COMPUTER NETWORKS

LAB REPORT

ASSIGNMENT 3

DEBJIT DHAR

BCSE UG 3

ROLL:002210501106

GROUP: A3

SUBMISSION: 28/10/2024

Problem Statement: Implement p-persistent CSMA technique with collision detection.

In this assignment, you have to implement a p-persistent CSMA technique with collision detection. Measure the performance parameters like throughput (i.e., average amount of data bits successfully transmitted per unit time) and forwarding delay (i.e., average end-to-end delay, including the queuing delay and the transmission delay) experienced by the CSMA frames (IEEE 802.3). Consider adding a Collision Detection module at the sender side. This will randomly inject collisions in the system.

Test the above schemes for the following cases (not limited to).

i) Plot the comparison graphs for throughput and forwarding delay by varying p.

ii) Compare efficiency of the above approach by varying p. State your observations on the impact of performance of CSMA-CD in different scenarios.

DESIGN

Theory:

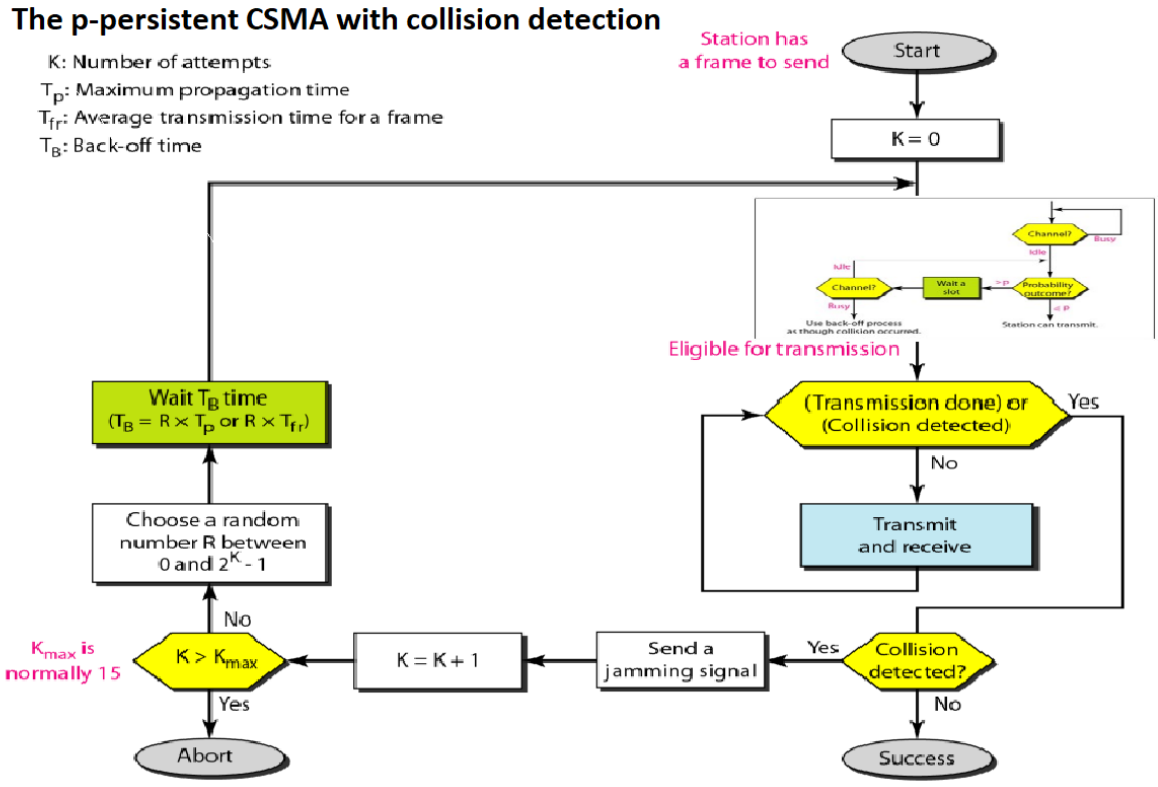
The p-persistent method is used if the channel has time slots with a slot duration equal to or greater than the maximum propagation time. The ppersistent approach combines the advantage of the other two strategies.

It reduces the chance of collision and improves efficiency. In this method, after the station finds the line idle it follows these steps:

1. with probability p, the station sends its frame.
2. with probability q=1-p, the station waits for the beginning of the next time slot and checks the line again.

a. if the line is idle, it goes to step 1.

b. if the line is busy, it acts as though a collision has occurred and uses the back-off procedure.



Explanation of Implementation

P-Persistence Implementation

Carrier Sensing:

Each station must sense the channel before attempting transmission to check if it's currently idle or busy.
If the channel is occupied (another station is transmitting), the station waits before re-sensing the channel.

P-Persistent Probability Check:

When the channel is sensed as idle, each station then uses a probabilistic check to decide whether to transmit:
A random number is generated, and if it's less than or equal to the given probability (p), the station proceeds with transmission.
If the random number is greater than p, the station waits for a short period (determined by inbetween_frametime) before re-evaluating the channel and retrying the probability check.
This probabilistic approach reduces the likelihood of collisions when multiple stations sense the channel as idle at the same time.

Collision Detection Implementation

Simultaneous Transmission Check:

The Channel class tracks the number of stations currently transmitting using a variable transmitting_stations.
Each station calls channel.begin_transmission() before transmitting:
This function increments the transmitting_stations counter and checks if more than one station is currently transmitting (i.e., transmitting_stations > 1).

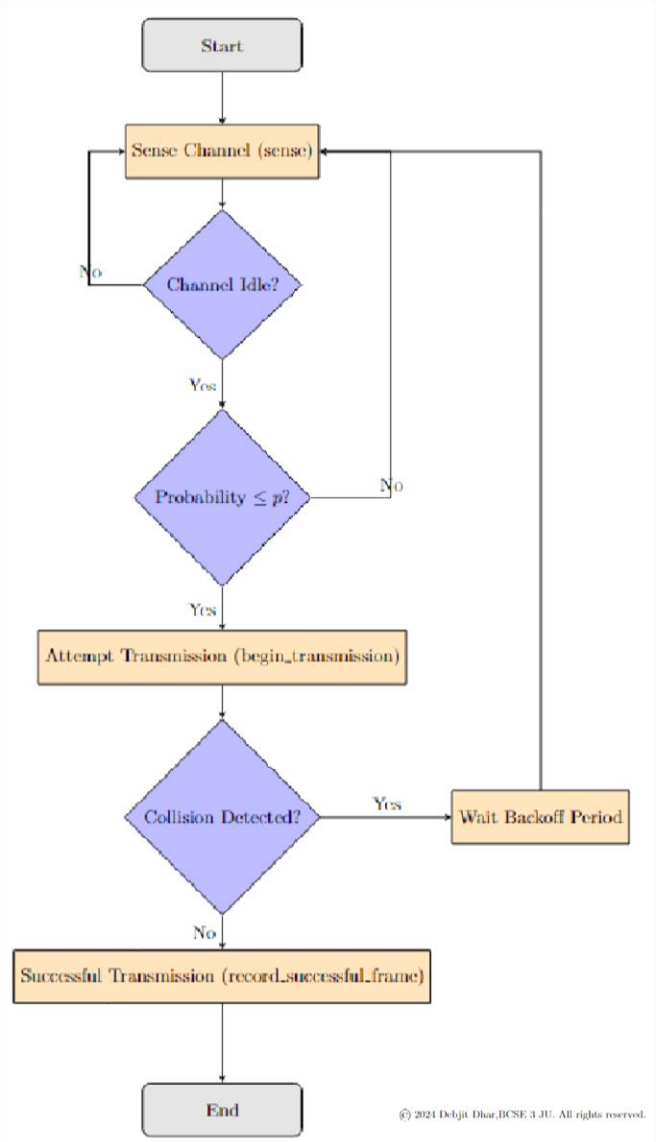
If two or more stations transmit at the same time, it's treated as a collision, and the function returns True, signaling a collision to the station.
Collision Handling:

If a collision is detected, the station:
Calls channel.record_collision() to log the collision attempt.
Immediately releases the channel by calling channel.end_transmission() to allow other stations to access the medium.
Exponential Backoff: After a collision, the station waits for a random backoff period before retrying transmission. The backoff period is selected randomly from a range (0 to max_backoff) to reduce the chance of repeated collisions by spacing out retry attempts for different stations.
Once the backoff period ends, the station returns to carrier sensing and retries the transmission attempt.
Successful Transmission:

If no other station is transmitting (i.e., transmitting_stations == 1), the station successfully transmits its frame.
After the successful transmission, the station:
Calls channel.record_successful_frame() to log a successful transmission.
Releases the channel lock by calling channel.end_transmission(), making the channel available for other stations.
Final Statistics and Metrics
After all transmissions are completed, the Channel class calculates and displays:

Efficiency: The ratio of successful frames to total transmission attempts, showing how effectively the channel was used.
Collision Rate: The ratio of collision events to total transmission attempts, indicating how frequently collisions occurred during the simulation.
Summary of How P-Persistence and Collision Detection are Achieved
P-persistence is achieved by having each station probabilistically decide whether to transmit based on a set probability, reducing collision likelihood when the channel is sensed as idle.
Collision detection is implemented by tracking simultaneous transmissions and treating any overlap as a collision. Stations then perform a random backoff before reattempting, simulating collision resolution as in actual CSMA/CD.
This approach effectively simulates p-persistent CSMA with collision detection and handling, closely mimicking how real Ethernet networks manage shared access and collisions on a network medium.

Structural Diagram



Implementation

Implementable code at: https://github.com/Debjit-Dhar/Networks

P-Persistent CSMA-CD Code Snippet:

Here I have only provided the p-persistent and collision detection module, sender and receiver remains same as Assignment 2.

```
import threading
import time
import random

class Channel:
    def __init__(self):
        self.lock = threading.Lock()
        self.total_frames = 0
        self.frame_attempts = 0
        self.collisions = 0
        self.transmitting_stations = 0

    def begin_transmission(self):
        with self.lock:
            self.transmitting_stations += 1
            if self.transmitting_stations > 1:
                # Collision detected if more than one station is
transmitting
                return True  # Collision detected
            return False  # No collision

    def end_transmission(self):
```

```
        with self.lock:
            self.transmitting_stations -= 1

    def record_successful_frame(self):
        self.total_frames += 1
        self.frame_attempts += 1

    def record_collision(self):
        self.collisions += 1
        self.frame_attempts += 1

    def get_statistics(self):
        try:
            efficiency = self.total_frames / self.frame_attempts
            collision_rate = self.collisions / self.frame_attempts
            return efficiency, collision_rate
        except ZeroDivisionError:
            return 0, 0

class PPersistentCsma(threading.Thread):
    def __init__(self, channel, index, nframes, probability,
frametime, inbetween_frametime, max_backoff):
        super().__init__()
        self.channel = channel
        self.index = index
        self.nframes = nframes
        self.probability = probability
        self.frametime = frametime
        self.inbetween_frametime = inbetween_frametime
        self.max_backoff = max_backoff
        random.seed(time.time())
```

```
    def run(self):
        for frame_num in range(1, self.nframes + 1):
            print(f"[ATTEMPT] Frame {frame_num}, Station
{self.index}")

            # Carrier Sense
            while random.random() > self.probability or
self.channel.transmitting_stations > 0:
                print(f"[-] CARRIER SENSE WAIT: Frame
{frame_num}, Station {self.index}")
                time.sleep(self.inbetween_frametime)

            # Attempt Transmission
            collision_detected = self.channel.begin_transmission()
            if collision_detected:
                # Collision handling
                print(f"[!] COLLISION: Frame {frame_num},
Station {self.index}")
                self.channel.record_collision()
                self.channel.end_transmission()

                # Exponential backoff
                backoff_time = random.uniform(0,
self.max_backoff)
                print(f"[-] BACKING OFF: Frame {frame_num},
Station {self.index}, Backoff Time: {backoff_time:.2f}s")
                time.sleep(backoff_time)
            else:
                # Simulate transmission time for a successful
transmission
```

```
                time.sleep(self.frametime)
                self.channel.record_successful_frame()
                print(f"[+] SUCCESS: Frame {frame_num}, Station
{self.index}")
                self.channel.end_transmission()

            # Wait a bit before the next frame attempt
            time.sleep(self.inbetween_frametime)

def main():
    nstations = 3  # Number of stations
    nframes = 5    # Number of frames to send per station
    probability = 0.5  # Probability threshold for p-persistent
CSMA
    frametime = 0.2  # Time to hold the channel when
successfully transmitting
    inbetween_frametime = 0.05  # Time between successive
frames for a station
    max_backoff = 1.0  # Maximum backoff time after a
collision

    # Initialize the channel and start threads
    channel = Channel()
    stations = [PPersistentCsma(channel, i + 1, nframes,
probability, frametime, inbetween_frametime, max_backoff)
for i in range(nstations)]

    for station in stations:
        station.start()
    for station in stations:
        station.join()
```

```
    # Print final statistics
    efficiency, collision_rate = channel.get_statistics()
    print(f"[INFO] Efficiency: {efficiency:.2f}")
    print(f"[INFO] Collision Rate: {collision_rate:.2f}")

if __name__ == "__main__":
    main()
```

Input-Output Format

Input Format

Number of Stations (nstations):

Type: Integer
Represents the number of stations attempting to transmit frames.
Number of Frames per Station (nframes):

Type: Integer
Represents the number of frames each station intends to send.
Probability (probability):

Type: Float
Represents the p-value (the probability threshold) used for transmission attempts.
Backoff Period (backoff_period):

Type: Float (default is random between 0.3 and 0.5)
Represents the time each station waits after detecting a collision before attempting to transmit again.
Output Format
The program produces console output detailing the following information throughout its execution:

Attempt Information:

Displays when a station attempts to send a frame.
Example:
[i] ATTEMPT: Frame 1, Station 1

Carrier Sense Information:

Indicates when a station senses the channel is busy.
Example:
[-] CARRIER SENSE: Frame 1, Station 1
Waiting Information:

Shows when a station is waiting due to the random probability check.
Example:
[-] WAITING: Frame 1, Station 1, Period: 0.45, x: 0.7
Collision Information:

Indicates if a transmission attempt failed due to a collision.
Example:
[-] FAILED: Frame 1, Station 1
Successful Transmission Information:

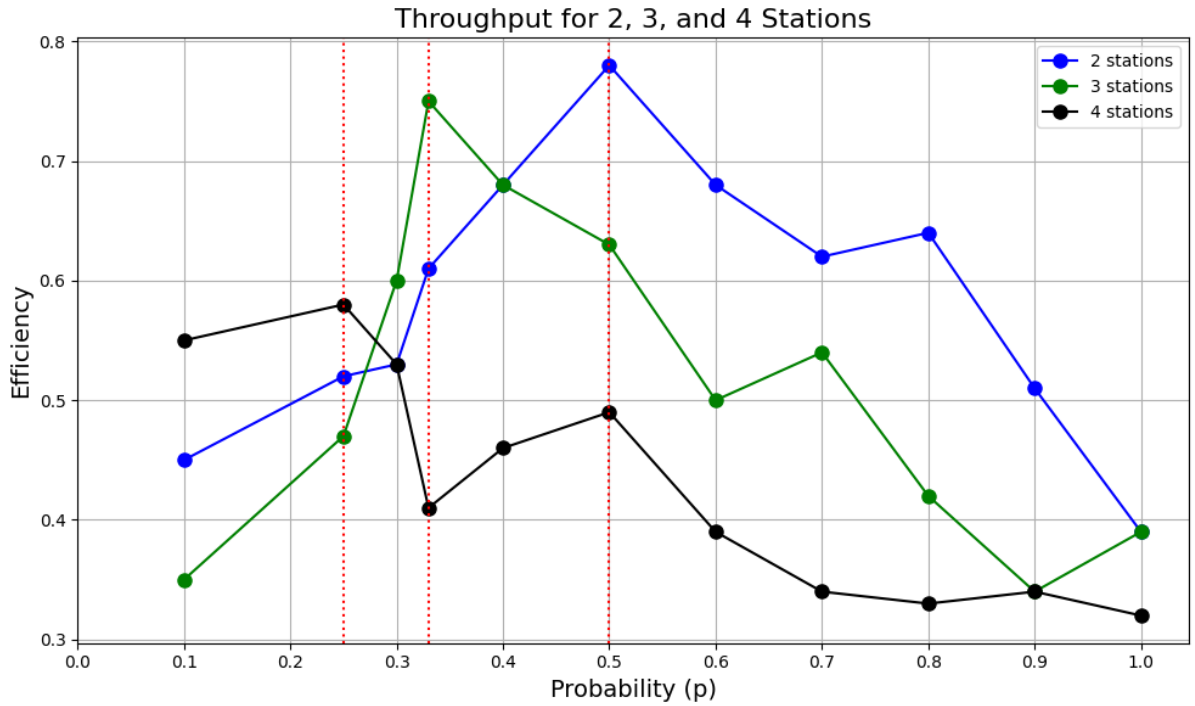
Displays when a station successfully sends a frame.
Example:
[+] SUCCESS: Frame 1, Station 1
Channel Utilization and Efficiency Statistics:

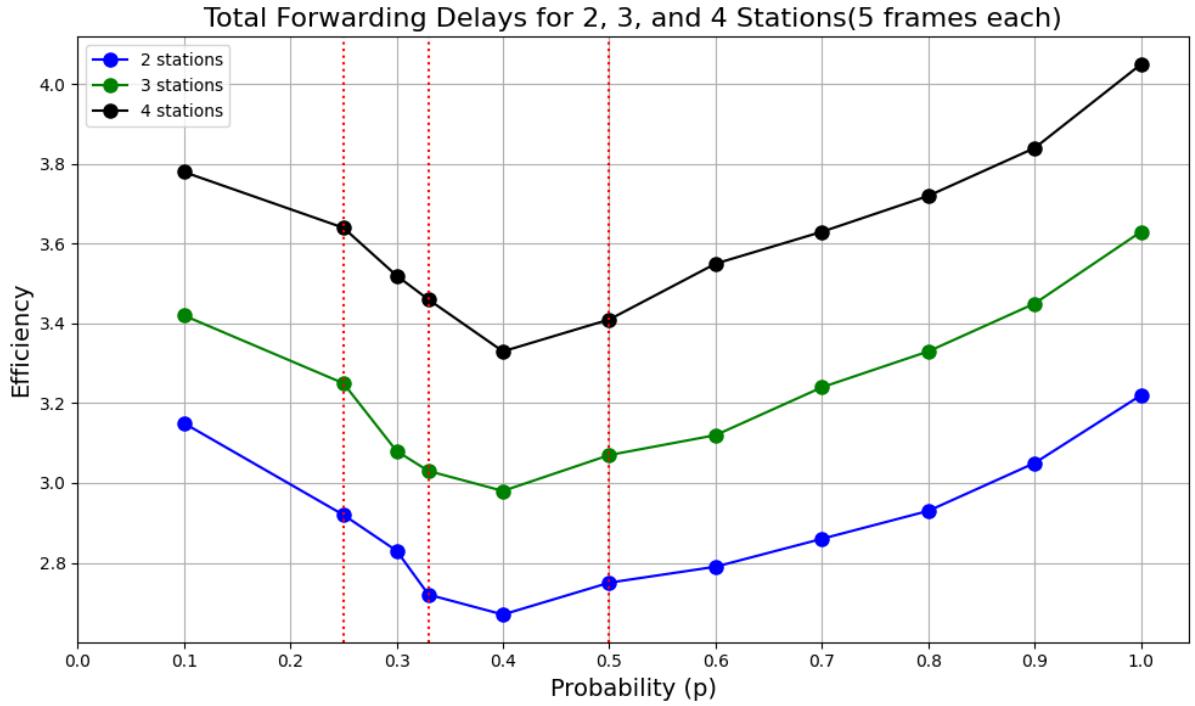
After all frames have been processed, the program prints statistics including channel utilization, channel idle time, and efficiency.
Example:
[i] INFO: Channel Utilisation: 0.67
[i] INFO: Channel Idle: 0.33
[i] INFO: Efficiency: 1.5
Summary of Example Output

[i] ATTEMPT: Frame 1, Station 1
[-] CARRIER SENSE: Frame 1, Station 1
[-] WAITING: Frame 1, Station 1, Period: 0.45, x: 0.7
[i] ATTEMPT: Frame 1, Station 2
[+] SUCCESS: Frame 1, Station 1
[i] INFO: Channel Utilisation: 0.67
[i] INFO: Channel Idle: 0.33
[i] INFO: Efficiency: 1.5

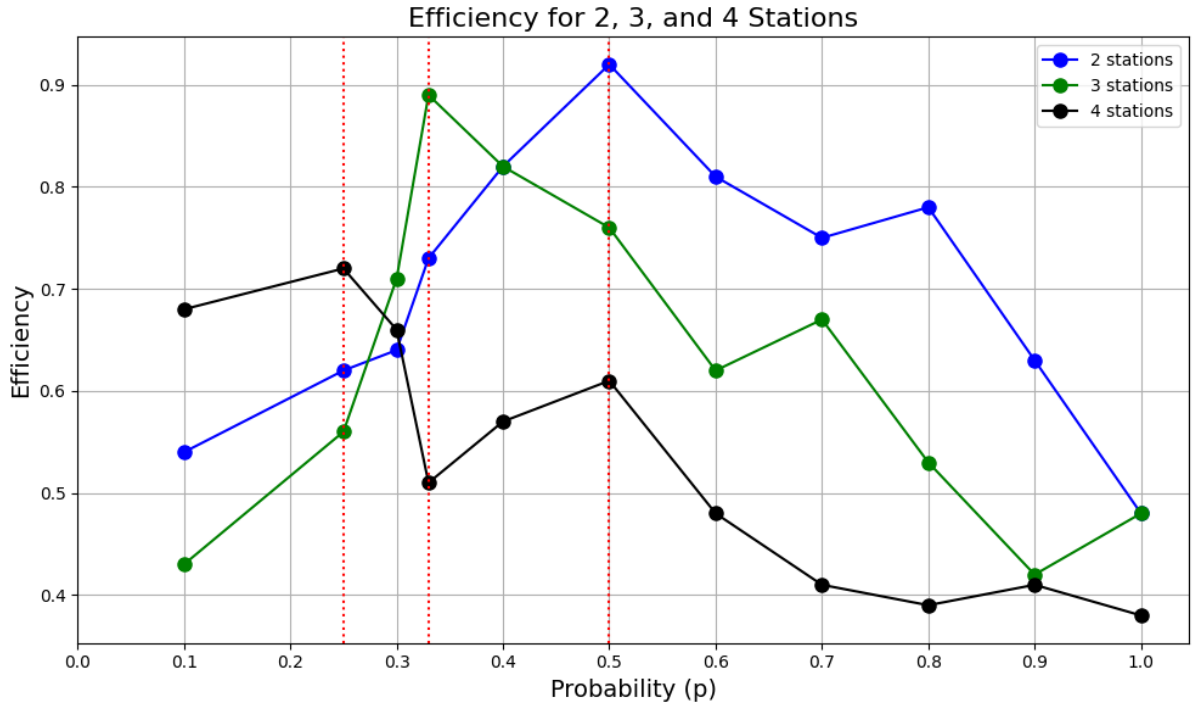
TEST CASES

i)Throughput and forwarding delay





ii)Efficiency of the approach



OBSERVATIONS

For Throughput:

Throughput generally varies with the probability p, showing an increase at lower values of p before reaching a peak and eventually declining as p approaches 1.
For each station count (2, 3, and 4), the throughput shows a single peak value at a certain p, which is considered optimal for minimizing idle and collision times.
Effect of Station Count on Throughput:

As the number of stations increases, the maximum achievable throughput decreases.
2 stations (blue line): Achieves the highest throughput across the probability range and peaks at a higher efficiency value compared to configurations with more stations.
3 stations (green line): The throughput peak is slightly lower than that for 2 stations, indicating a greater contention and more collisions.
4 stations (black line): Has the lowest peak throughput and a more pronounced drop at high probabilities, reflecting the increased impact of collisions in a more congested network.
Impact of Probability on Throughput:

For lower probabilities (around p = 0.1 - 0.3), throughput increases as p rises because each station has a small chance to transmit, leading to minimal collisions and idle time.
In the middle probability range (around p = 0.3 - 0.5), the system achieves an optimal balance between collision avoidance and utilization, leading to peak throughput.

For higher values of p (above 0.5), throughput decreases due to frequent collisions. Stations are more likely to attempt transmission simultaneously, leading to increased delays and backoff, which reduces the effective throughput.
Interpretation:

The graph aligns with the expected behavior of p-persistent CSMA/CD, where an optimal probability allows stations to achieve maximum throughput while balancing idle time and collision frequency.
Increasing the station count generally leads to a reduced peak throughput and a quicker decline as p increases due to more frequent collisions.

For Forwarding Delay:

1. **General Trend in Forwarding Delay:**
    - Forwarding delay generally follows a **U-shaped curve**, where it decreases initially as p rises to an optimal point and then begins to increase again as p approaches 1.
    - This reflects the fact that at low probabilities, forwarding delay is high due to long idle times. As p increases, idle times reduce, and forwarding delay decreases until reaching a minimum. At high probabilities, collisions become more frequent, increasing the delay.
2. **Effect of Station Count on Forwarding Delay:**
    - The number of stations directly affects the average forwarding delay.
    - **2 stations** (blue line): Experiences the lowest overall delay across the probability range. With

fewer stations, the network has fewer collisions, resulting in lower delays.

- **3 stations** (green line): Experiences moderately higher delays compared to the 2-station setup, especially at high probabilities where collisions are more likely.
- **4 stations** (black line): Shows the highest delay, particularly at higher probabilities, due to increased contention and collisions among stations.

3. **Impact of Probability on Forwarding Delay:**
   - **Low Probability ($p < 0.3$):** Forwarding delay starts high due to long idle times, as each station has a low probability of attempting transmission.
   - **Optimal Probability Range ($p \approx 0.3 - 0.5$):** Forwarding delay reaches a minimum as idle times decrease and the network operates efficiently with minimal collisions.
   - **High Probability ($p > 0.5$):** Delay begins to increase as the probability rises. At high probabilities, stations are more likely to transmit simultaneously, leading to more collisions and increased delays due to backoff mechanisms.
4. **Interpretation:**
   - This U-shaped curve is characteristic of p-persistent CSMA/CD networks, where an optimal transmission probability minimizes both idle time and collision delay.
   - With more stations, the minimum delay point occurs at a slightly lower p value, and the delay increases more sharply at higher probabilities due to the heightened risk of collision.

Combined Observations:

- **Throughput and Delay Relationship:**
  - Both graphs show that there is an optimal probability (around p = 0.3 - 0.5) where throughput peaks, and forwarding delay is minimized.
  - Beyond this point, as p increases, throughput decreases, and forwarding delay rises. This is due to the increased likelihood of collisions when multiple stations attempt transmission simultaneously.
- **Effect of Station Count:**
  - As the number of stations increases, throughput decreases and forwarding delay increases due to the greater likelihood of contention and collision.
  - This illustrates the importance of choosing an optimal p value that accounts for the number of stations in the network to maximize efficiency and minimize delay.

For Efficiency:

1. **General Trend in Efficiency:**
   - For each number of stations, efficiency initially increases with p, reaches a peak, and then begins to decline as p approaches 1.
   - This trend reflects the trade-off between idle time and collision frequency in p-persistent CSMA/CD.
2. **Effect of Probability (p) on Efficiency:**
   - **Low Probability (p < 0.3):** Efficiency is low because each station has a low probability of transmitting. This leads to significant idle time, as stations are overly cautious about using the channel.

- **Optimal Probability Range (p ≈ 0.3 - 0.5):** Efficiency reaches its peak in this range, where stations are more likely to use the channel effectively without causing excessive collisions.
- **High Probability (p > 0.5):** Efficiency decreases at high probabilities. As p approaches 1, the chance of simultaneous transmissions increases, leading to more collisions and consequently lower efficiency due to repeated backoff and retransmission attempts.

3. **Effect of Station Count on Efficiency:**
   - **2 Stations (blue line):** Achieves the highest peak efficiency and maintains a relatively high efficiency across the probability range. With fewer stations, there are fewer chances for collisions, so the protocol can perform more efficiently.
   - **3 Stations (green line):** Shows a lower peak efficiency than 2 stations and a more noticeable decline as p increases, indicating more frequent collisions as the number of stations increases.
   - **4 Stations (black line):** Reaches the lowest peak efficiency and experiences a sharp decline at higher values of p, reflecting the greater impact of contention and collisions among a larger number of stations.
4. **Key Insights:**
   - The optimal probability range (where efficiency peaks) shifts slightly lower as the number of stations increases. This suggests that a lower transmission probability is needed to maintain high efficiency in more crowded networks.
   - At very high p values, all configurations show a similar trend of declining efficiency due to the likelihood of

multiple stations attempting to access the channel at the same time, which causes collisions and reduces successful transmissions.

COMMENTS

This assignment has greatly enhanced my understanding of p-persistent carrier sense multiple access with collision detection. I have also learnt how to implement a real-time simulation for the same. Furthermore, this implementation also highlighted the disadvantages of csma-cd and why other methods such as csma-ca and channelization protocols are required.

I would like to express my heartfelt thanks to my teachers Dr Sarbani Roy and Dr Nandini Mukherjee for their guidance and support throughout this journey. Their encouragement has played a key role in helping me better comprehend these concepts.