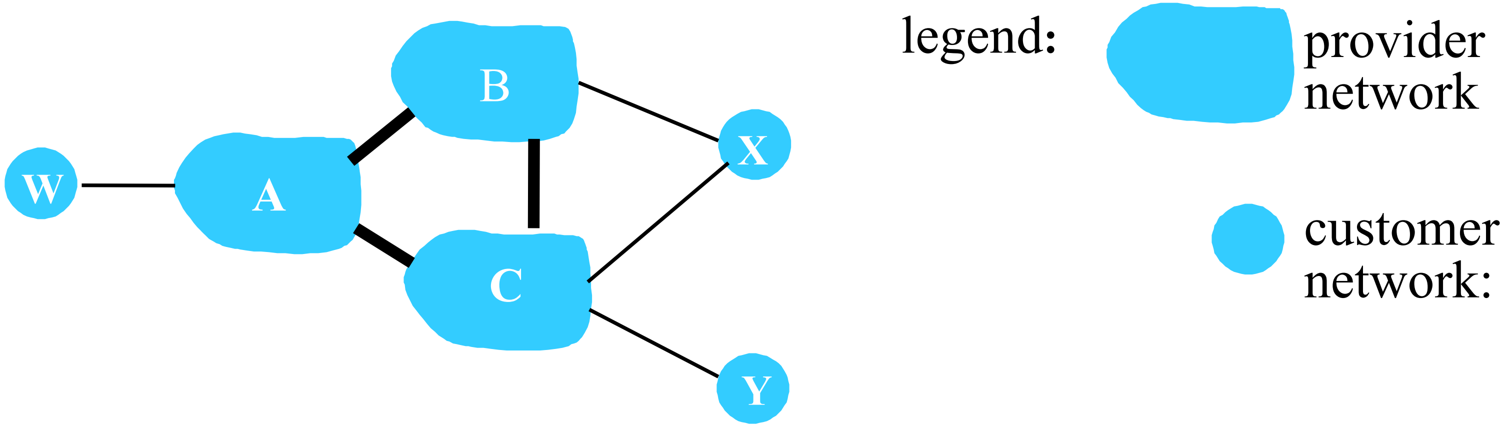
- BGP systems initially exchange entire routing tables. Afterwards, only updates are exchanged.
- BGP messages have the following information:
  - Origin of path information: RIP, OSPF, …
  - AS_Path: List of ASs on the path to reach the dest
  - Next_Hop: IP address of the border router to be used as the next hop to reach the dest
  - Unreachable: If a previously advertised route has become unreachable
- BGP speakers generate update messages to all peers when it selects a new route or some route becomes unreachable.

BGP Routing Policy Example



- A,B,C are provider networks
- X,W,Y are customer (of provider networks)
- X is dual-homed: attached to two networks
  - X does not want to route from B via X to C
  - .. so X will not advertise to B a route to C

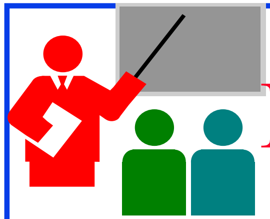
BGP Routing Policy Example (Cont)



- A advertises path A-W to B
- B advertises path B-A-W to X
- Should B advertise path B-A-W to C?
  - No way! B gets no "revenue" for routing C-B-A-W since neither W nor C are B's customers
  - B wants to force C to route to w via A
  - B wants to route *only* to/from its customers!

Intra- vs. Inter-AS Routing

- **Policy:**
  - Inter-AS: admin wants control over how its traffic routed, who routes through its net.
  - Intra-AS: single admin, so no policy decisions needed
- **Scale**:
  - Hierarchical routing saves table size, reduced update traffic
- **Performance**:
  - Intra-AS: can focus on performance
  - Inter-AS: policy may dominate over performance

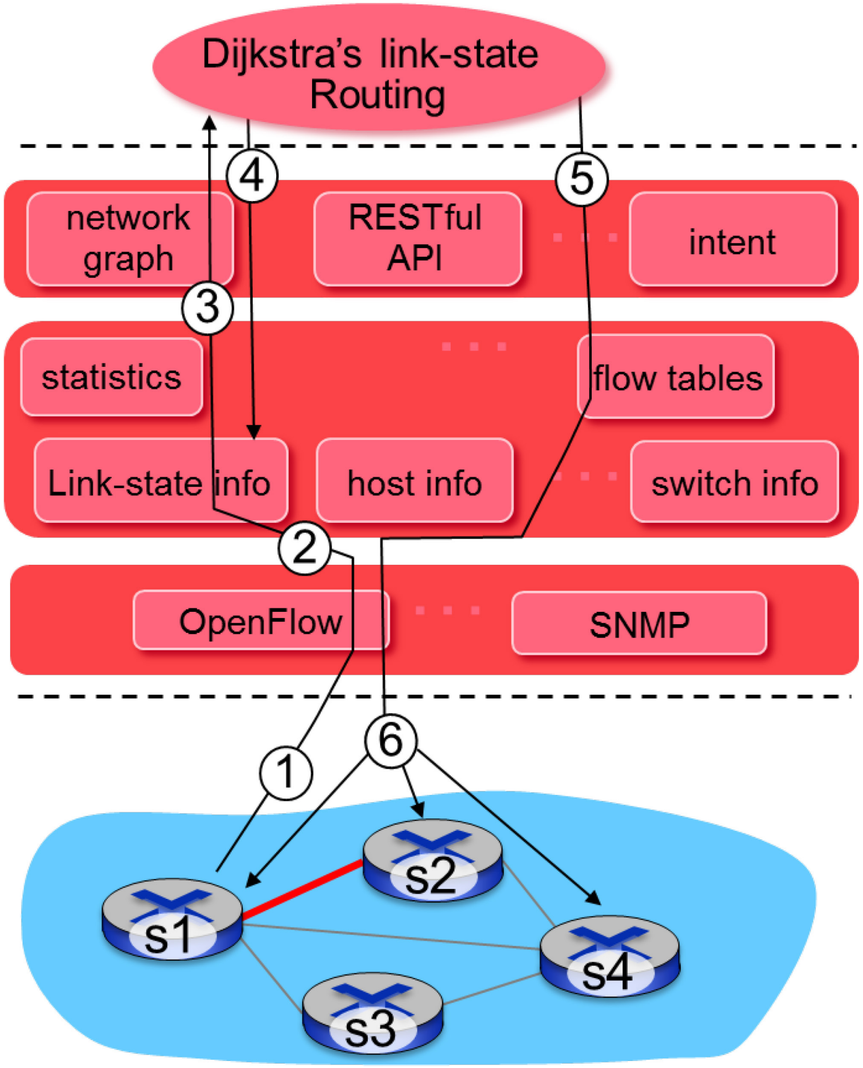


Routing Protocols: Summary

1. OSPF uses link-state routing and divides the autonomous systems into multiple areas.
   Area border router, AS boundary router, designated router
2. BGP is an inter-AS protocol ⇒ Policy driven

Ref: Read Section 5.3 and 5.4 of the textbook and try review questions R7-R13.

SDN Control Plane



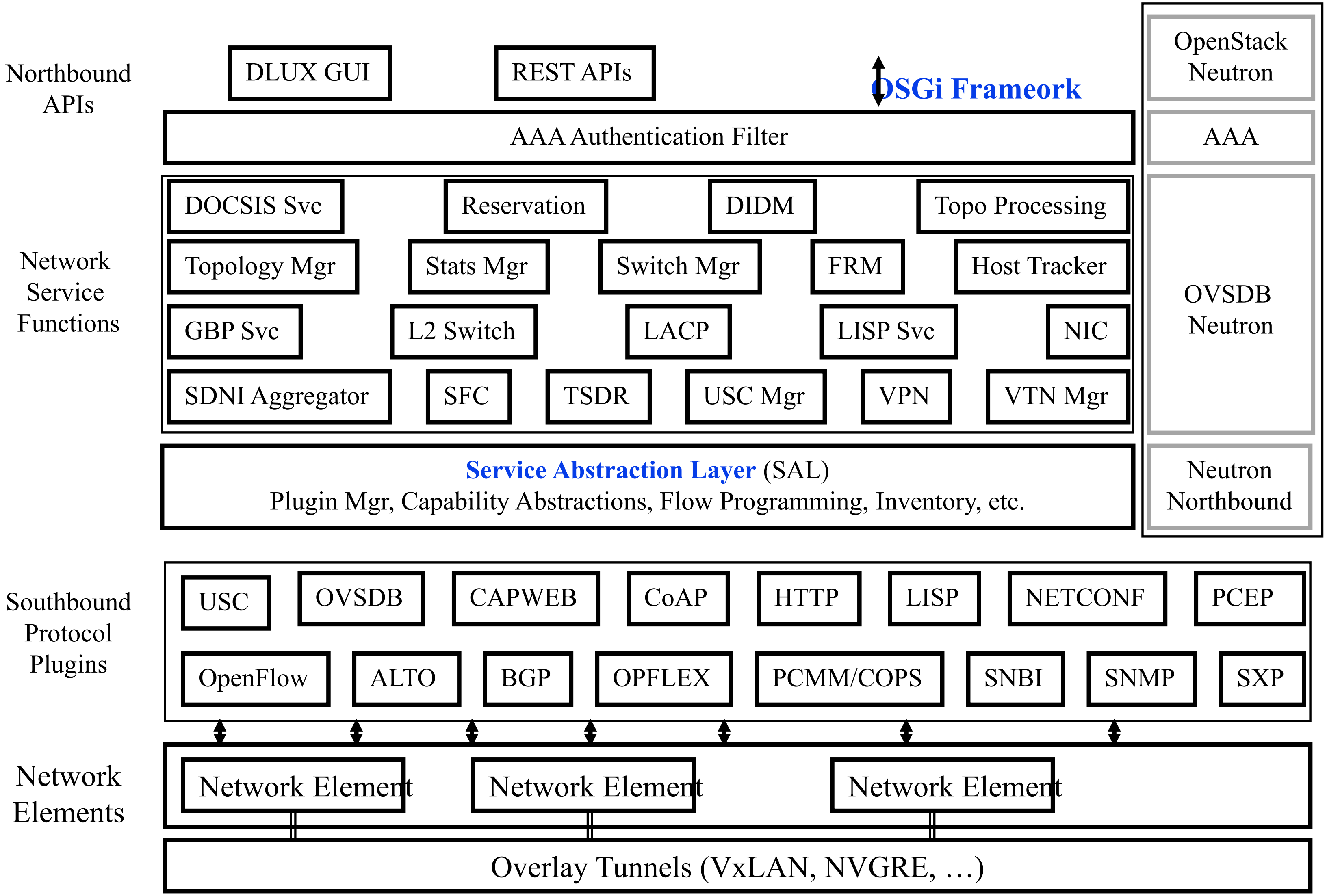
1. S1, experiencing link failure using OpenFlow port status message to notify controller

2. SDN controller receives OpenFlow message, updates link status info

3. Dijkstra's routing algorithm application has previously registered to be called when ever link status changes. It is called.

4. Dijkstra's routing algorithm access network graph info, link state info in controller, computes new routes

Controller Example: OpenDaylight



Northbound APIs

DLUX GUI

REST APIs

OSGi Frameork

AAA Authentication Filter

Network Service Functions

DOCSIS Svc

Reservation

DIDM

Topo Processing

Topology Mgr

Stats Mgr

Switch Mgr

FRM

Host Tracker

GBP Svc

L2 Switch

LACP

LISP Svc

NIC

SDNI Aggregator

SFC

TSDR

USC Mgr

VPN

VTN Mgr

Service Abstraction Layer (SAL)
Plugin Mgr, Capability Abstractions, Flow Programming, Inventory, etc.

OpenStack Neutron

AAA

OVSDB Neutron

Neutron Northbound

Southbound Protocol Plugins

USC

OVSDB

CAPWEB

CoAP

HTTP

LISP

NETCONF

PCEP

OpenFlow

ALTO

BGP

OPFLEX

PCMM/COPS

SNBI

SNMP

SXP

Network Elements

Network Element

Network Element

Network Element

Overlay Tunnels (VxLAN, NVGRE, …)

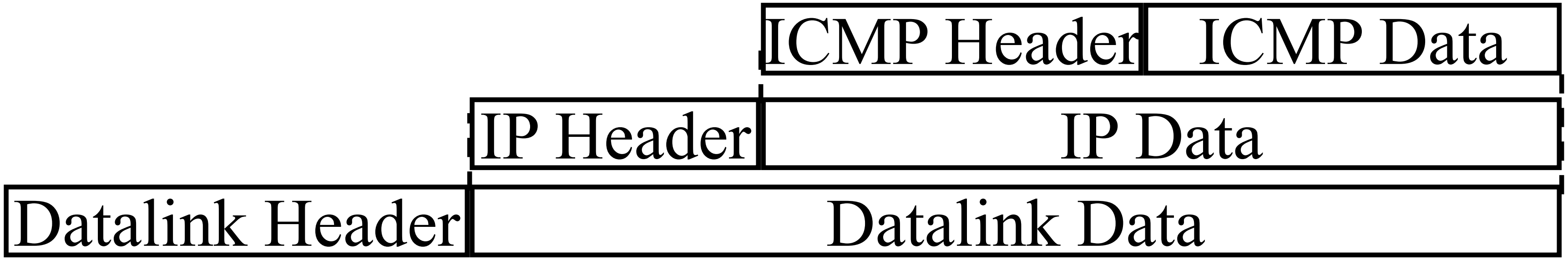
OpenDaylight SDN Controller

- Multi-company collaboration under Linux foundation
- Many projects including OpenDaylight Controller
- Dynamically linked in to a Service Abstraction Layer (SAL) ⇒ SAL figures out how to fulfill the service requested by higher layers irrespective of the southbound protocol
- Modular design
- A rich set of North-bound APIs via RESTful (Web page like) services

Ref: Read Section 5.5 and try review questions R14-R18.

ICMP

- Internet Control Message Protocol
- Required companion to IP. Provides feedback from the network.
- ICMP: Used by IP to send error and control messages
- ICMP uses IP to send its messages (Not UDP)
- ICMP does not report errors on ICMP messages.
- ICMP reports error only on the first fragment



ICMP: Message Types

IP Header	
Type of Message	8b
Error Code	8b
Checksum	16b
Parameters, if any	Var
Information	Var

Type	Message
0	Echo reply
3	Destination unreachable
4	Source quench
5	Redirect
8	Echo request
11	Time exceeded
12	Parameter unintelligible
13	Time-stamp request
14	Time-stamp reply
15	Information request
16	Information reply
17	Address mask request
18	Address mask reply

ICMP Messages

- Source Quench: Please slow down!
  I just dropped one of your datagrams.
- Time Exceeded: Time to live field in one of your packets became zero." or "Reassembly timer expired at the destination.
- Fragmentation Required: Datagram was longer than MTU and "No Fragment bit" was set.
- Address Mask Request/Reply: What is the subnet mask on this net? Replied by "Address mask agent"
- PING uses ICMP echo
- Tracert uses TTL expired

Ref: Read Section 5.6 of the textbook and try erview questions R19-R20.

Trace Route Example

```
C:\>tracert www.google.com

Tracing route to www.l.google.com [74.125.93.147]
over a maximum of 30 hops:

  1     3 ms     1 ms     1 ms  192.168.0.1
  2    12 ms    10 ms     9 ms  bras4-l0.stlsmo.sbcglobal.net [151.164.182.113]
  3    10 ms     8 ms     8 ms  dist2-vlan60.stlsmo.sbcglobal.net [151.164.14.163]
  4     9 ms     7 ms     7 ms  151.164.93.224
  5    25 ms    22 ms    22 ms  151.164.93.49
  6    25 ms    22 ms    22 ms  151.164.251.226
  7    30 ms    28 ms    28 ms  209.85.254.128
  8    61 ms    57 ms    58 ms  72.14.236.26
  9    54 ms    52 ms    51 ms  209.85.254.226
 10    79 ms   160 ms    67 ms  209.85.254.237
 11    66 ms    57 ms    68 ms  64.233.175.14
 12    60 ms    58 ms    58 ms  qw-in-f147.google.com [74.125.93.147]

Trace complete.
```

Lab 5A: ICMP

- [14 points] Download the Wireshark traces from http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip
- Open *icmp-ethereal-trace-1* in Wireshark.
  Select **View → Expand All**. Answer the following questions:

1. Examine Frame 3.
   - A. What is the IP address of your host? What is the IP address of the destination host?
   - B. Why is it that an ICMP packet does not have source and destination port numbers?
   - C. What are the ICMP type and code numbers? What other fields does this ICMP packet have? How many bytes are the checksum, sequence number and identifier fields?

Lab 5A (Cont)

2. Examine Frame 4. What are the ICMP type and code numbers?

- ❑ Open *icmp-ethereal-trace-2* in Wireshark. Answer the following questions:

3. Examine Frame 2. What fields are included in this ICMP error packet?
4. Examine Frames 100, 101, and 102. How are these packets different from the ICMP error packet 2? Why are they not error packets?

Network Management

- □ What is Network Management?
- □ Components of Network Management
- □ How is Network Managed?
- □ SNMP protocol

What is Network Management?

- Traffic on Network = Data + Control + Management
- **Data** = Bytes/Messages sent by users
- **Control** = Bytes/messages added by the system to properly transfer the data (e.g., routing messages)
- **Management** = Optional messages to ensure that the network functions properly and to handle the issues arising from malfunction of any component
- If all components function properly,
  control is still required but management is optional.
- Examples:
  - Detecting failures of an interface card at a host or a router
  - Monitoring traffic to aid in resource deployment
  - Intrusion Detection

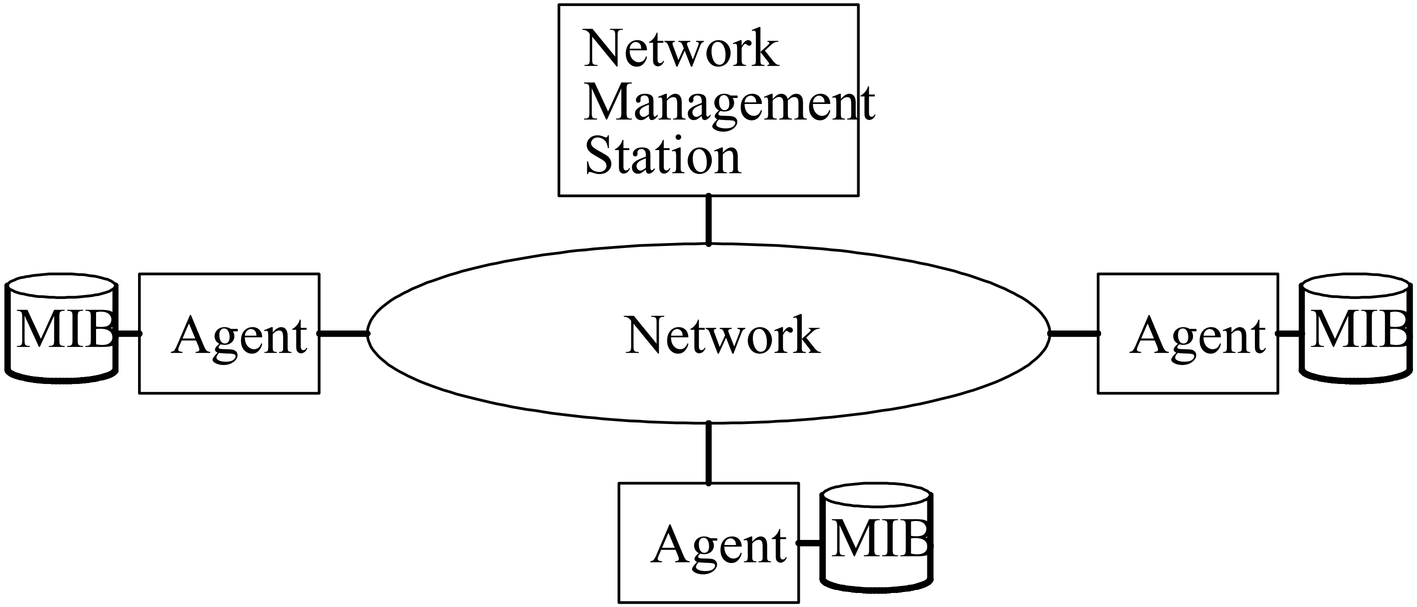
Components of Network Management

1. **Fault Management**:
   Detect, log, and respond to fault conditions
2. **Configuration Management**:
   Track and control which devices are on or off
3. **Accounting Management**:
   Monitor resource usage for records and billing
4. **Performance Management**:
   Measure, report, analyze, and control traffic, messages
5. **Security Management**:
   Enforce policy for access control, authentication, and authorization

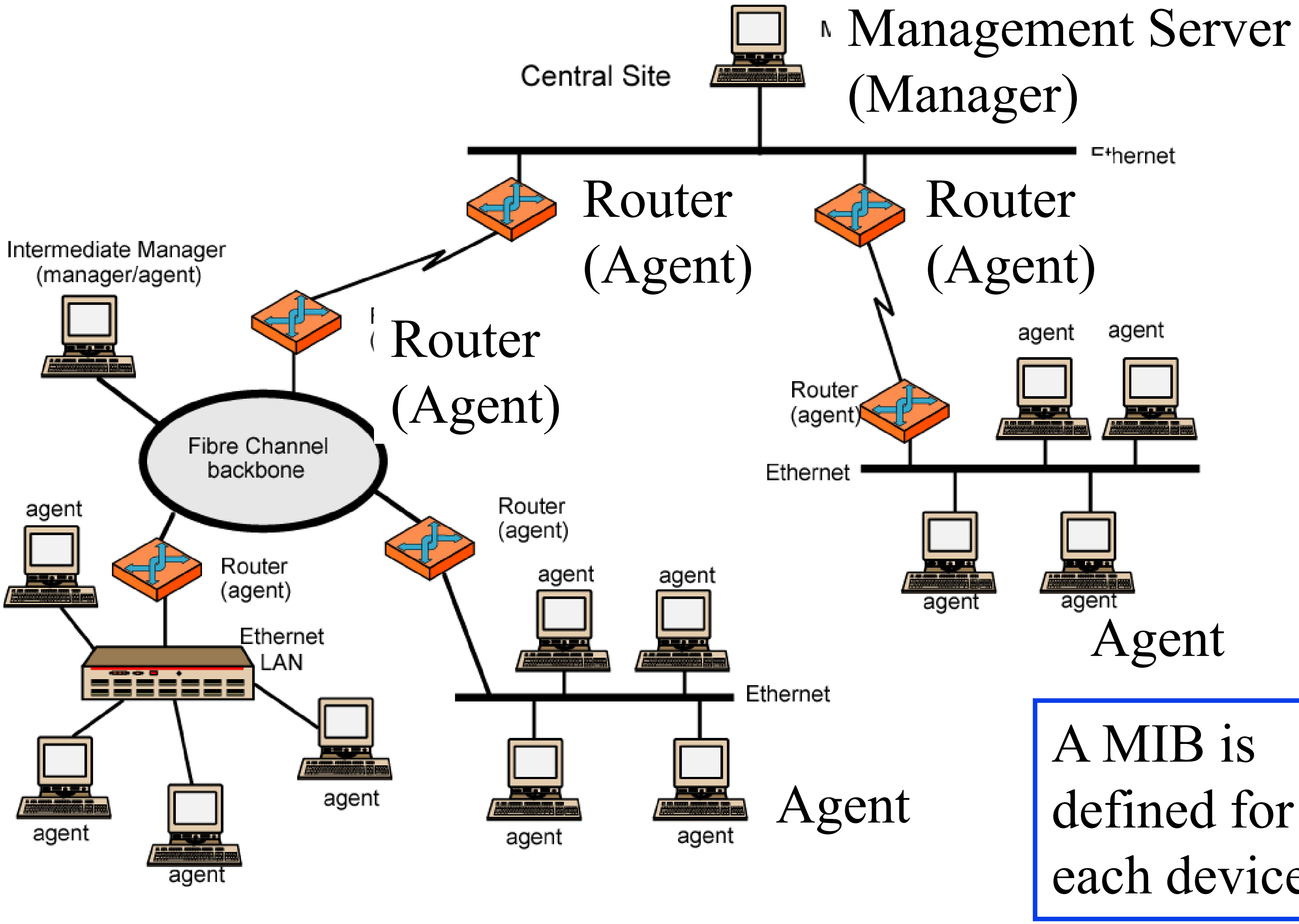
- **FCAPS**

How is Network Managed?

- Management = Initialization, Monitoring, Control
- Manager, Agents, and
  Management Information Base (MIB)



Example of Network Management



A MIB is defined for each device

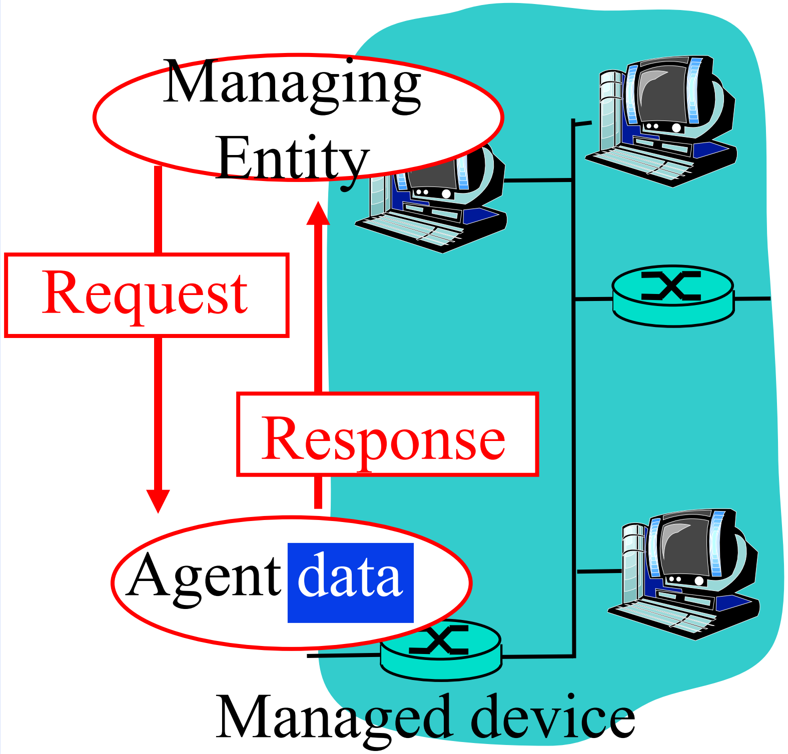
SNMP

- Based on Simple Gateway Management Protocol (SGMP) – RFC 1028 – Nov 1987
- SNMP = Simply *N*ot *M*y *P*roblem [Marshall Rose]
  Simple Network Management Protocol
- RFC 1058, April 1988
- Only Five commands

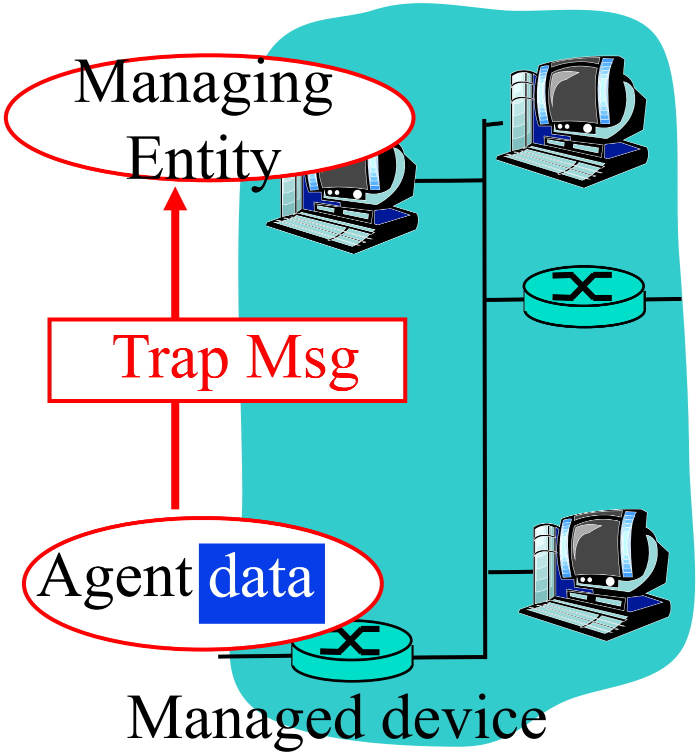
Command	Meaning
get-request	Fetch a value
get-next-request	Fetch the next value (in a tree)
get-response	Reply to a fetch operation
set-request	Store a value
trap	An event

SNMP protocol

Two ways to convey MIB info, commands:

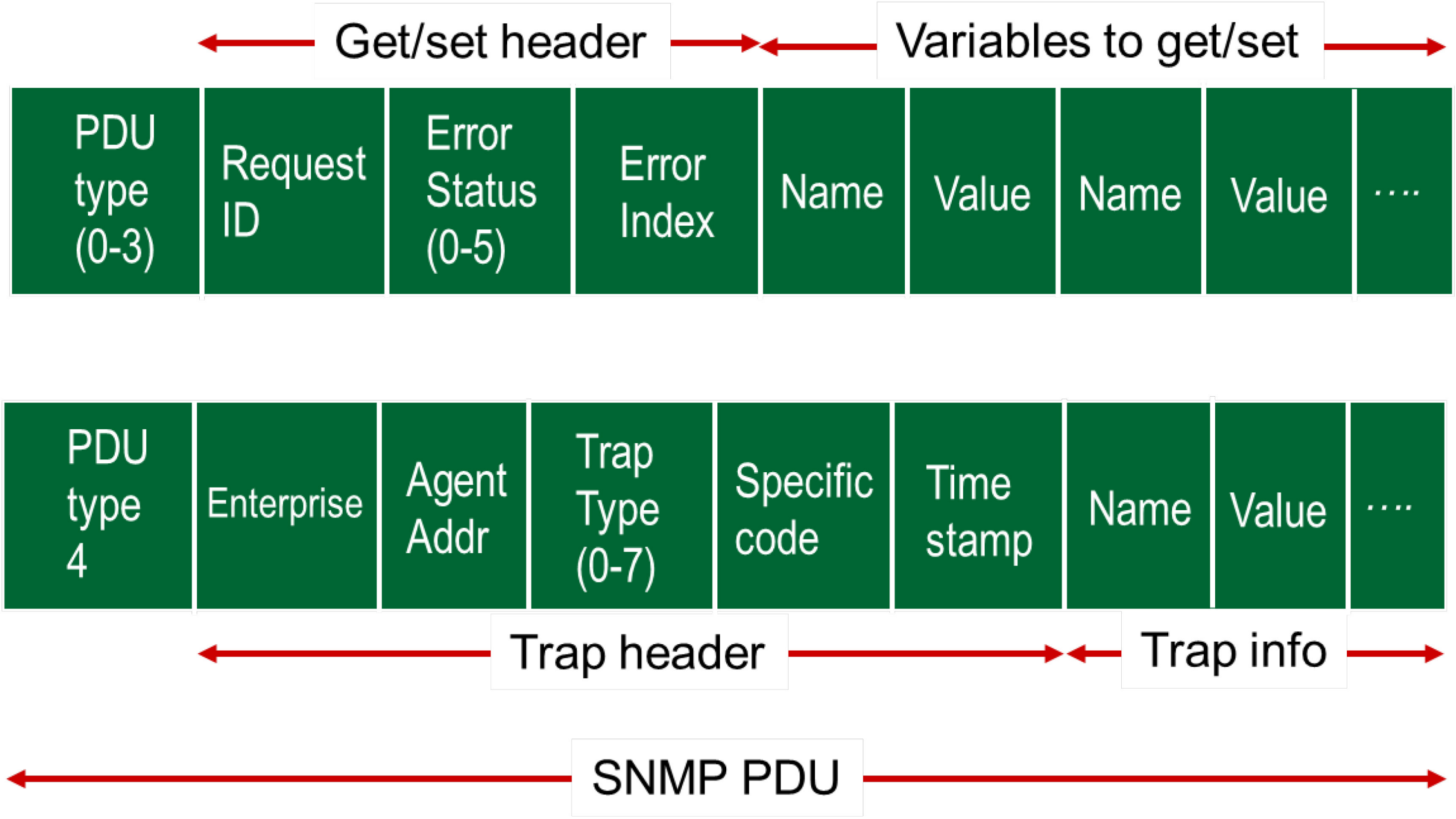


Request/response mode



Trap mode

SNMP Message Formats



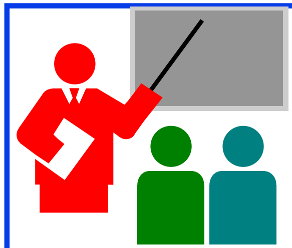
Get/set header | Variables to get/set

PDU type (0-3)	Request ID	Error Status (0-5)	Error Index	Name	Value	Name	Value	….

PDU type 4	Enterprise	Agent Addr	Trap Type (0-7)	Specific code	Time stamp	Name	Value	….

Trap header | Trap info

SNMP PDU

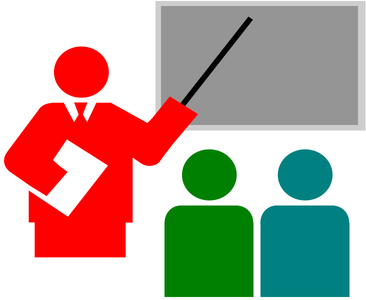


Network Management: Summary

1. Management = Initialization, Monitoring, and Control
2. Standard MIBs defined for each object
3. SNMP = Only 5 commands in the first version

Ref: Read Section 5.7 of the textbook and try review questions R21-R23.

Network Layer Control Plane: Summary



1. Dijkstra's algorithm allows path computation using link state
2. Bellman Ford's algorithm allows path computation using distance vectors.
3. OSPF is a link state IGP.
4. BGP is an EGP and uses path vectors
5. SDN controllers use various algorithms for centralized computation of path and other policies
6. ICMP is IP control protocol is used to convey errors
7. SNMP is the simple network management protocol to manage all devices and protocols in a network

Lab 5B: ICMP Ping Programming

[25 points] In this lab, you will gain a better understanding of Internet Control Message Protocol (ICMP). You will learn to implement a Ping application using ICMP request and reply messages.

Ping is a computer network application used to test whether a particular host is reachable across an IP network. It is also used to self-test the network interface card of the computer or as a latency test. It works by sending ICMP "echo reply" packets to the target host and listening for ICMP "echo reply" replies. The "echo reply" is sometimes called a pong. Ping measures the round-trip time, records packet loss, and prints a statistical summary of the echo reply packets received (the minimum, maximum, and the mean of the round-trip times and in some versions the standard deviation of the mean).

Your task is to develop your own Ping application in Python. Your application will use ICMP but, in order to keep it simple, will not exactly follow the official specification in RFC 1739. Note that you will only need to write the client side of the program, as the functionality needed on the server side is built into almost all operating systems.

You should complete the Ping application so that it sends ping requests to a specified host separated by approximately one second. Each message contains a payload of data that includes a timestamp. After sending each packet, the application waits up to one second to receive a reply. If one second goes by without a reply from the server, then the client assumes that either the ping packet or the pong packet was lost in the network (or that the server is down).

Lab 5B (Cont)

Code

Below you will find the skeleton code for the client. You are to complete the skeleton code. The places where you need to fill in code are marked with **#Fill in start** and **#Fill in end**. Each place may require one or more lines of code. This code was written for Python V2.7 and may not run on higher versions.

Additional Notes

In "receiveOnePing" method, you need to receive the structure ICMP_ECHO_REPLY and fetch the information you need, such as checksum, sequence number, time to live (TTL), etc. Study the "sendOnePing" method before trying to complete the "receiveOnePing" method.

You do not need to be concerned about the checksum, as it is already given in the code.

This lab requires the use of raw sockets. In some operating systems, you may need **administrator/root privileges** to be able to run your Pinger program.

Testing the Pinger

First, test your client by sending packets to localhost, that is, 127.0.0.1.

Then, you should see how your Pinger application communicates across the network by pinging servers in different continents.

What to Hand in

- You will hand in the complete client code and screenshots of your Pinger output for four target hosts: north-america.pool.ntp.org, europe.pool.ntp.org, asia.pool.ntp.org, south-america.pool.ntp.org

Lab 5B (Cont)

Skeleton Python Code for the ICMP Pinger

```
from socket import *
import os
import sys
import struct
import time
import select
import binascii
ICMP_ECHO_REQUEST = 8

def checksum(string):
    csum = 0
    countTo = (len(string) // 2) * 2
    count = 0
    while count < countTo:
        thisVal = ord(string[count+1]) * 256 + ord(string[count])
        csum = csum + thisVal
        csum = csum & 0xffffffff
        count = count + 2

    if countTo < len(string):
        csum = csum + ord(string[len(string) - 1])
        csum = csum & 0xffffffff

    csum = (csum >> 16) + (csum & 0xffff)
    csum = csum + (csum >> 16)
    answer = ~csum
    answer = answer & 0xffff
    answer = answer >> 8 | (answer << 8 & 0xff00)
    return answer
```

Lab 5B (Cont)

```
def receiveOnePing(mySocket, ID, timeout, destAddr):
    timeLeft = timeout
    while 1:
        startedSelect = time.time()
        whatReady = select.select([mySocket], [], [], timeLeft)
        howLongInSelect = (time.time() - startedSelect)
        if whatReady[0] == []: # Timeout
            return "Request timed out."
        timeReceived = time.time()
        recPacket, addr = mySocket.recvfrom(1024)
        #Fill in start
        #Fetch the ICMP header from the IP packet
        #Fill in end
        timeLeft = timeLeft - howLongInSelect
        if timeLeft <= 0:
            return "Request timed out.”
```

Lab 5B (Cont)

```
def sendOnePing(mySocket, destAddr, ID):
    # Header is type (8), code (8), checksum (16), id (16), sequence (16)
    myChecksum = 0
    # Make a dummy header with a 0 checksum
    # struct -- Interpret strings as packed binary data
    header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0, myChecksum, ID, 1)
    data = struct.pack("d", time.time())
    # Calculate the checksum on the data and the dummy header.
    myChecksum = checksum(str(header + data))

    # Get the right checksum, and put in the header
    if sys.platform == 'darwin':
        # Convert 16-bit integers from host to network byte order
        myChecksum = htons(myChecksum) & 0xffff
    else:
        myChecksum = htons(myChecksum)
    header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0, myChecksum, ID, 1)
    packet = header + data

    mySocket.sendto(packet, (destAddr, 1)) # AF_INET address must be tuple, not str
    # Both LISTS and TUPLES consist of a number of objects
    # which can be referenced by their position number within the object.
```

Lab 5B (Cont)

```
def doOnePing(destAddr, timeout):
    icmp = getprotobyname("icmp")
    # SOCK_RAW is a powerful socket type. For more details: http://sock-raw.org/papers/sock_raw
    mySocket = socket(AF_INET, SOCK_RAW, icmp)
    myID = os.getpid() & 0xFFFF     # Return the current process i
    sendOnePing(mySocket, destAddr, myID)
    delay = receiveOnePing(mySocket, myID, timeout, destAddr)
    mySocket.close()
    return delay

def ping(host, timeout=1):
    # timeout=1 means: If one second goes by without a reply from the server,
    # the client assumes that either the client's ping or the server's pong is lost
    dest = gethostbyname(host)
    print("Pinging " + dest + " using Python:")
    print("")
    # Send ping requests to a server separated by approximately one second
    while 1 :
        delay = doOnePing(dest, timeout)
        print(delay)
        time.sleep(1)# one second
    return delay
```

Acronyms

ABR	Area border router
API	Application Programming Interface
AS	Autonomous System
ASBR	Autonomous System Boundary Router
BDR	Backup Designated Router
BGP	Border Gateway Protocol
BR	Backbone Router
CAPWAP	Control and Provisioning of Wireless Access Points
CCITT	Consultative Committee for International Telegraph and Telephone (now ITU-T)
CoAP	Constrained Application Protocol
COPS	Common Open Policy Service
DIDM	Device Identifier and Driver Management
DLUX	OpenDaylight User Interface
DOCSIS	Data over Cable Service Interface Specification
DR	Designated Router
eBGP	exterior BGP

Acronyms (Cont)

EGP	Exterial Gateway Protocol
ERP	Exterior Router Protocol
FCAPS	Fault Configuration Accounting Performance and Security
FRM	Forwarding Rules Manager
GBP	Group Based Policy
GUI	Graphical User Interface
HTTP	Hyper-Text Transfer Protocol
iBGP	interior BGP
ICMP	IP Control Message Protocol
ID	Identifier
IDRP	ICMP Router Discovery Protocol
IGP	Interior Gateway Protocol
IGRP	Interior Gateway Routing Protocol
IP	Internet Protocol
IRP	Interior Router Protocol
ISO	International Standards Organization

Acronyms (Cont)

LACP	Link Aggregation Control Protocol
LSA	Link State Advertisements
MIB	Management Information Base
MTU	Maximum Transmission Unit
NETCONF	Network Configuration Protocol
NIC	Network Interface Card
OSGi	Open Service Gatway Initiative
OSI	Open Service Interconnection
OSPF	Open Shortest Path First
OVSDB	Open Vswitch Database
PCEP	Path Computation Element Protocol
PCMM	Packet Cable Multimedia
REST	Representational State Transfer
RESTful	Representational State Transfer
RFC	Request for Comments
SAL	Service Abstraction Layer

Acronyms (Cont)

SDN	Software Defined Networking
SDNI	SDN domains interface
SFC	Service Function Chaining
SGMP	Simple Gateway Management Protocol
SNBI	Secure Network Bootstrapping Interface
SNMP	Simple Network Management Protocol
SXP	SGT (Security Group Tags) Exchange Protocol
TCP	Transmission Control Protocol
ToS	Type of Service
TSDR	Time Series Data Repository
TTL	Time to Live
UDP	User Datagram Protocol
USC	Unified Secure Channel
VPN	Virtual Private Network
VTN	Virtual Tenant Network

Scan This to Download These Slides

Raj Jain
http://rajjain.com

http://www.cse.wustl.edu/~jain/cse473-20/i_5nlc.htm

Related Modules

CSE 567: The Art of Computer Systems Performance Analysis
https://www.youtube.com/playlist?list=PLjGG94etKypJEKjNAa1n_1X0bWWNyZcof

CSE473S: Introduction to Computer Networks (Fall 2011),
https://www.youtube.com/playlist?list=PLjGG94etKypJWOSPMh8Azcgy5e_10TiDw

CSE 570: Recent Advances in Networking (Spring 2013)
https://www.youtube.com/playlist?list=PLjGG94etKypLHyBN8mOgwJLHD2FFIMGq5

CSE571S: Network Security (Spring 2011),
https://www.youtube.com/playlist?list=PLjGG94etKypKvzfVtutHcPFJXumyyg93u

Video Podcasts of Prof. Raj Jain's Lectures,
https://www.youtube.com/channel/UCN4-5wzNP9-ruOzQMs-8NUw