ECE/COMPSCI 356 Computer Network Architecture

Lecture 14: Midterm review

Neil Gong

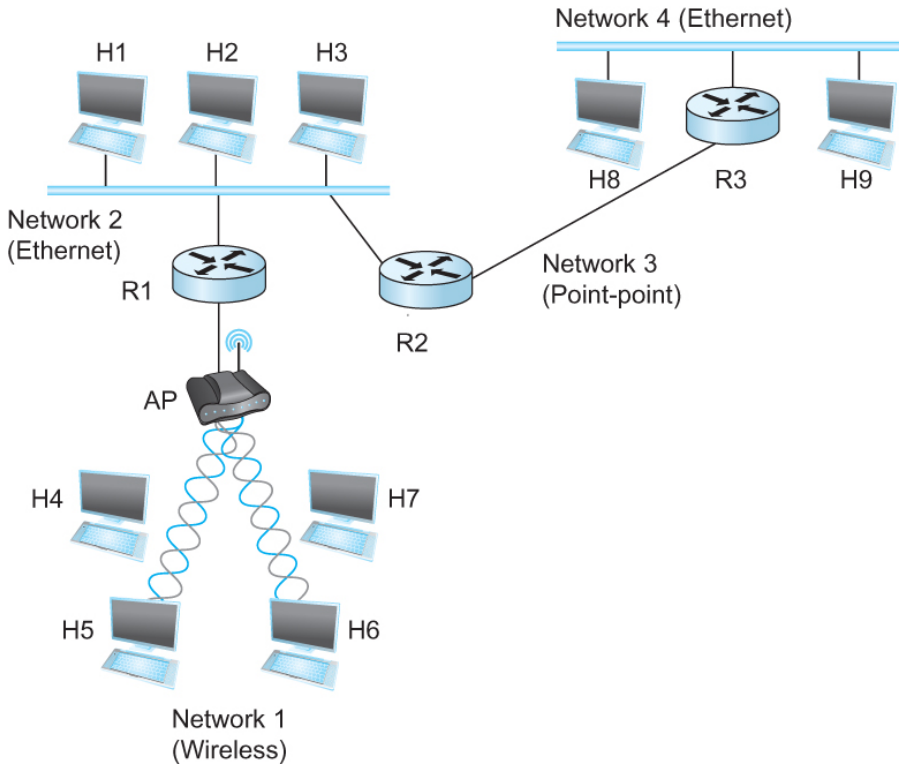
neil.gong@duke.edu

Slides credit: Xiaowei Yang, PD

Midterm

- Time
  - 03/06 W, 10:05 – 11:20am
- Closed-book
- A note sheet (double-sided US letter) is allowed
- Calculator is allowed
- Covering all lectures so far

An example



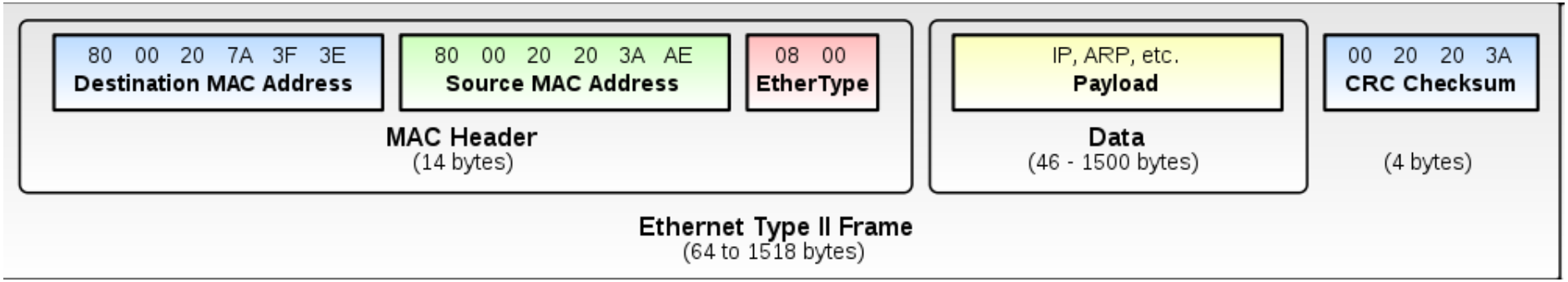
What happens when H9 joins the network and sends an IP packet to H1?

Joining the network

- Configure H9 via DHCP
  - IP address
    - Private IP address when NAT is used
    - NAT keeps a table mapping private to public addresses
  - Default router
    - R3
      - In H9's forwarding table, R3 is the NextHop for all IP packets

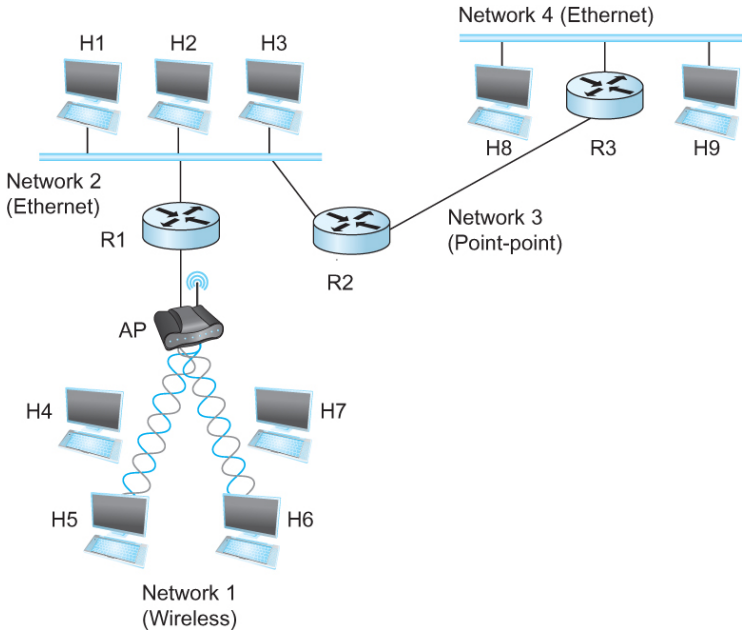
Sending IP packet to H1

- H9 checks its forwarding table
  - NextHop is R3
- H9 and R3 are connected via Ethernet, a multi-access link
- Encapsulate the IP packet into an Ethernet frame
  - If the IP packet is too large, then fragment it to multiple IP packets
    - May use MTU discovery to find the minimum MTU
  - Adding Ethernet frame header and trailer to each IP packet
  - Sending frame to R3


80 00 20 7A 3F 3E
Destination MAC Address
80 00 20 20 3A AE
Source MAC Address
08 00
EtherType
MAC Header
(14 bytes)
IP, ARP, etc.
Payload
Data
(46 - 1500 bytes)
00 20 20 3A
CRC Checksum
(4 bytes)
Ethernet Type II Frame
(64 to 1518 bytes)

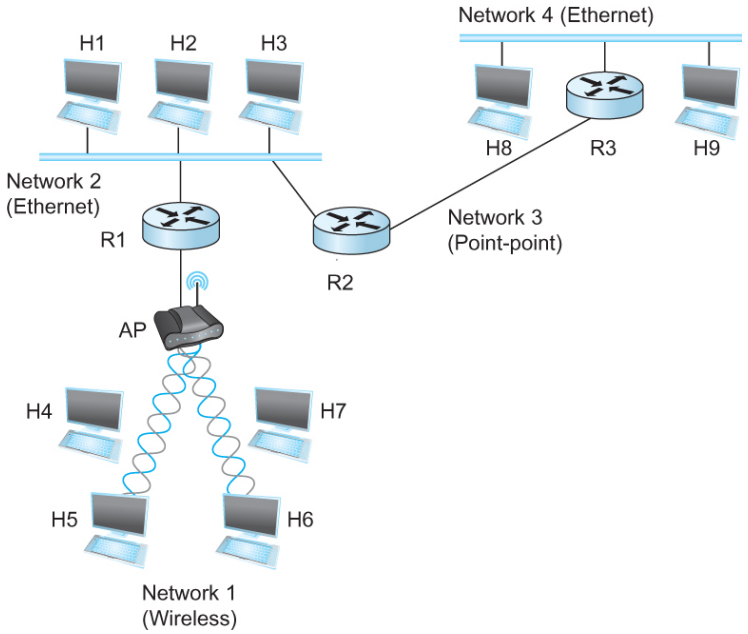
R3

- R3 receives Ethernet frame from H9
  - Checks CRC
  - Removes header and trailer
- R3 checks its forwarding table
  - NextHop is R2
- Where is forwarding table from?
  - Routing protocol
    - Intradomain routing: RIP and OSPF
    - Interdomain routing: BGP
- Send the IP packet to R2
  - Via point-to-point link
  - Add header and/or trailer depending on the protocol
  - Send the frame to R2



R2

- R2 receives frame from R3
  - Error detection
  - Removes header and trailer
- R2 checks its forwarding table
  - H1 is in the same network
- Encapsulate IP packet to Ethernet frame
  - If MAC address of H1 is unknown, use ARP to map IP to MAC address
  - Send Ethernet frame to H1



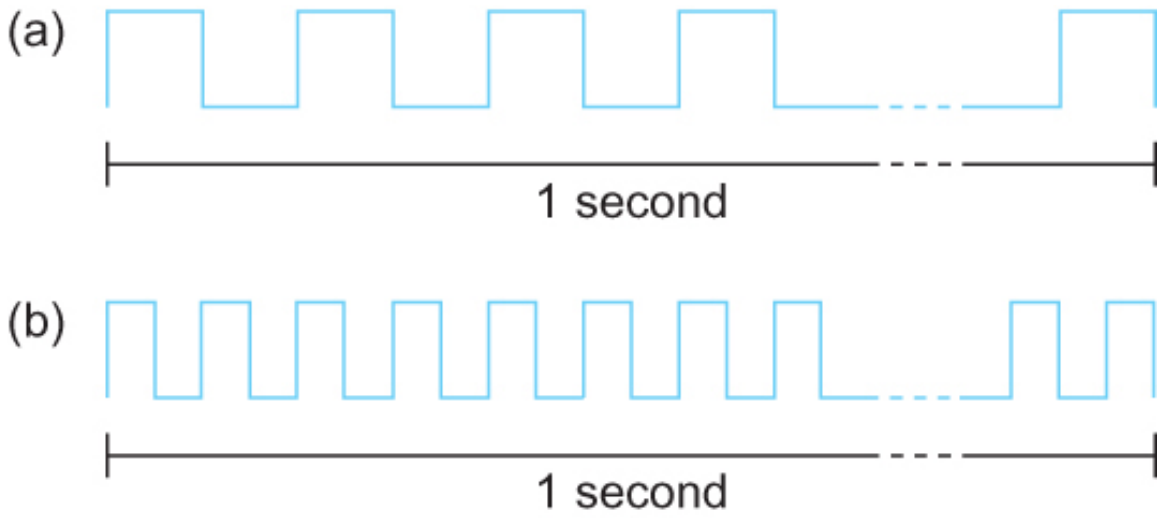
Link layer functions

- Transmit a frame between two nodes in point-to-point link or multi-access link
- Link layer functions
  - Encoding
    - NRZ, NRZI, Manchester, 4B/5B
  - Framing
    - Byte-oriented, bit-oriented, clock-based
  - Error detection
    - CRC, Checksum
  - Reliable transmission
    - stop-and-wait, sliding window
    - May also be used at higher layers
      - TCP and application layer

Physical layer

- Data are transmitted as electromagnetic waves in some medium
- Performance metrics
  - Bandwidth
  - Delay
  - Delay x bandwidth
  - Transfer time
  - Throughput

Bandwidth

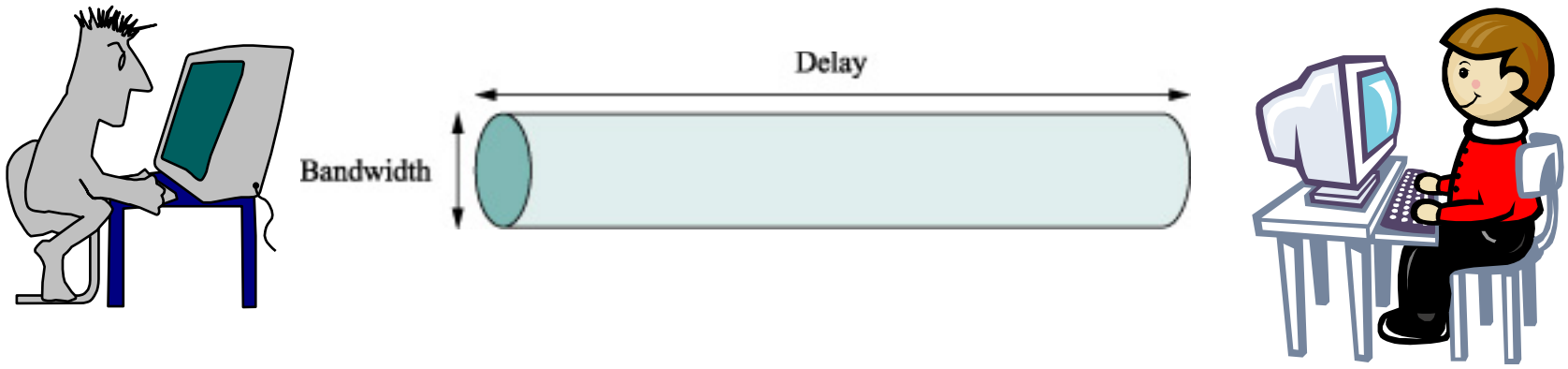


Number of bits that can be transmitted per second

Delay

- How long does it take for one bit to travel from one end of link to the other?
- Length Of Link / Speed Of WaveInMedium
- Round-trip time (RTT): 2 x delay

Delay x bandwidth product



- We think the link between two nodes as a hollow pipe
- Delay length of the pipe and bandwidth the width of the pipe
- Volume of the pipe: #bits the sender sends before the receiver receives the first bit
- When the pipe is full, no more bits can be pumped into it

Transfer time and Throughput

- Transfer time = delay+ transmission time + queuing time

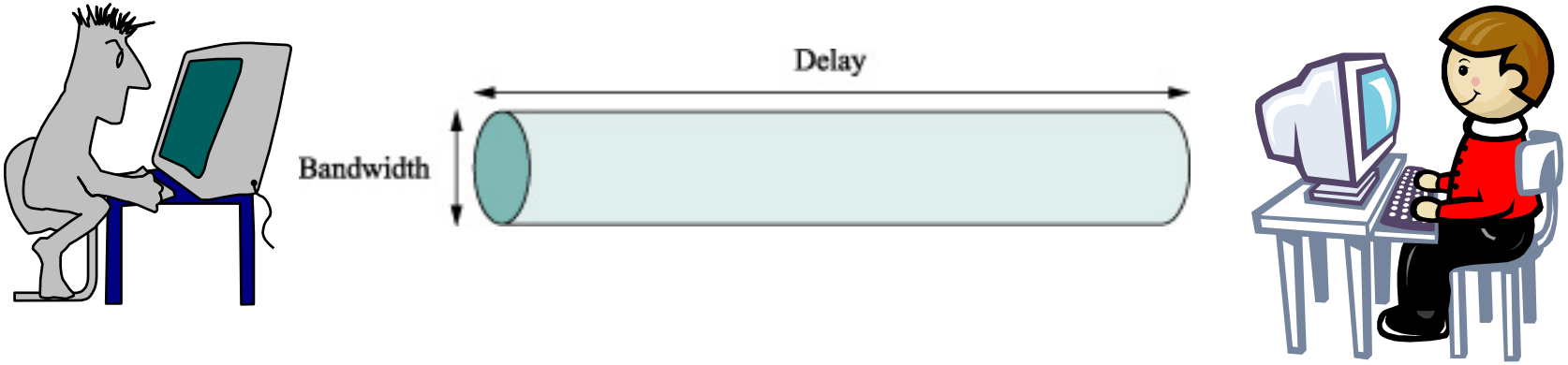
  transmission time = Transfer Size/bandwidth

- Transfer time: also called latency/delay

- Throughput = Transfer Size/Transfer Time

Example questions

Transfer time



- Bandwidth:1Mbps
- One-way delay:100ms
- Data:1MB
- Delay * Bandwidth = 100Kb
- 1MB/100Kb = 80 pipes of data
- Transfer time=80 * 100ms + 100ms = 8.1s

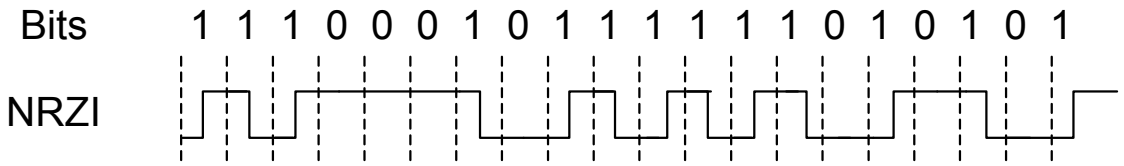
Encoding

2. (5 pts) Show the 4B/5B encoding, and the resulting NRZI signal, for the following bit sequence:

1110 0101 0000 0011

11100 01011 11110 10101

The corresponding NRZI sequence is:



Questions may also be related to bit stuffing, etc.

CRC Error Detection

5. (10 pts) Suppose we want to transmit the message 11001111 and protect it from errors using the CRC polynomial x^3+1.

(a) Use polynomial division to determine the message that should be transmitted.

(b) Suppose the leftmost bit of the message is inverted due to noise on the transmission link. What is the result of receiver's CRC calculations? How does the receiver know that an error has occurred?

(a) We take the message 11001111, append 000 to it, and divide by 1001 according to the method shown in Section 2.4.3. The remainder is 101; what we transmit is the original message with this remainder appended, or 11001111101.

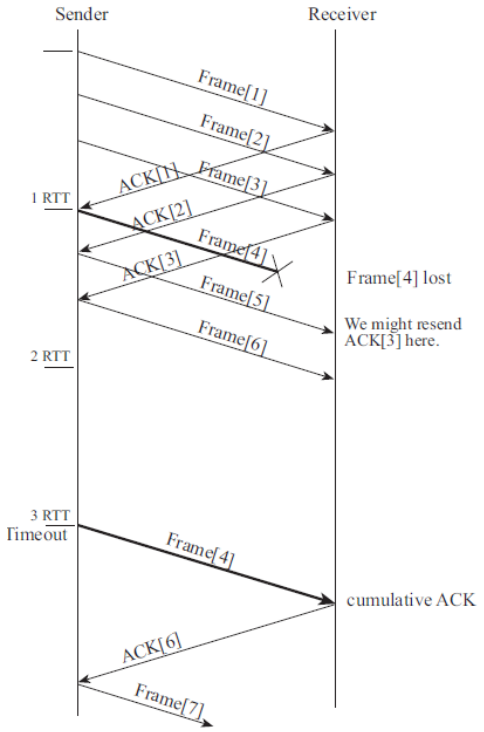
(b) Inverting the first bit of the transmission gives 01001111101; dividing by 1001 ($x^3 + 1$) gives a remainder of 10; the fact that the remainder is nonzero tells us a bit error occurred.

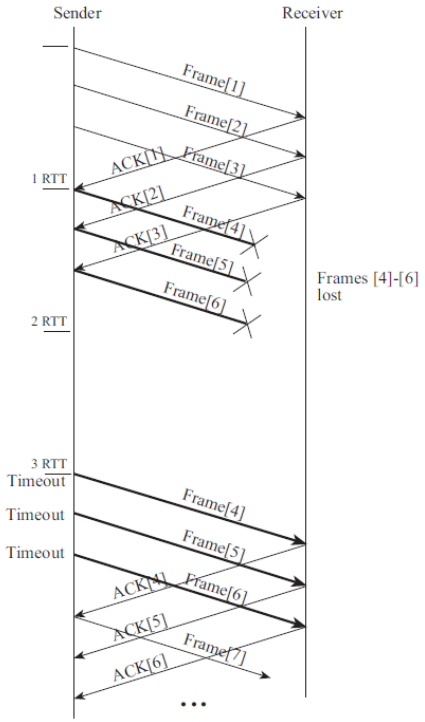
Reliable Transmission via Sliding Window

7. (10 pts) Draw a timeline diagram for the sliding window algorithm with SWS = RWS = 3 frames. Assume frame 4 is lost. Use a timeout of approximately 2 x RTT. Frame index starts from 1.

(a) Frame 4 is lost

(b) Frame 4, 5, 6 are lost



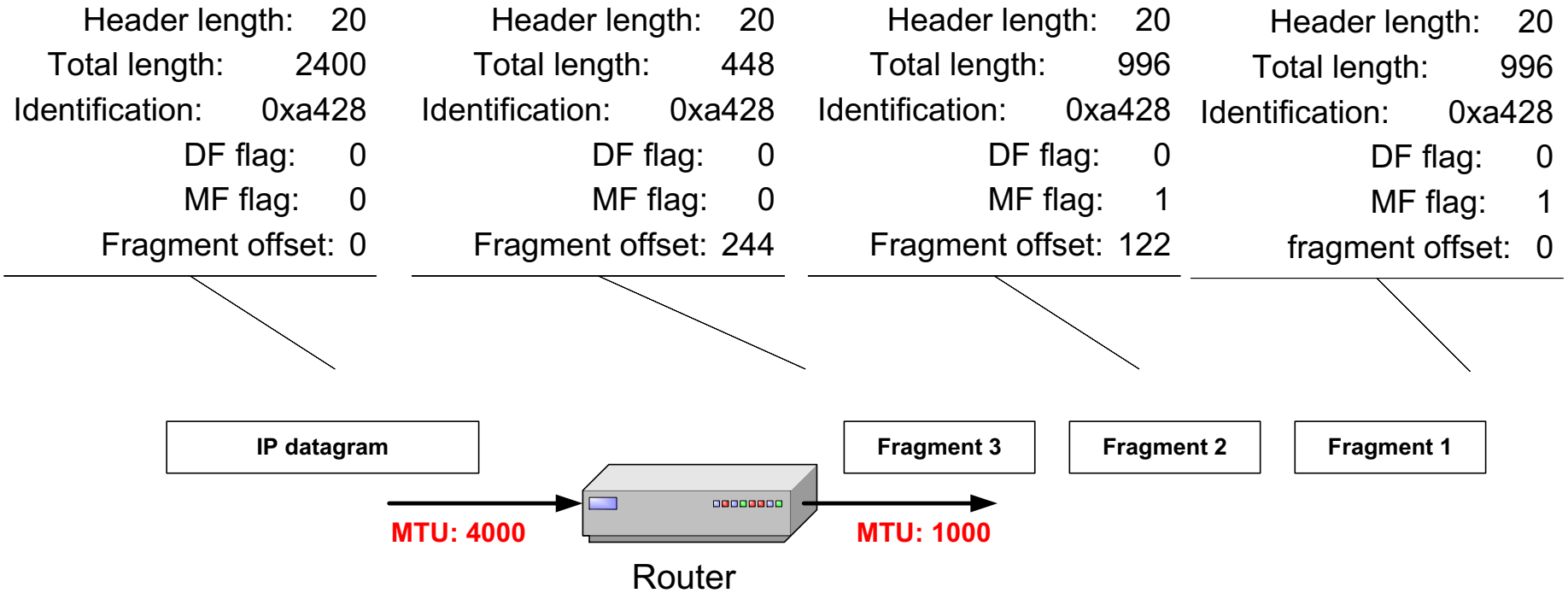


IP layer

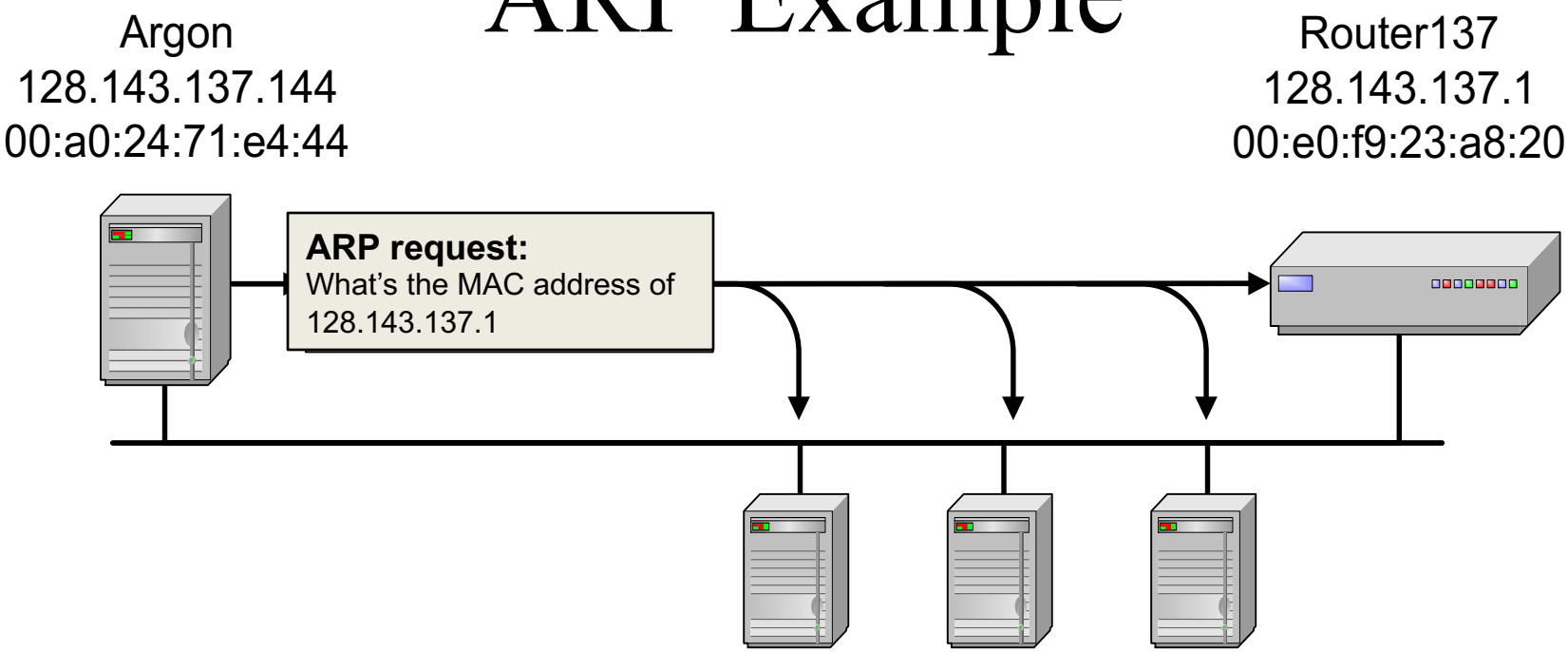
- Header format
- Fragmentation
- ARP
- ICMP and its applications
- Routing
  - Intradomain routing: RIP, OSPF
  - Interdomain routing: BGP
- DHCP, NAT, IPv6, IP tunnels

IP Packet Fragmentation

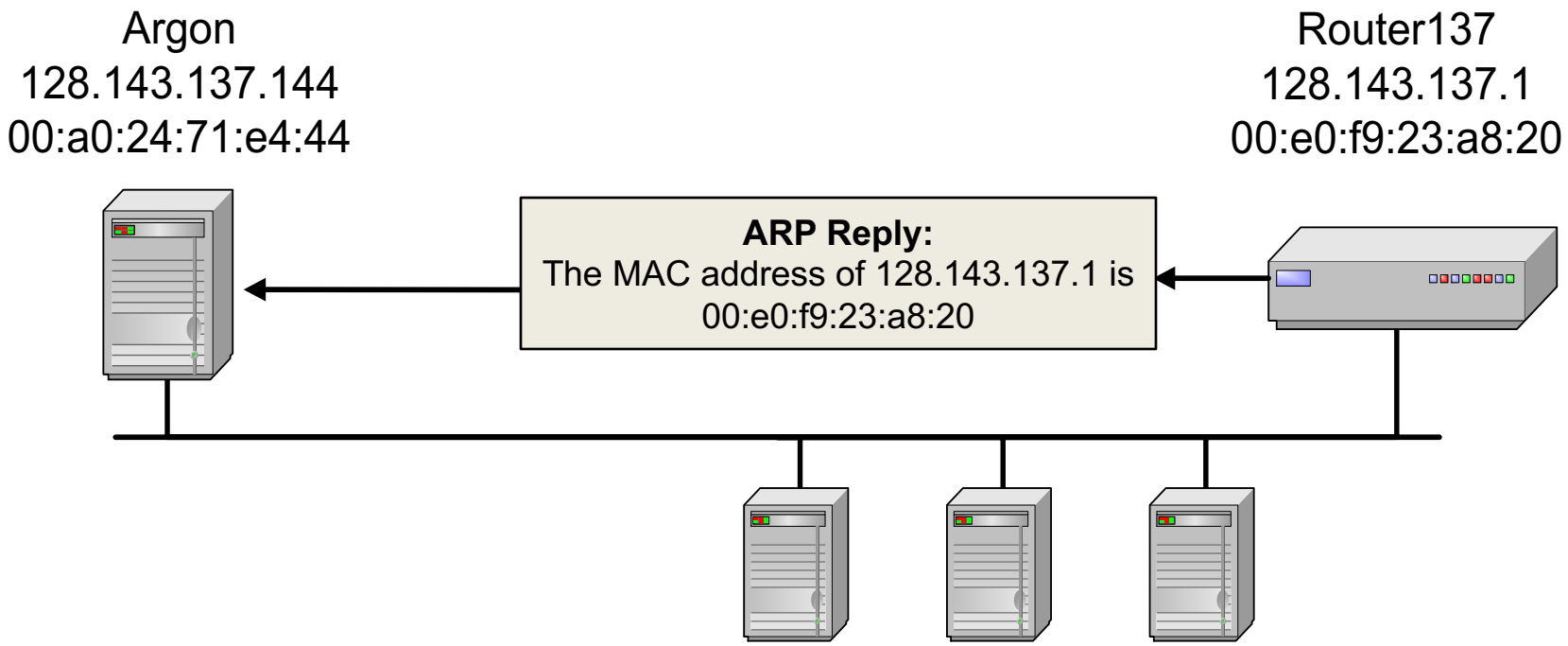
- A datagram with size 2400 bytes must be fragmented according to an MTU limit of 1000 bytes



ARP Example



- *ARP Request from Argon is broadcasted:*
  - Source addr in Ethernet header: 00:a0:24:71:e4:44
  - Destination addr in Ethernet header: FF:FF:FF:FF:FF:FF
- Source hardware address: 00:a0:24:71:e4:44
- Source protocol address: 128.143.137.144
- Target hardware address: 00:00:00:00:00:00
- Target protocol address: 128.143.137.1

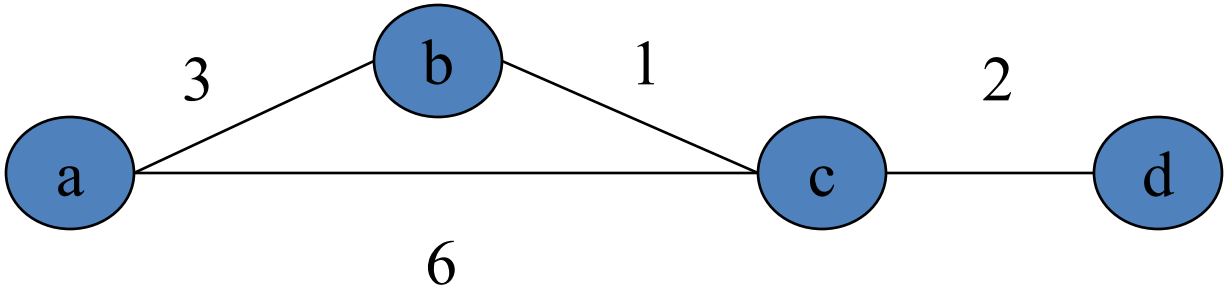


- *ARP Reply from Router137 is unicasted:*
  - Source addr: 00:e0:f9:23:a8:20
  - Dst addr: 00:a0:24:71:e4:44
- Source hardware address: 00:e0:f9:23:a8:20
- Source protocol address: 128.143.137.1
- Target hardware address: 00:a0:24:71:e4:44
- Target protocol address: 128.143.137.144

Distance vector algorithm: updates

- When a node x receives a distance vector from a neighbor v, it updates its own distance vector using the Bellman-Ford Equation
  - Considering entry (u, $d_x(u)$, NextHop)
  - If $c(x,v) + d_v(u) < d_x(u)$ then
    - $d_x(u) = c(x,v) + d_v(u)$
    - NextHop=v

Distance vector algorithm: an example



- t = 0
- a = ((a, 0, a), (b, 3, b), (c, 6, c))
- b = ((a, 3, a), (b, 0, b), (c, 1, c))
- c = ((a, 6, a), (b, 1, b), (c, 0, c) (d, 2, d))
- d = ((c, 2, c), (d, 0, d))

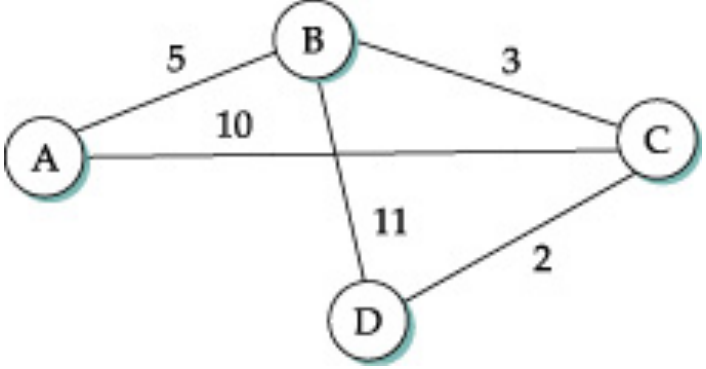
- t = 1
- a = ((a, 0, a), (b, 3, b), (c, 4, b), (d, 8, c))
- b = ((a, 3, a), (b, 0, b), (c, 1, c), (d, 3, c))
- c = ((a, 4, b), (b, 1, b), (c, 0, c), (d, 2, d))
- d = ((a, 8, c), (b, 3, c), (c, 2, c), (d, 0, d))

- t = 2
- a = ((a, 0, a), (b, 3, b), (c, 4, b), (d, 6, b))
- b = ((a, 3, a), (b, 0, b), (c, 1, c), (d, 3, c))
- c = ((a, 4, b), (b, 1, b), (c, 0, c), (d, 2, d))
- d = ((a, 6, c), (b, 3, c), (c, 2, c), (d, 0, d))

Forward search algorithm

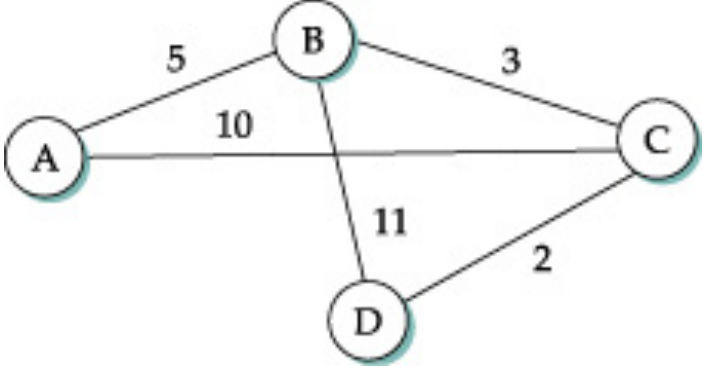
- Two lists: Tentative and Confirmed
- Each entry: (destination, cost, NextHop)

1. Confirmed = {(s,0,s)}
2. Let Next = Confirmed.last
3. For each Nbr of Next
   - Cost = Next.cost + Next → Nbr
     - If Nbr not in Confirmed or Tentative
       - Add (Nbr, Cost, Nbr) to Tentative if Next.Nexthop is s
       - Add (Nbr, Cost, Next.Nexthop) to Tentative if Next.Nexthop is not s
     - If Nbr is in Tentative and Cost is less than Nbr.Cost
       - Update Nbr.Cost to Cost and Nbr.Nexthop to Next.Nexthop
4. If Tentative not empty, pick the entry with smallest cost in Tentative and move it to Confirmed, and return to Step 2
   - Pick the smallest cost from a smaller list Tentative, rather than the rest of the graph



Forward search algorithm in OSPF

Step	Confirmed	Tentative
1	(D,0,D)	
2		
3		
4		
5		
6		
7		



Forward search algorithm in OSPF

Step	Confirmed	Tentative
1	(D,0,D)	
2	(D,0,D)	(B,11,B), (C,2,C)
3	(D,0,D), (C,2,C)	(B,11,B)
4	(D,0,D), (C,2,C)	(B,5,C) (A,12,C)
5	(D,0,D), (C,2,C), (B,5,C)	(A,12,C)
6	(D,0,D),(C,2,C),(B,5,C)	(A,10,C)
7	(D,0,D),(C,2,C),(B,5,C), (A,10,C)	