# Experiential Learning Workshop on Networking

Mar 25, 2017

Dr. Ram P Rustagi
Dept of CSE
PES University
rprustagi@pes.edu

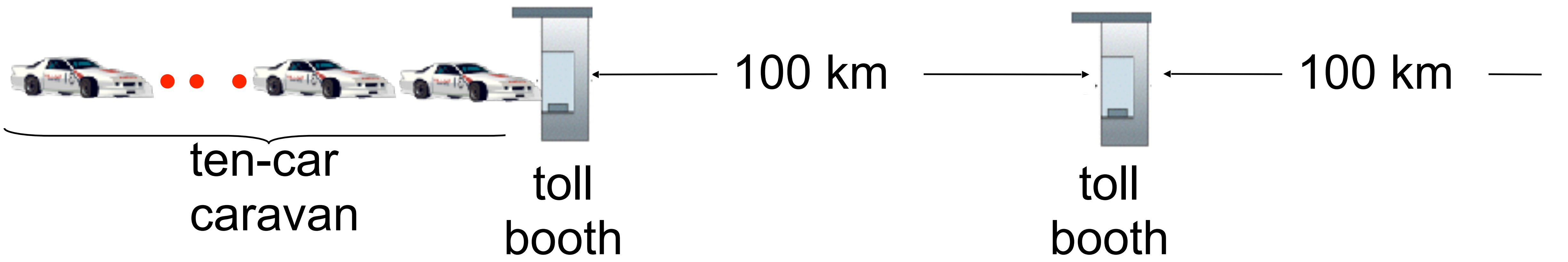
# Reference

- Computer Networks: A Top Down Approach
  - Kurose, Ross
    - Pearson

# Exploration Topics

- Overview of Delays
- Exercise - 1
- Overview of IP
- Exercise - 2
- Misc Content
- Summary

# Caravan analogy




- cars "propagate" at 100 km/hr
- toll booth takes 12 sec to service car (bit transmission time)
- car~bit; caravan ~ packet

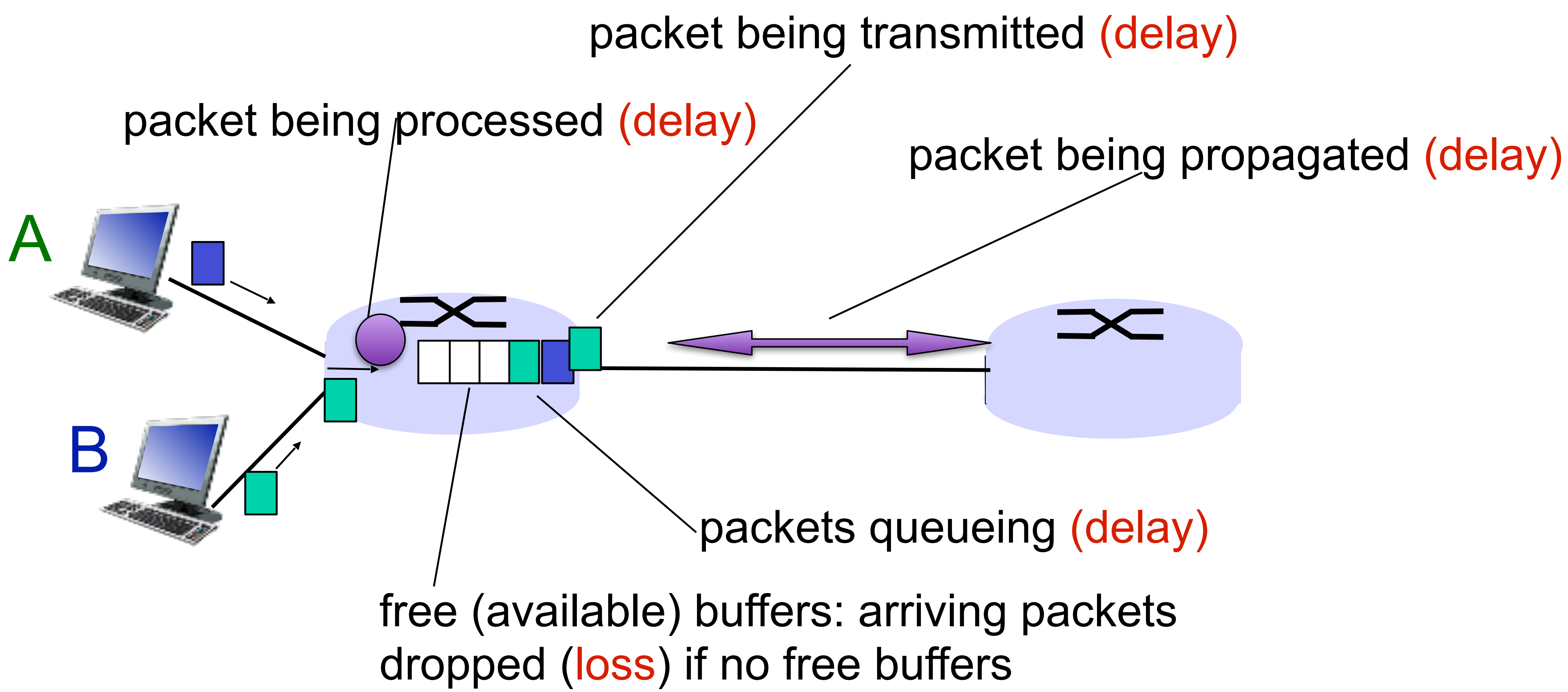
- time to "push" entire caravan through toll booth onto highway = 12*10 = 120 sec
- time for last car to propagate from 1st to 2nd toll both: 100km/(100km/hr)= 1 hr

- *Q:* How long until caravan is lined up before 2nd toll booth?
- *A:* 62 minutes

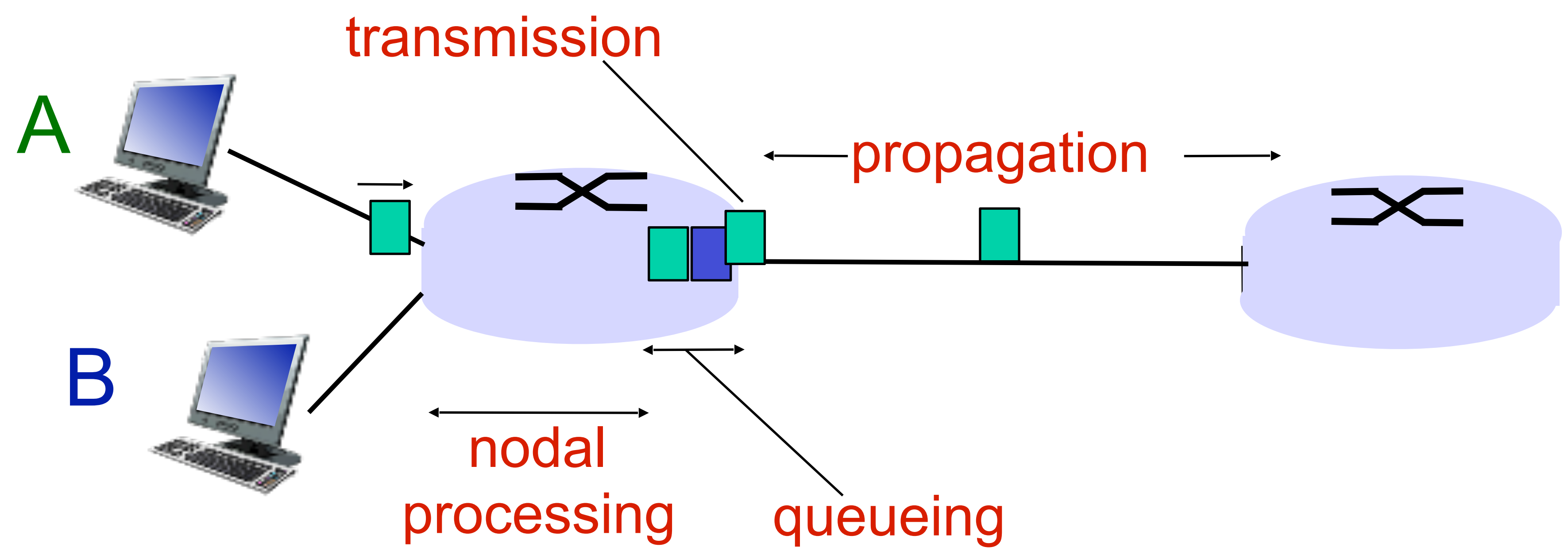
# How do loss and delay occur?

packets *queue* in router buffers

- packet arrival rate to link (temporarily) exceeds output link capacity
- packets queue, wait for turn



# Four sources of packet delay




$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

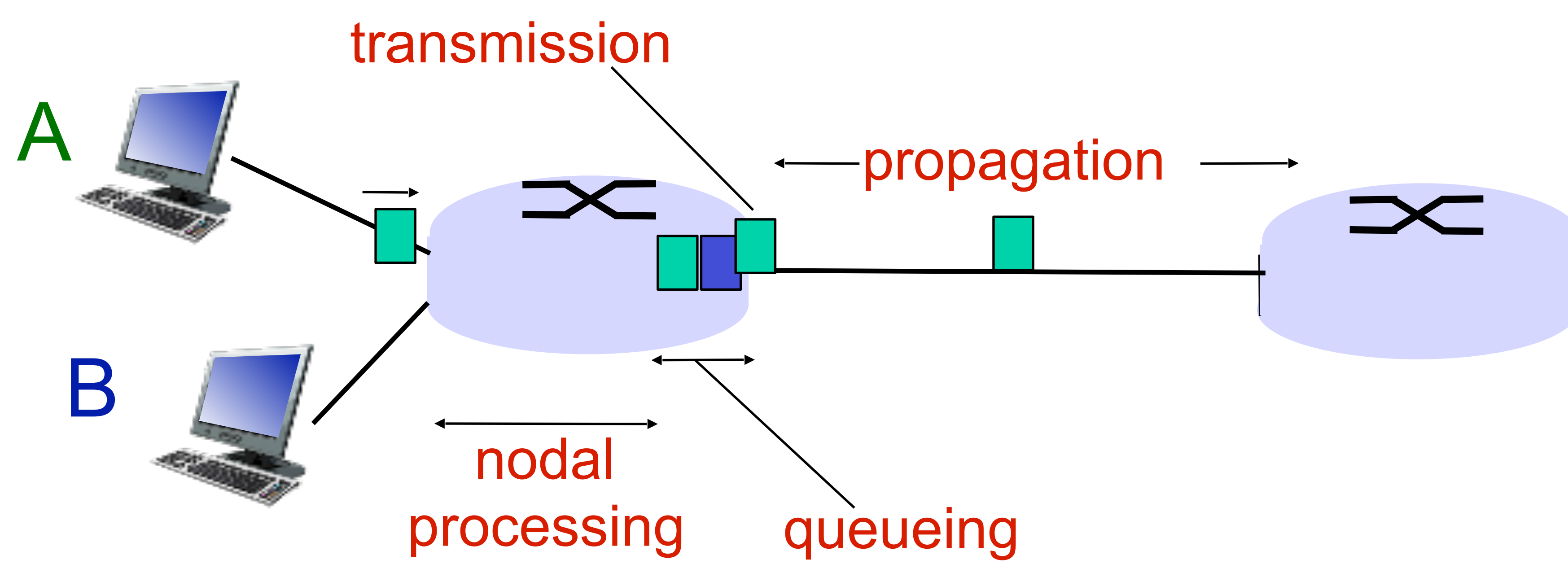
## $d_{\text{proc}}$: nodal processing

- check bit errors
- determine output link
- typically < msec

## $d_{\text{queue}}$: queueing delay

- time waiting at output link for transmission
- depends on congestion level of router

# Four sources of packet delay




$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

**$d_{trans}$: transmission delay:**

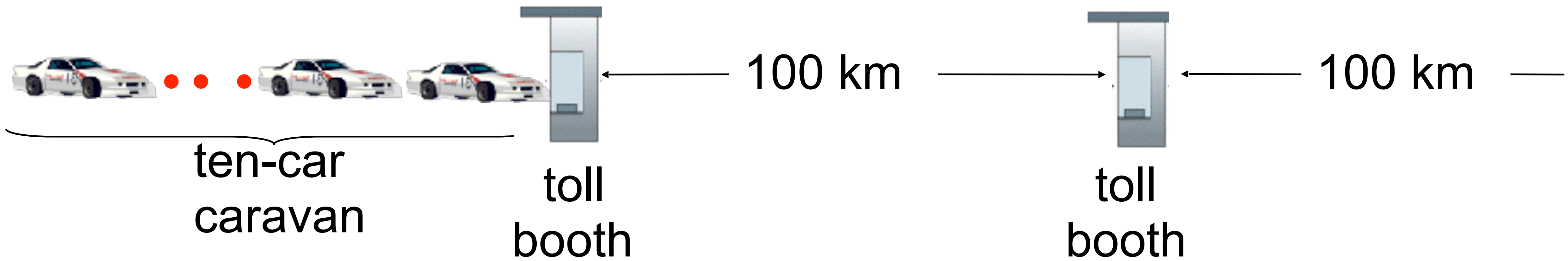
- *L*: packet length (bits)
- *R*: link *bandwidth (bps)*
- $d_{trans} = L/R$

**$d_{prop}$: propagation delay:**

- *d*: length of physical link
- *s*: propagation speed in medium (~2x10$^8$ m/sec)
- $d_{prop} = d/s$

$d_{trans}$ and $d_{prop}$ *very different*

# Caravan analogy (more)



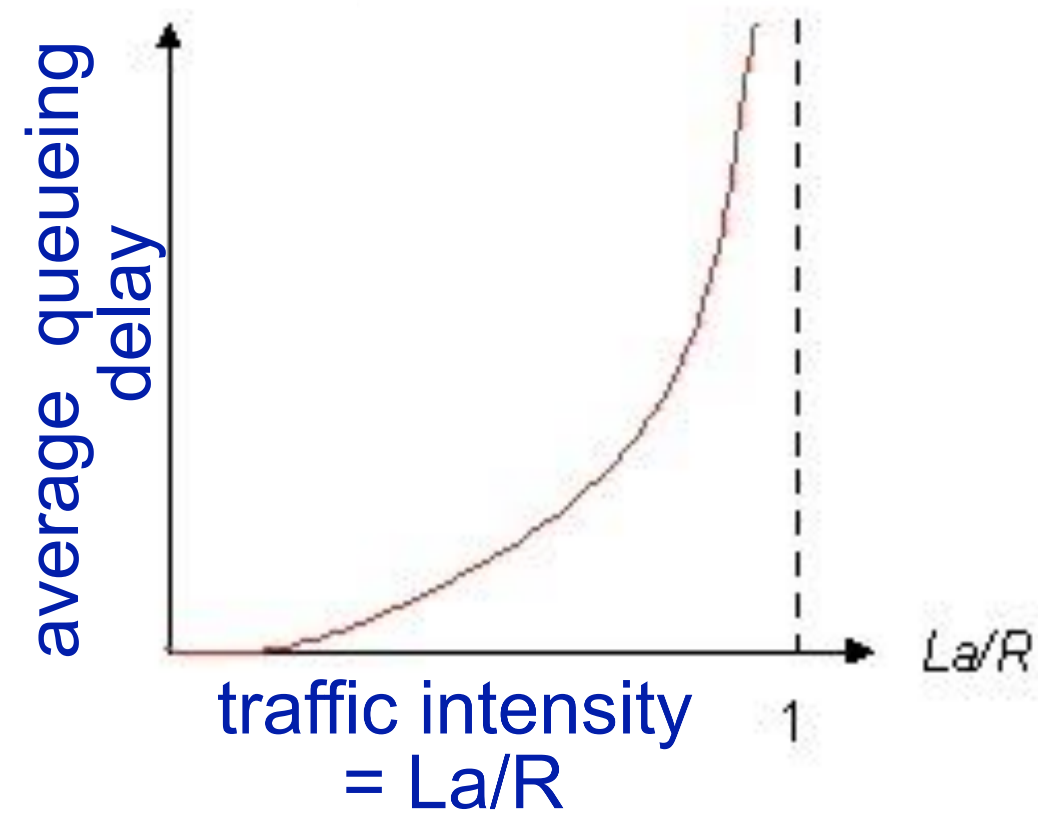

- suppose cars now "propagate" at 1000 km/hr
- and suppose toll booth now takes one min to service a car
- *Q:* Will cars arrive to 2nd booth before all cars serviced at first booth?
- Yes/No ?
- *A: Yes!* after 7 min, 1st car arrives at second booth; three cars still at 1st booth

# Queueing delay (revisited)

- 4 components of delay
  - Processing delay
  - Queuing delay
  - Transmission delay
  - Propagation delay
- Which delay is unpredictable and depends on traffic pattern
  - Periodically: one packet per L/R seconds
  - In bursts but periodically
    - N packets arrive in burst in N(L/R) seconds
  - In general, packets arrival is random
    - Packets are spaced random amount of time

# Queueing delay (revisited)

- *R:* link bandwidth (bps)
- *L:* packet length (bits)
- a: average packet arrival rate




- *La/R* ~ 0: avg. queueing delay small
- *La/R* -> 1: avg. queueing delay large
- *La/R* > 1: more "work" arriving than can be serviced, average delay infinite!


La/R ~ 0


La/R -> 1

* Check out the Java applet for an interactive animation on queuing and loss

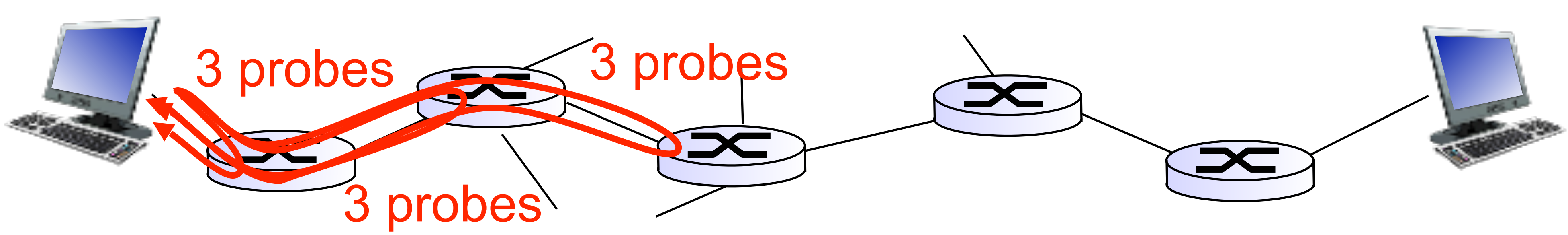
# End to End Delay

- Total delay at all routers from src to dstn
  - Assume N-1 routers (N links)
  - Assume queuing delay to be zero
    - In real life it may not be zero
  - The end to end delay is given by

$$\boxed{d_{\text{end-end}} = N * (d_{\text{proc}} + d_{\text{trans}} + d_{\text{prop}})}$$

# "Real" Internet delays and routes

- what do "real" Internet delay & loss look like?
- `traceroute` program: provides delay measurement from source to router along end-end Internet path towards destination. For all *i:*
  - sends three packets that will reach router *i* on path towards destination
  - router *i* will return packets to sender
  - sender times interval between transmission and reply.

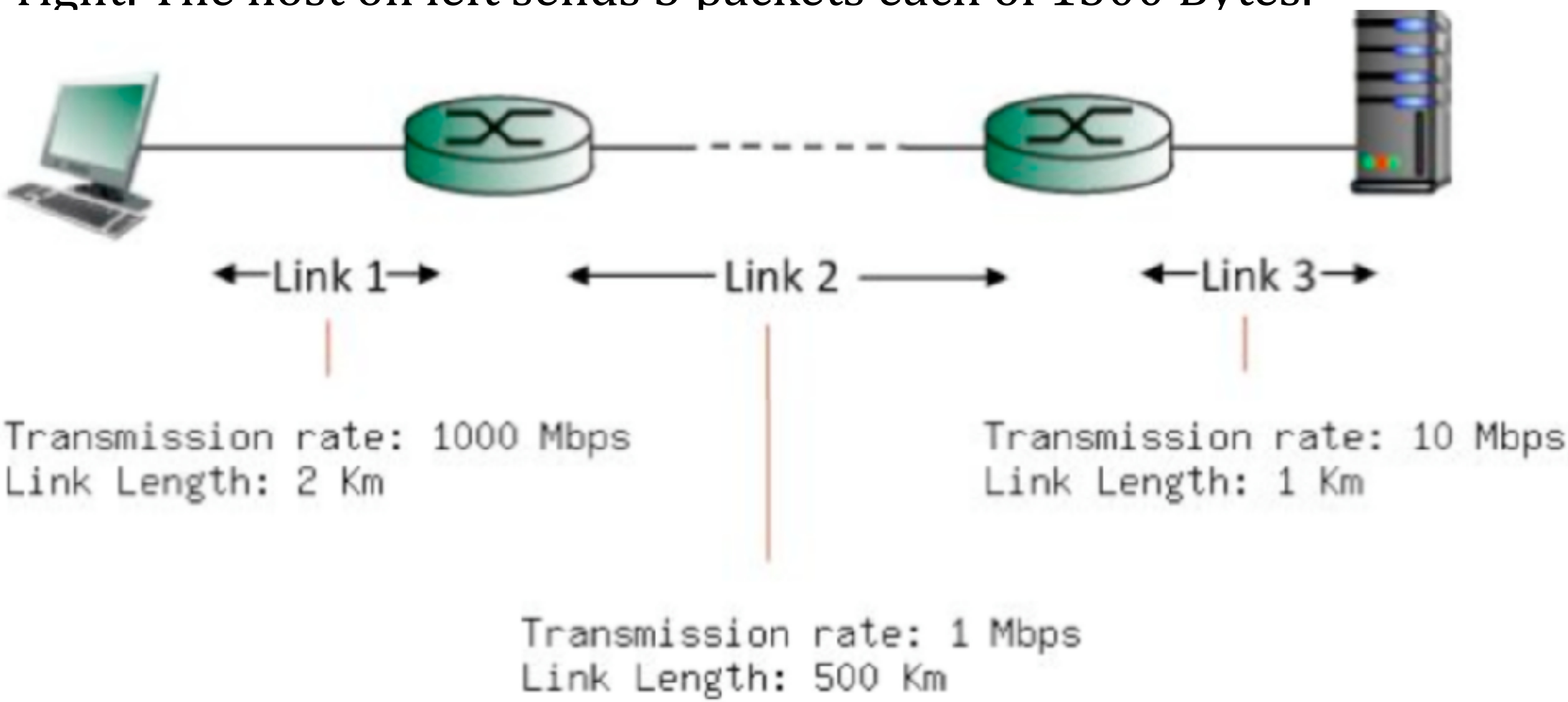


# Summary

- delay

# Practice Quiz

Consider the following figure. Find the end-to-end delay (including the transmission delays, propagation and queuing delays, but ignoring the processing delays) from when the left host begins transmitting the first bit of a packet to the time when the last bit of last packet is received at the server at the right. The host on left sends 3 packets each of 1500 Bytes.



# Practice Quiz - Answer

Propagation delay on link 1: 2*1000*1000/(2x10**8)=0.01ms
Propagation delay on link 2: 500*1000*1000(2x10**8)=2.5ms
Propagation delay on link 3: 1*1000*1000/(2x10**8)=0.005ms
Total propagation delay = 0.01 + 2.5 + 0.005 = 2.515ms

Packet size in bits: 1500x8=12000
Transmission delay for one pkt on link 1 : 12000/(1000x10**6) = 12us = 0.012ms

Transmission delay on one pkt on link 2 : 12000/(1x10**6) = 12000us = 12ms

Transmission delay on one pkt on link 3 : 12000/(10x10**6) = 1200us = 1.2ms.

Switch 1 will start transmission of first packet after 0.012ms,

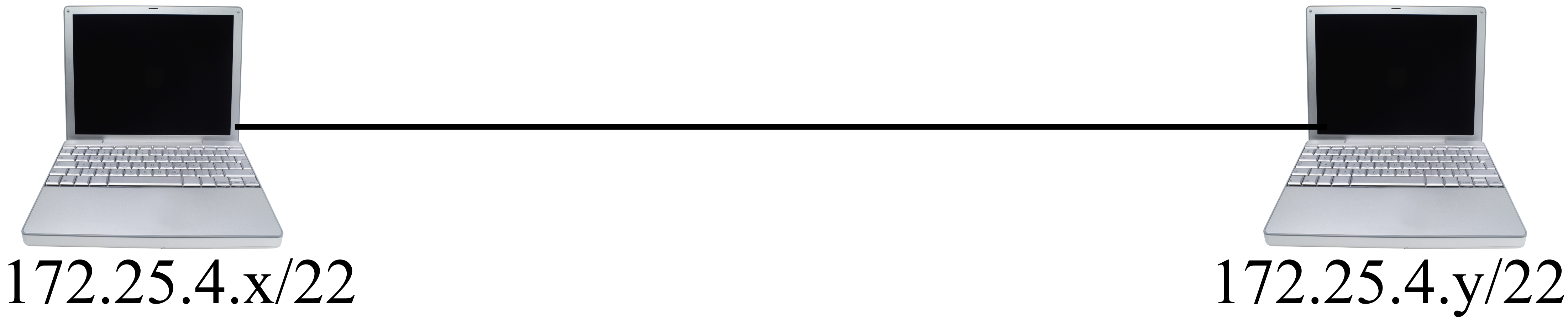
2nd packet after 12.012ms, and

3rd packet after 24.012ms

# Exploration Topics

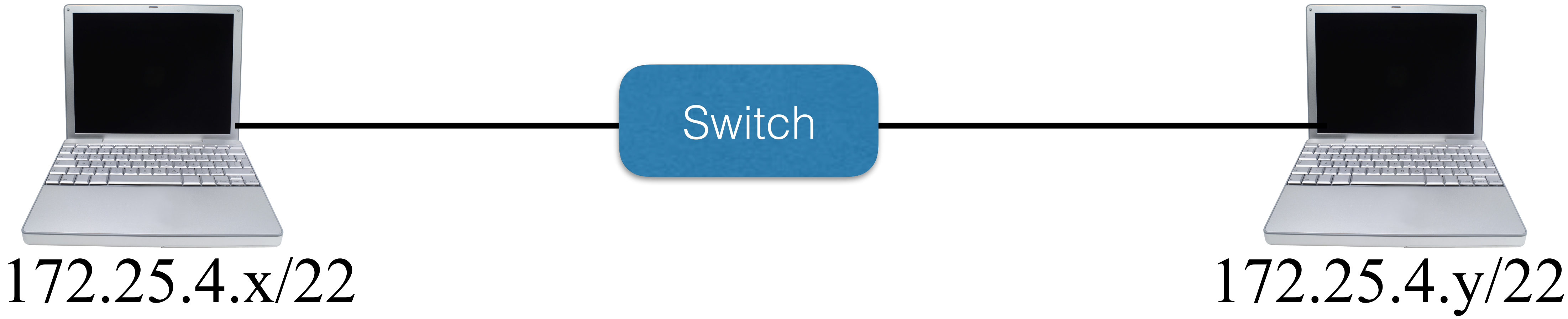
- Overview of Delays
- Exercise 1 - Delays
- Overview of IP
- Exercise - 2
- Misc Content
- Summary

# Transmission Delay: Case 1



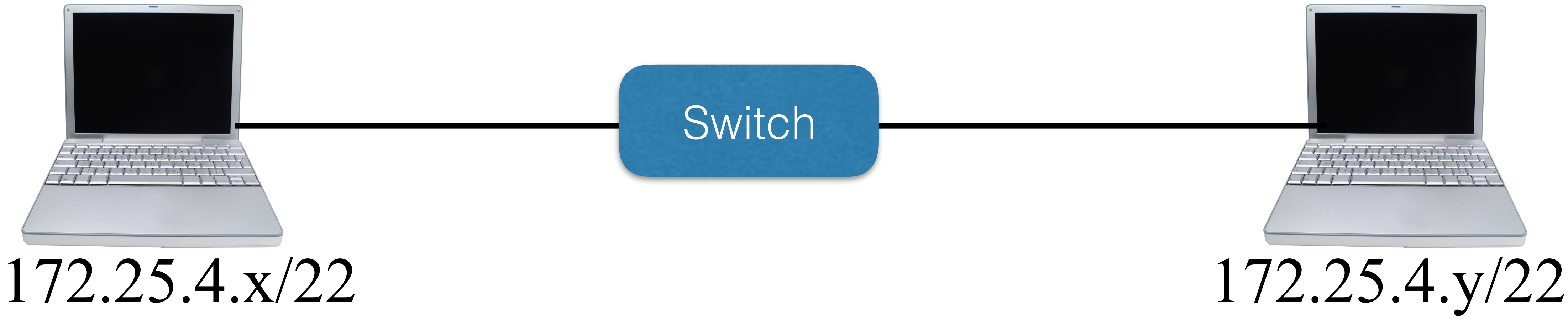

- Change packet size from 200 bytes in increments of 200 bytes i.e. 200, 400, 600, 800, 1000Bytes
- Note the delays in ping response
- Note the difference in delay response
- Difference will be transmission delay for 200 bytes

# Transmission Delay: Case 2



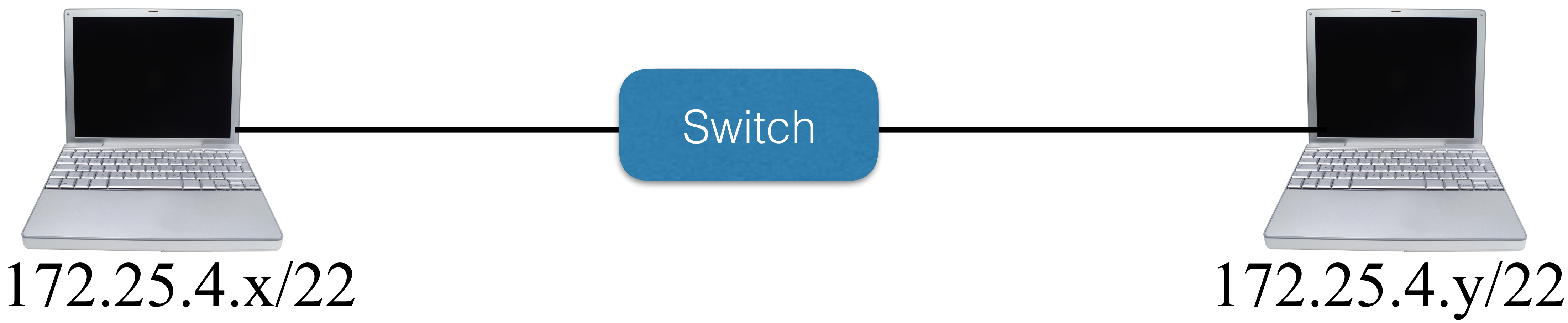

- Change packet size from 200 bytes in increments of 200 bytes i.e. 200, 400, 600, 800, 1000Bytes
- Note the delays in ping response
- Compare the delays from case 1
- Difference will be transmission delay for 2 links

# Processing Delay: Case 3



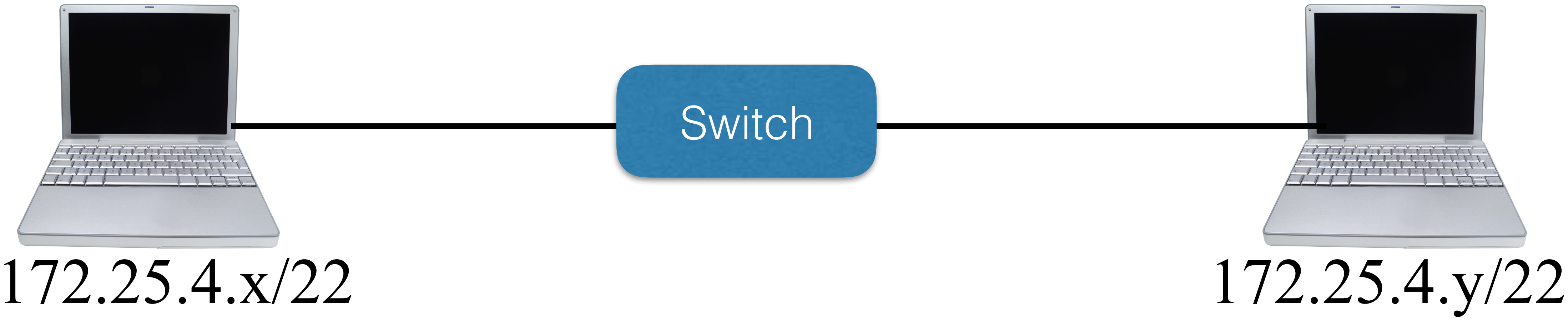

- Run a server program (server.py)
  - It mimics processing (sleep) delay of N seconds
- Run the client program (client.py)
- Note the delays in response
- Primarily it is processing delay

# Queuing Delay: Case 4




- Run a server program (server.py)
  - It mimics processing (sleep) delay of N seconds
- Run the multiple invocations of client program.
- Note the delays in response
  - Each client will have different response time.
  - Corresponds to queueing delay

# Addressing Queuing Delay: Case 5




- Run a server program (serverQ.py)
  - It mimics processing (sleep) delay of N seconds
  - Run multiple instances of server program
- Run the multiple invocations of client program.
- Note the delays in response
  - Each client will talk to a different server.
  - No queueing delay

# Exploration Topics

- Overview of Internet
- Overview of Setup
- Overview of OSI Layers
- Overview of Tools
- Understanding Wireshark
- Exercise - 1
- Overview of IP
- Exercise - 2
- Misc Content
- Summary

# IP Overview

- IPv4 addresses are unique and universal
  – exceptions ?
- IPv4 address is 32 bit long
  – total available addresses: **4,294,967,296**
- Uses Dotted Decimal Notation (DDN)
  – example: 119.82.126.182
- Exercise:
  – Find the error in following addresses
    - 119.082.126.182
    - 119.82.126.182.80
    - 119.82.126.282
    - 119.01010010.126.82

# IP Addressing

- Address types
  - Unicast
  - Multicast
  - Broadcast
  - Anycast
- Classful addressing:
  - first byte value determines the class
  - Class A, B, C, D, & E
- Large part of address space is wasted

# IP subnets

- Identified by subnet masks : a.b.c.d/n
- A router is needed to connect two networks
- Masks for classful addresses
  - Class A: `255.0.0.0` or /8
  - Class B: 255.255.0.0 or /16
  - Class C: 255.255.255.0 or /24
- Classful addressing obsolete now
  - replaced with classless addressing (CIDR)
- RFCs
  - RFC 1518: Architecture for IP addr allocation with CIDR
  - RFC 1466: Guidelines for IP addr space management
  - RFC 917: Internet subnets

# Subnets

- Few terms to understand
  - network portion and host portion
  - network number
    - apply subnet mask to IP address (bitwise AND)
  - Broadcast address
    - set all bits to 1 in host portion
  - network mask
    - set all bits to 0 in host portion
  - first available address in the block
    - value of host portion = 1
  - last available address in the block
    - value of host portion = $2^n-2$

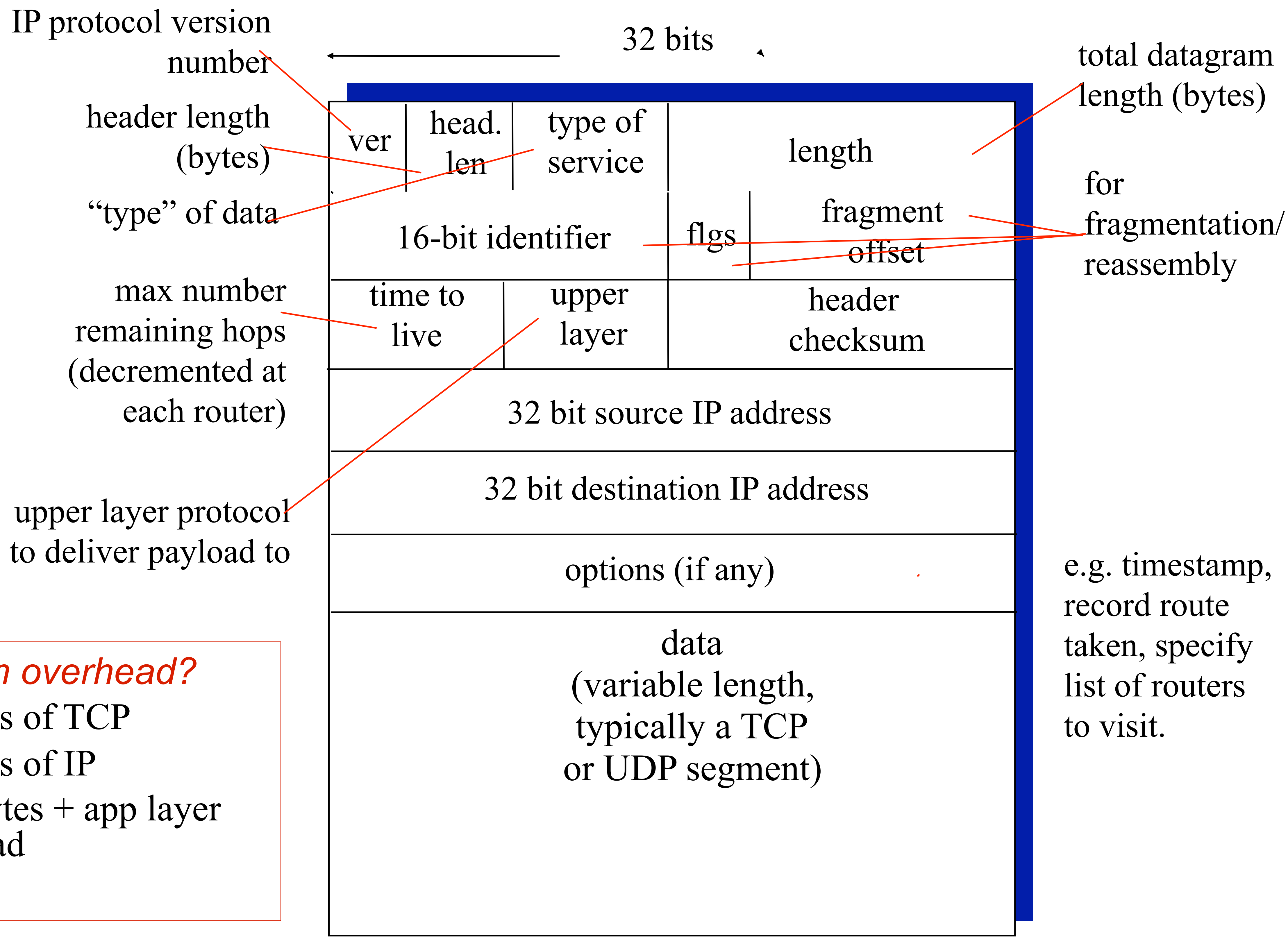
# IP Subnets

- Exercise 1
  - a block of addresses is granted to a small organization. one of the address is `119.82.126.182/27`. Find out the following:
    - the network number
    - subnet mask
    - broadcast address
    - first & last available address
- Exercise 2:
  - repeat the above exercise for address
    - `192.168.100.200/18`

# IP Packet Format

- Header + Data
- Header
  - fixed header - 20 bytes
    - src IP, dst IP, TTL, Hlen, Pkt Len
  - options
    - generally not used
    - record route, source route, timestamp
- data

# IP Packet Format




*how much overhead?*

- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead

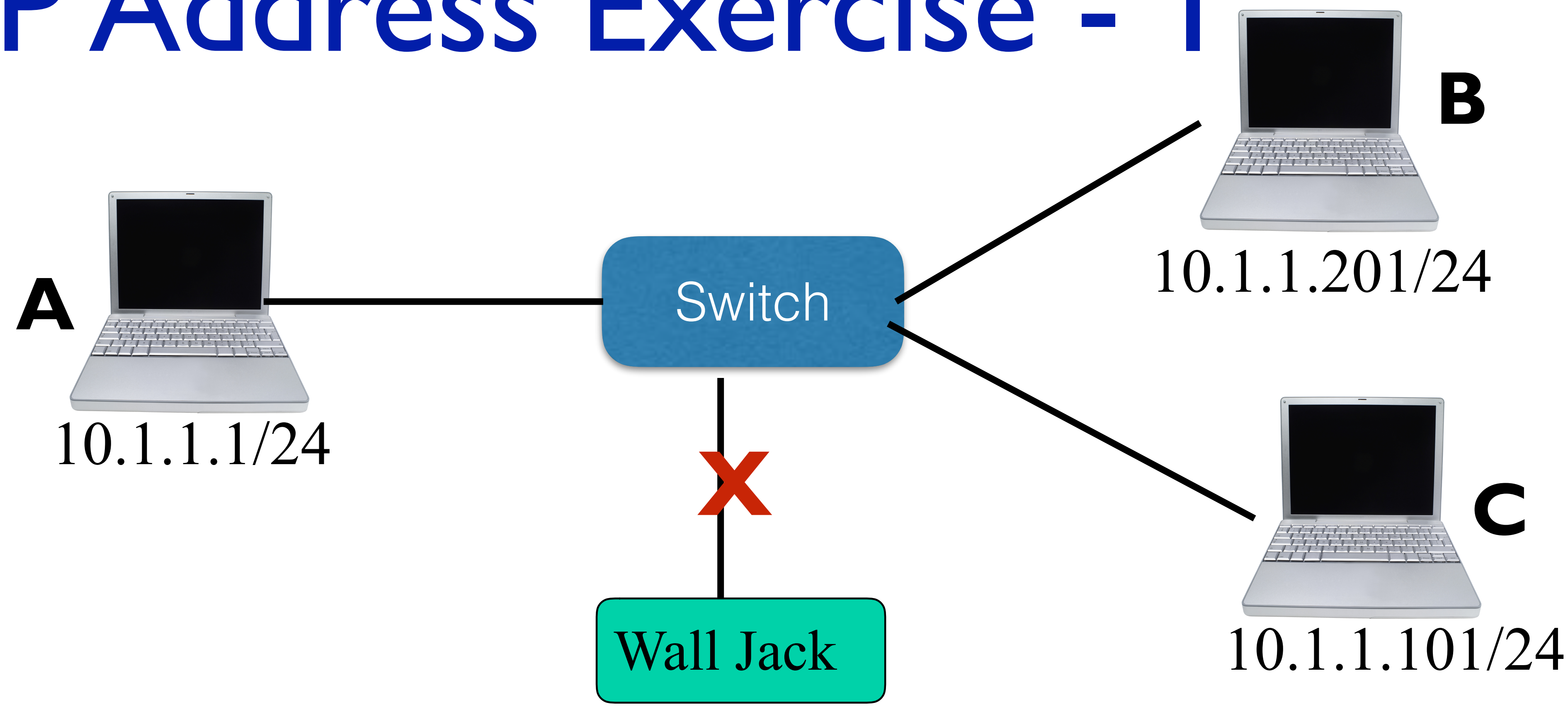
# Exploration Topics

- Overview of Internet
- Overview of Setup
- Overview of OSI Layers
- Overview of Tools
- Understanding Wireshark
- Exercise - 1
- Overview of IP
- Exercise - 2
- Misc Content
- Summary

# IP Addressing

- Assign IP Address to your machine
- Ping your neighbours
- Change your subnet
- See reachability
- Analyze IP packet header
- Change TTL
- Change default route.
- Access internet
- Assign ARP mapping
- access other hosts

# IP Address Exercise - 1




- Assign the address manually.
- Check reachability among each other
- A, B & C should be reachable from other.
- Change the subnet mask from /24 to /26.
- Check reachability? What happens

# IP Address Exercise - 1

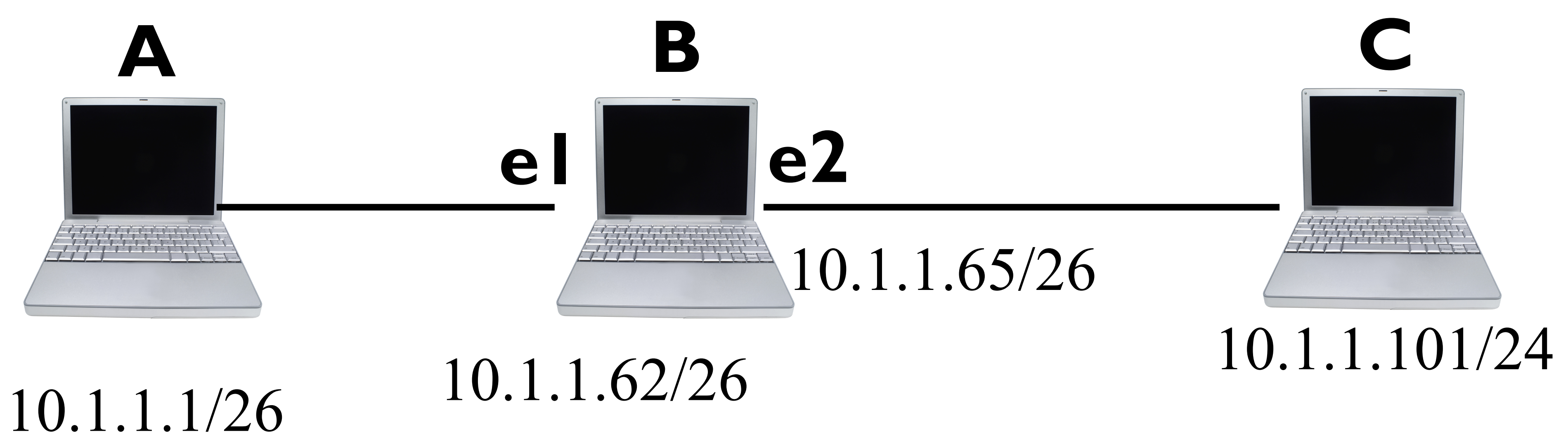
- Understanding of subnetting.
- The mask of /26 makes the following
  - 10.1.1.1/26 : Network 1
    - 10.1.1.1 - 10.1.1.62
  - 10.1.1.101/26 : Network 2
    - 10.1.1.65 - 10.1.1.126
  - 10.1.1.201/26 : Network 3
    - 10.1.1.193 - 10.1.1.254

# IP Address Exercise - 2 (Subnetting)

- Understanding subnetting
- Change IP address of B to following
  - 10.1.1.62/26
    - A and B should be able to reach each other
  - 10.1.1.193/26
    - B and C should be able to reach each other
- Look at ARP table of your system
  - $ arp -an

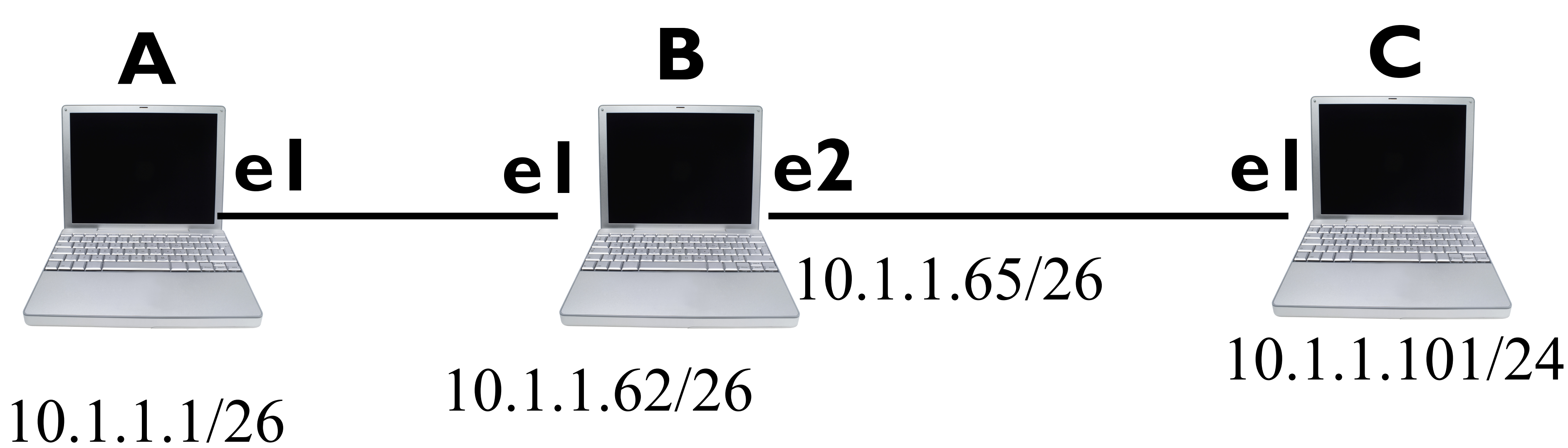
# IP Address Exercise - 3 (Routing)

- Connect 3 systems in following way
- Check connectivity
  - A can reach B (e1)
  - C can reach B (e2)
  - A can not reach C and vice versa
    - Requires routing



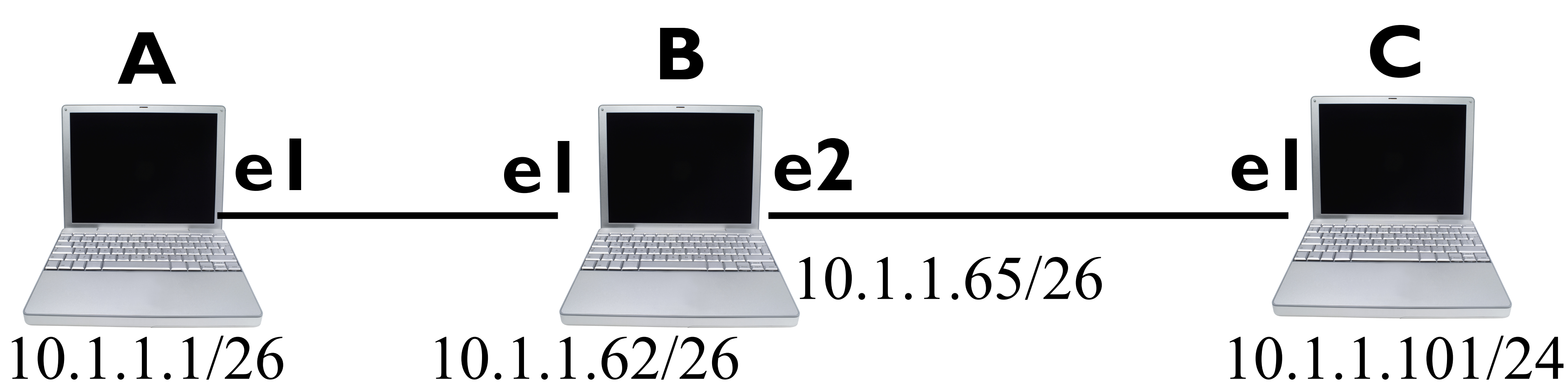
# IP Address Exercise - 4 (Routing)

- Connect 3 systems in following way
- Convert B into a router
  - `sudo sysctl -w net.ipv4.ip_forward=1`
- On A
  - sudo ip route add 10.1.1.64/26 dev <e1> #enp2s0
- On C
  - sudo ip route add 10.1.1.0/24 dev <e1> #enp2s0



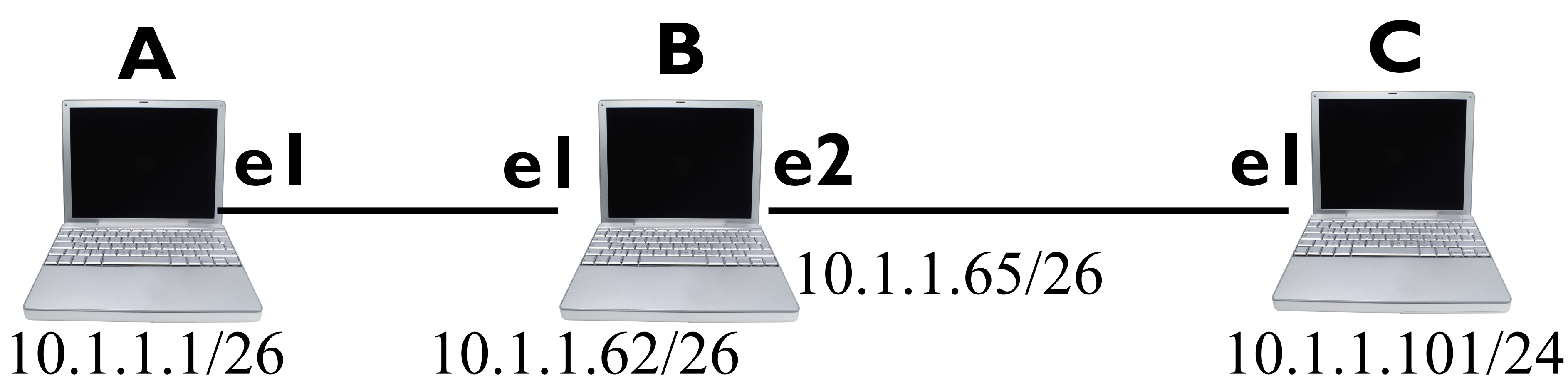
# IP Address Exercise - 5 (Routing)

- Add additional IP addresses to A, B & C as follows.
  - A : 10.2.1.1/24
    - sudo ip addr add 10.2.1.1/24 dev <e1>
  - B(e1) : 10.2.1.201/24
  - B(e2): 10.3.1.1/24
  - C : 10.3.1.201/24
- Define routing check reachability of A & C



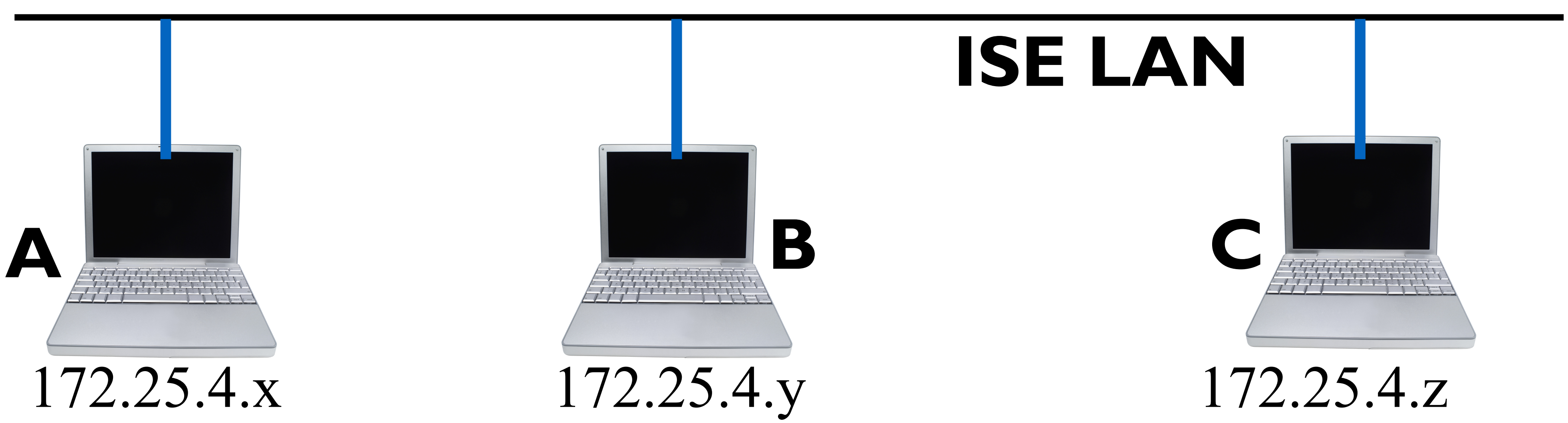
# IP Address Exercise - 5 (Snooping)

- Add additional IP addresses to A, B & C as follows.
  - A : 10.2.1.1/24
    - sudo ip addr add 10.2.1.1/24 dev <e1>
  - B(e1) : 10.2.1.201/24
  - B(e2): 10.3.1.1/24
  - C : 10.3.1.201/24
- Define routing to check reachability of A & C
  - Capture traffic at B (snoop on A-C chat)



# Exercise - 6 (MITM)

- Restore connectivity to as before in the lab.
- Should get IP Address 172.25.4.x/22
- Note down your default router
  - $ip route show
- Make a group of 3 team
- Objective:: When A & C communicate, B can snoop
- Use ARP Spoofing to fool A & C go via B



# Exercise - 6 MITM Contd.

- Convert B into a router
  - `sudo sysctl -w net.ipv4.ip_forward=1`
- Insstall ARP Sniffer on B
  - `sudo apt-get install dsniff`
- Issue ARP Spoof command on B for A & C
  - `arpspoof -i <e1> -t 172.25.4.x -r 172.25.4.z`
- Run wireshark on B for IP address A & C
  - capture filter: host 172.25.4.x or host 172.25.4.z
- Let A & C chat
- Read the communication in wireshark on B.

# Thank You