# CN_HW2_2152034

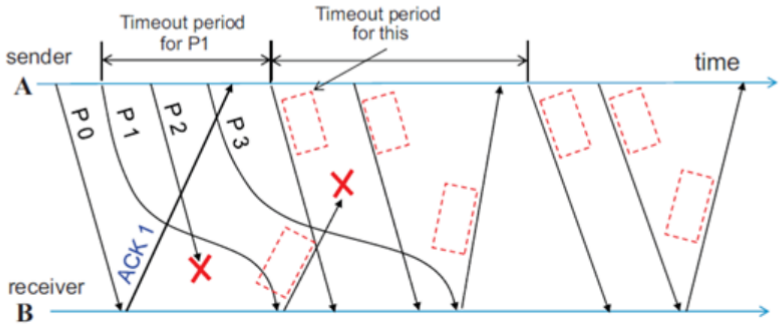
## 1. ARQ

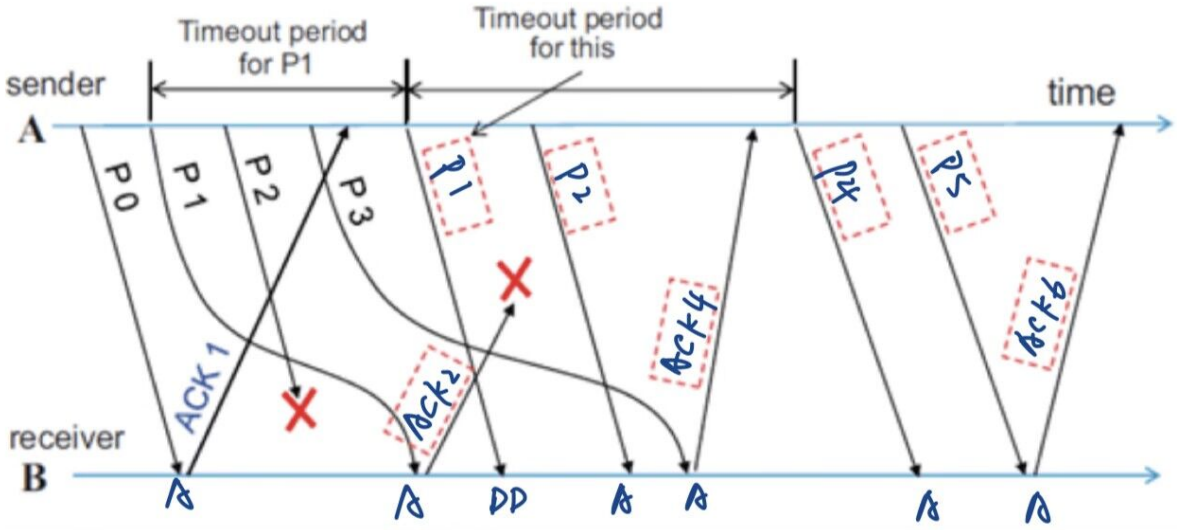
**1. ARQ (20 points)**

Consider the data transmission from sender A to receiver B via the Go-Back-N (with NAK) protocol with a large window size N (so that packet transmissions will not be limited by the window size).

a) Fill in the boxes of the figure below. Write the packet number in the form of `Px`, and the response in the form of `ACKx` or `NAKx`. (14 points)

b) Write the corresponding actions (**A** for Accept, **DD** for discard as duplicates, **DE** for discard as error) for each packet at the receiver end. (6 points)




## 2. Medium Access Control

**2. Medium Access Control (25 points, 5 points each)**

Briefly answer the following questions **in your own words.**

a) Describe the CSMA/CA mechanism used in IEEE 802.11 (WiFi). You may draw a diagram, describe in words, or write pseudo code for this.

b) Why can't we use CSMA/CD in wireless LAN?

c) Why do we need the RTS-CTS mechanism?

d) How does a bridge learn the hosts in a LAN? (Hint: Read Chapter 4.8.2 and 4.8.3 in the textbook)

e) What are the differences between a bridge/switch and a hub? (Hint: Read Chapter 4.8.4 in the textbook)

(a)

1. **Sense the Medium**: Before transmitting data, a device using CSMA/CA listens to the wireless medium to check if it is idle. If the medium is busy, it waits until it becomes idle.
2. **Interframe Space (IFS):** After detecting an idle medium, the device waits for a specific Interframe Space duration. Different types of frames have different IFS values, with higher priority frames having shorter IFS durations.
3. **Random Backoff**: After the IFS, the device enters a random backoff period. It selects a random backoff value from a contention window (CW). The CW size depends on the network congestion and is dynamically adjusted based on network conditions.
4. **Carrier Sense**: During the backoff period, the device continuously senses the medium. If the medium becomes busy, the backoff timer is frozen, and the device waits until the medium is idle again.
5. **Transmit or Continue Backoff**: Once the backoff timer expires and the medium is still idle, the device starts transmitting its frame. If the medium becomes busy during the backoff timer, the device freezes the timer and resumes the backoff process when the medium is idle again.
6. **Acknowledgment (ACK) and Retransmission**: After transmitting a frame, the device waits for an acknowledgment (ACK) frame from the receiver. If the device does not receive an ACK within a specified timeout period, it assumes a collision has occurred and retransmits the frame after a random backoff.

```
var mediumBusy = false
var backoffTimer = 0

function sendData(frame):
    waitUntilMediumIdle()
    waitForIFS()
    backoffTimer = randomBackoff()
    while backoffTimer > 0:
        if mediumBusy:
            freezeBackoffTimer()
        else:
            backoffTimer = backoffTimer - 1
            wait(1)
    if backoffTimer == 0:
        transmitFrame(frame)
```

```
        waitforACK()

function receiveData(frame):
    if frame.isACK():
        stopBackoffTimer()
    else:
        sendACK()

function waitUntilMediumIdle():
    while mediumBusy:
        senseMedium()

function waitForIFS():
    wait(IFS)

function randomBackoff():
    randomBackoff = random(CW)
    return randomBackoff

function transmitFrame(frame):
    send(frame)

function waitForACK():
    timeout = ACK_timeout
    while timeout > 0:
        if receivedACK():
            break
        else:
            timeout = timeout - 1
            wait(1)

function sendACK():
    ackFrame = createACK()
    send(ackFrame)

function senseMedium():
    mediumBusy = isMediumBusy()

function stopBackoffTimer():
    backoffTimer = 0

function freezeBackoffTimer():
    while mediumBusy:
        senseMedium()
```

(b)

1. **Collision Detection**: CSMA/CD relies on the ability to detect collisions on the network. In wired Ethernet, collisions can be detected by monitoring the transmission medium for simultaneous signals. However, in wireless LANs, due to the nature of radio waves, collisions cannot be reliably detected. The receiver in a wireless LAN cannot simultaneously listen to multiple devices transmitting simultaneously.
2. **Hidden Terminal Problem**: Wireless LANs are susceptible to the hidden terminal problem, where two devices may be out of range of each other but within range of a common access point. In such scenarios, a device may start transmitting data without being aware of another device transmitting on the same frequency. This can lead to collisions and degraded performance.
3. **Exposed Terminal Problem**: Conversely, the exposed terminal problem can occur in wireless LANs. It happens when a device refrains from transmitting data even though it would not interfere with other ongoing transmissions. This cautious behavior is due to the presence of another device in its communication range that is transmitting to a different recipient. This unnecessary restraint can result in reduced network efficiency.
4. **Inefficiency**: CSMA/CD relies on collision detection and retransmission, which can lead to increased overhead and reduced network efficiency. In wireless LANs, where collisions cannot be detected reliably, the retransmission process would be inefficient and prone to further collisions.

(c)

1. **Hidden Terminal Problem**: The hidden terminal problem occurs when two devices are out of range of each other but within range of a common access point. Without the RTS-CTS mechanism, a device may start transmitting data without being aware of another device transmitting on the same frequency, resulting in collisions and degraded performance. The RTS-CTS mechanism helps alleviate this problem by allowing devices to exchange control frames before transmitting data, ensuring that all devices are aware of ongoing transmissions.
2. **Virtual Carrier Sensing**: The RTS-CTS handshake provides a form of virtual carrier sensing. The sender first sends an RTS frame to the intended receiver, indicating its intention to transmit data. The receiver, upon receiving the RTS frame, replies with a CTS frame granting permission to the sender to transmit. Other devices within range of the sender and receiver also receive the RTS and CTS frames, effectively detecting ongoing transmissions and refraining from transmitting to avoid collisions.
3. **Collision Avoidance**: The RTS-CTS mechanism helps avoid collisions by reserving the wireless medium for the duration of the data transmission. When the sender receives the CTS frame, it knows that the medium is clear and proceeds to transmit its data without contention from other devices. Similarly, other devices within range of the sender and receiver defer their transmissions during this reserved period, reducing the chances of collisions and improving overall network efficiency.
4. **Frame Fragmentation**: In addition to collision avoidance, the RTS-CTS mechanism allows for frame fragmentation. If a data frame is large, it can be fragmented into smaller frames, and the RTS-CTS handshake is used to reserve the medium for the entire duration of the fragmented transmission. This ensures that the receiver can successfully receive all fragments without

interruptions from other devices.

(d)

1. **Initialization**: When a bridge is powered on or starts operating, it initializes its MAC address table (also known as a forwarding table or MAC forwarding database) as empty.
2. **Frame Reception**: The bridge receives Ethernet frames on its interfaces. Each frame contains the source MAC address and destination MAC address.
3. **MAC Address Learning**: The bridge examines the source MAC address of the received frame. It checks if the source MAC address is already present in its MAC address table.

   a. If the source MAC address is not present in the table, the bridge adds an entry to the table. The entry includes the source MAC address and the interface on which the frame was received. This associates the MAC address with a specific interface.

   b. If the source MAC address is already present in the table, the bridge updates the entry with the interface on which the frame was received. This refreshes the association between the MAC address and the interface.
4. **Frame Forwarding**: After learning the source MAC address, the bridge examines the destination MAC address of the frame. It looks up the MAC address table to determine the interface associated with the destination MAC address.

   a. If the destination MAC address is found in the MAC address table, the bridge forwards the frame only to the interface associated with that MAC address.

   b. If the destination MAC address is not found in the MAC address table, the bridge floods the frame to all interfaces except the one on which the frame was received. This is done to ensure that the frame reaches its destination, as the bridge does not yet know the location of the destination host.
5. **MAC Address Aging**: To adapt to changes in the network, bridges typically employ MAC address aging. Entries in the MAC address table are periodically expired and removed if they have not been recently used. This allows the bridge to adapt to changes in the network topology and update its MAC address table accordingly.

(e)

A hub and a bridge/switch are both networking devices used to connect multiple devices in a network, but they work in different ways and have different functions.

1. **Location**:
   - Hub: A hub operates at the physical layer.
   - Bridge/Switch: A bridge/switch operates at the data link layer.
2. **Operation**
   - Hub: A hub broadcasts incoming data to all connected devices, regardless of the intended recipient. This means that all devices connected to a hub receive all traffic, which can lead to network congestion and unnecessary traffic on devices that don't need it.
   - Bridge/Switch: A bridge or switch intelligently filters and forwards data frames based on their MAC addresses. It maintains a MAC address table that maps MAC addresses to specific ports

3. **Collision Domain**
    - Hub: When multiple devices connected to a hub transmit data simultaneously, collisions can occur.
    - Bridge/Switch: A bridge or switch creates separate collision domains for each of its ports.
4. **Bandwidth**
    - Hub: Since a hub broadcasts all incoming data to all connected devices, the available bandwidth is shared among all devices. As more devices transmit data simultaneously, the available bandwidth is divided further, leading to potential network congestion and decreased performance.
    - Bridge/Switch: A bridge or switch provides dedicated bandwidth to each connected device. By selectively forwarding data frames only to the appropriate ports, a bridge/switch allows devices to communicate without sharing bandwidth, resulting in better overall network performance.

# 3. IP Addressing

**3. IP Addressing (10 points)**

Briefly answer the following questions.

a) For a network with IP address 192.168.8.0/26, how many host IP addresses can be assigned in this network? (3 points)

b) With respect to the number of destinations, IP addresses can be categorized into three types. What are they? Give an example for each type of addresses. (4 points)

c) ARP is an aiding protocol for IP. Describe in your own words how ARP request and ARP reply works. (3 points)

(a)

In a network with the IP address 192.168.8.0/26, the "/26" represents the subnet mask, which determines the number of available host IP addresses within the network.

A /26 subnet mask has 26 network bits and 6 host bits. The formula to calculate the number of host IP addresses is 2 raised to the power of the number of host bits minus 2 (as the first and last IP addresses in the subnet are reserved for the network address and broadcast address, respectively).

In this case, with 6 host bits, the calculation would be:

$$2^6 - 2 = 64 - 2 = 62$$

Therefore, in the network with the IP address 192.168.8.0/26, we can assign 62 host IP addresses to devices within that network.

(b)

1. Unicast Address: A unicast address identifies a single network interface or device. It is used to send data packets from one source to a specific destination. Examples of unicast addresses include:
    - 192.168.1.10: This IP address represents a single device on a private network.
    - 8.8.8.8: This IP address is the public DNS server provided by Google.
2. Multicast Address: A multicast address is used to send data packets to a group of devices that have joined a specific multicast group. It allows efficient one-to-many communication. Examples of multicast addresses include:
    - 224.0.0.1: This IP address is the "All Systems" multicast group address, which is used for network discovery and configuration.
    - 239.255.255.250: This IP address is used for the Simple Service Discovery Protocol (SSDP) in Universal Plug and Play (UPnP) applications.
3. Broadcast Address: A broadcast address is used to send data packets to all devices within a specific network. It is a special address that is used to reach all hosts on a network segment simultaneously. Examples of broadcast addresses include:
    - 192.168.1.255: This IP address represents a broadcast address in a private network with the subnet 192.168.1.0/24.
    - 255.255.255.255: This IP address is the limited broadcast address that reaches all devices on the local network segment.

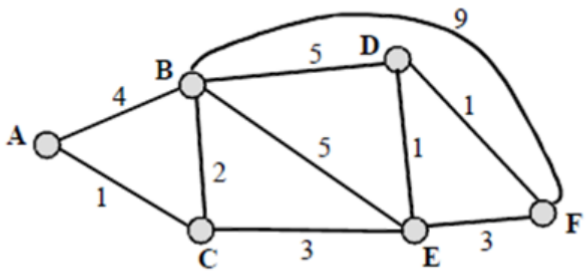
(c)

1. ARP Request:
    - When a device wants to send data to another device in the same network, but it knows the IP address of the destination device and not its MAC address, it needs to resolve the MAC address.
    - The sending device broadcasts an ARP request message to the entire local network. The ARP request message includes the sender's IP address, MAC address, and the IP address of the destination device it wants to communicate with.
    - All devices on the local network receive the ARP request message, but only the device with the matching IP address provided in the request will respond.
2. ARP Reply:
    - The device with the matching IP address specified in the ARP request generates an ARP reply message.
    - The ARP reply message is unicast directly to the device that sent the ARP request, containing the MAC address of the responding device.
    - Upon receiving the ARP reply, the sender updates its ARP cache, associating the IP address of the destination device with the corresponding MAC address.
    - The sender can now use the resolved MAC address to encapsulate the IP packets and send them directly to the destination device.

# 4. Routing: Dijkstra

**4. Routing: Dijkstra (20 points)**

Consider the network topology as follow.




a) Describe the Dijkstra algorithm with pseudo code. (5 points)

b) Find the shortest path from node A to all the other nodes in the network with the Dijkstra algorithm. Make sure you show all your steps (with the table!). (10 points)

c) Consider a networking condition in which you are asked to write an algorithm to find the most reliable path, i.e., the path with the least Bit Error Rate (BER). Assume each link in the network has a BER (in the range of 0 and 1) that is independent of other links. Can we directly use Dijkstra algorithm? If not, how to change it to solve this problem? (5 points)

(a)

```
Dijkstra(G, source):
    dist[source] = 0                // Distance from source to source is 0
    create empty set Q              // Q will hold the set of all nodes in the
graph

    for each vertex v in G:
        dist[v] = infinity          // Initialize all distances to infinity
        add v to Q                  // Add each vertex to the set Q

    while Q is not empty:
        u = vertex in Q with min dist[u]
        remove u from Q

         // Calculate the potential shortest distance from source to v
        for each neighbor v of u:
            alt = dist[u] + length(u, v)
            // Update the distance of v with the new shortest distance
            if alt < dist[v]:
                dist[v] = alt

    return dist
```

(b)

| step | set | D(B),P(B) | D(C),P(C) | D(D),P(D) | D(E),P(E) | D(F),P(F) |
|---|---|---|---|---|---|---|
| 0 | A | 4, A | 1, A | $\infty$ | $\infty$ | $\infty$ |
| 1 | A, C | 3, C | 1, A | $\infty$ | 4, C | $\infty$ |
| 2 | A, C, B | 3, C | 1, A | 8, B | 4, C | 12, B |
| 3 | A, C, B, E | 3, C | 1, A | 5, E | 4, C | 7, E |
| 4 | A, C, B, E, D | 3, C | 1, A | 5, E | 4, C | 6, D |
| 5 | A, C, B, E, D, E | 3, C | 1, A | 5, E | 4, C | 6, D |

result:

| destination | path |
|---|---|
| B | A $\rightarrow$C$\rightarrow$B |
| C | A $\rightarrow$C |
| D | A $\rightarrow$C$\rightarrow$E$\rightarrow$D |
| E | A $\rightarrow$C$\rightarrow$E |
| F | A $\rightarrow$C$\rightarrow$E$\rightarrow$D$\rightarrow$F |

(c)

The Dijkstra's algorithm is designed to find the shortest path based on the weights or distances associated with the edges in a graph.

To adapt the algorithm to find the most reliable path, we can modify the edge weights or distances in the graph to reflect the link reliability or BER.

One common approach is to convert the BER values into link "reliability" values, where a higher reliability implies a lower BER. We can use the formula:

$$Reliability = 1 - BER$$

```
ModifiedDijkstra(G, source):
    reliability[source] = 1             // Reliability from source to source is 1
    create empty set Q                   // Q will hold the set of all nodes in the
graph

    for each vertex v in G:
        reliability[v] = 0               // Initialize all reliabilities to 0
        add v to Q                       // Add each vertex to the set Q

    while Q is not empty:
```

```
        // Select the node with the maximum reliability from the set Q
        u = vertex in Q with max reliability[u]
        remove u from Q

        // Calculate the potential reliability of the path from source to v
        for each neighbor v of u:
            new_rel = reliability[u] * (1 - BER(u, v))
             // Update the reliability of v with the new highest reliability
            if new_rel > reliability[v]:
                reliability[v] = new_rel

    return reliability
```

# 5. Bellman-Ford

**5. Routing: Bellman-Ford (10 points)**

Consider the same network topology as in Question 4, and find the shortest path to node A with the Bellman-Ford algorithm. Update Order B → C → D → E → F. Make sure you show all your steps (with the table!).

| cycle | n(B),D(B) | n(C),D(C) | n(D),D(d) | n(E),D(E) | n(F),D(F) |
|---|---|---|---|---|---|
| 0 | (-, ∞) | (-, ∞) | (-, ∞) | (-, ∞) | (-, ∞) |
| 1 | (A, 4) | (A, 1) | (B, 9) | (C, 4) | (E, 7) |
| 2 | (C, 3) | (A, 1) | (E, 5) | (C, 4) | (D, 6) |
| 3 | (C, 3) | (A, 1) | (E, 5) | (C, 4) | (D, 6) |

# 6. QoS

**6. QoS (15 points)**

Give real-world examples for the following scheduling schemes.

a) FIFO Queue (4 points)

b) Priority Queue (4 points)

c) Round Robin (4 points)

d) Weighted Fair Queue (WFQ) (3 points)

a) FIFO Queue (First-In, First-Out Queue):

- Supermarket checkout lines: Customers are served in the order they arrived, with the first customer in line being served first.
- Print queue: Print jobs are processed in the order they were submitted, with the oldest job being printed first.
- Buffer management in networking: Incoming network packets are placed in a queue and processed in the order they were received.

b) Priority Queue:

- Operating system task scheduling: Processes with higher priority, such as critical system tasks or real-time processes, are given precedence over lower priority tasks.
- Emergency room triage: Patients with severe injuries or life-threatening conditions are prioritized for immediate medical attention.
- Air traffic control: Aircraft with higher priority, such as those experiencing an emergency or carrying VIPs, are given priority in landing or takeoff sequencing.

c) Round Robin:

- CPU scheduling in operating systems: Each process is allocated a fixed time quantum or slice, and the processor switches between processes in a circular manner, providing fair execution time to each process.
- Time-sharing systems: Multiple users are given equal time slices to execute their tasks in a shared computing environment.
- Tournament-style sports competitions: Teams compete against each other in a round-robin format, where each team plays against every other team once.

d) Weighted Fair Queue (WFQ):

- Quality of Service (QoS) in networking: Network traffic is divided into different classes with assigned weights, and each class is allocated a fair share of bandwidth based on its weight.
- Video streaming services: Different video streams are assigned different weights based on their importance or quality, ensuring that higher-quality streams receive more bandwidth and smoother playback.
- Multi-tenant cloud resource allocation: Virtual machines or containers belonging to different tenants are assigned different resource weights to ensure fair distribution of computing resources while considering varying workload demands.