SDM Institute of Technology

Ujire- 574 240

(Affiliated to VTU, Belagavi, Accredited by AICTE, New Delhi)

Department of Artificial Intelligence & Data Science

A Laboratory Manual of

Computer Networks (BCS502)

Semester: V

Prepared by:

Mr. Amith K S
Assistant Professor
Department of Artificial Intelligence & Data Science

COMPUTER NETWORK LABORATORY (Effective from the academic year 2023 -2024) SEMESTER – V			
Course Code	BCS502	CIE Marks	25
Number of Contact Hours/Week	3:0:2:0	SEE Marks	00
Credits – 2			

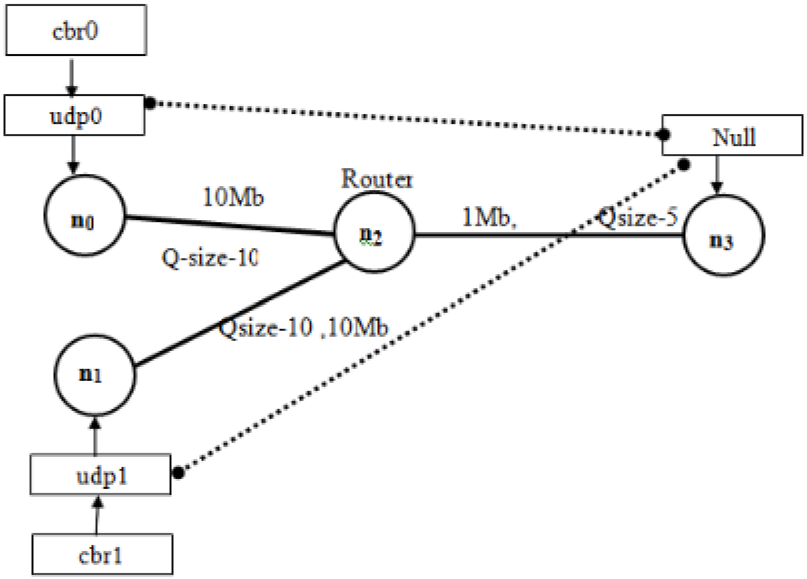
Sl. No.	PROGRAMS
1	Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth, and find the number of packets dropped
2	Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.
3	Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination..
4	Develop a program for error detecting code using CRC-CCITT (16- bits).
5	Develop a program to implement a sliding window protocol in the data link layer
6	Develop a program to find the shortest path between vertices using the Bellman-Ford and path vector routing algorithm.
7	Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present.
8	Develop a program on a datagram socket for client/server to display the messages on client side, typed at the server side.
8	Develop a program for a simple RSA algorithm to encrypt and decrypt the data.
10	Develop a program for congestion control using a leaky bucket algorithm.

Experiment 1

Implement three nodes point–to–point network with duplex links between them. Set the queue size, vary the bandwidth, and find the number of packets dropped.

Basic Explanation

This experiment simulates a simple network of three nodes connected by duplex links. By adjusting the queue size and bandwidth, you observe how network congestion leads to packet drops.



Program (exp1.tcl)

```
if {$argc != 2} {
    puts "Usage: ns exp1.tcl <bandwidth> <queue_size>"
    exit 1
}
set bw [lindex $argv 0]
set qsize [lindex $argv 1]

set ns [new Simulator]
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]

$ns duplex-link $n0 $n1 $bw 10ms DropTail
$ns duplex-link $n1 $n2 $bw 10ms DropTail
$ns queue-limit $n1 $n2 $qsize
```

```
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0

set null0 [new Agent/Null]
$ns attach-agent $n2 $null0
$ns connect $udp0 $null0

$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"

set tr [open exp1.tr w]
$ns trace-all $tr

proc finish {} {
    global ns tr
    close $tr
    exit 0
}
$ns at 5.0 "finish"
$ns run
```

How to Run

```
ns exp1.tcl 2Mb 10
awk '$1 == "d" {drop++} END {print "Packets dropped:", drop}' exp1.tr
```

Sample Output

```
Packets dropped: 12
```

Viva Questions

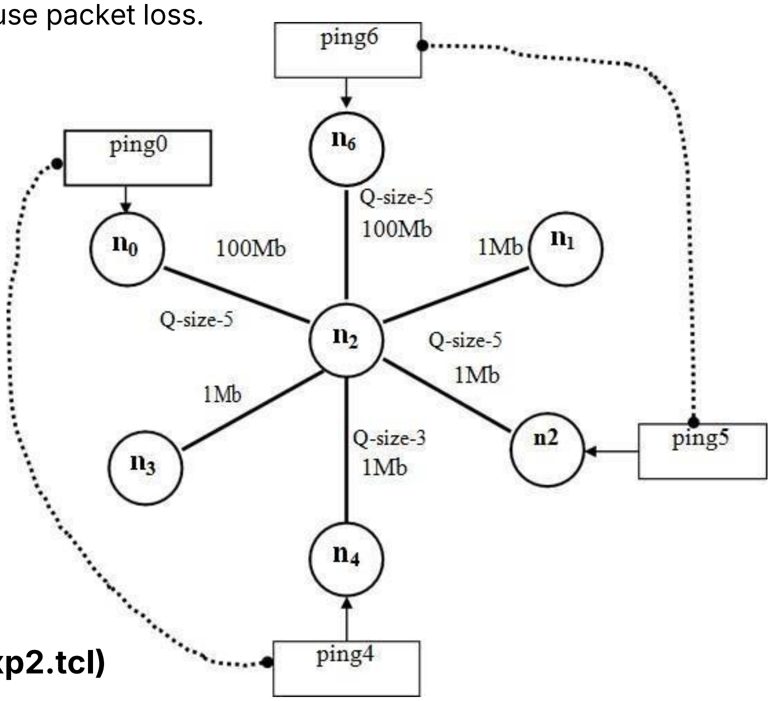
- What is the effect of queue size on packet loss?
- How does bandwidth affect congestion?
- What is the purpose of duplex links?

Experiment 2

Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.

Basic Explanation

This experiment uses ping/traceroute in a 6-node network to observe how congestion and queue limitations cause packet loss.



Program (exp2.tcl)

```
if {$argc != 2} {
    puts "Usage: ns exp2.tcl <bottleneck_bw> <queue_size>"
    exit 1
}
set bw [lindex $argv 0]
set qsize [lindex $argv 1]

set ns [new Simulator]
for {set i 0} {$i < 6} {incr i} { set n($i) [$ns node] }
$ns duplex-link $n(0) $n(1) 1Mb 10ms DropTail
$ns duplex-link $n(1) $n(2) 1Mb 10ms DropTail
$ns duplex-link $n(2) $n(3) $bw 10ms DropTail
$ns queue-limit $n(2) $n(3) $qsize
$ns duplex-link $n(3) $n(4) 1Mb 10ms DropTail
$ns duplex-link $n(4) $n(5) 1Mb 10ms DropTail

set ping0 [new Agent/Ping]
```

```
$ns attach-agent $n(0) $ping0
$ns connect $ping0 $n(5)

set tr [open exp2.tr w]
$ns trace-all $tr

proc finish {} {
    global ns tr
    close $tr
    exit 0
}
$ns at 0.5 "$ping0 send"
$ns at 5.0 "finish"
$ns run
```

How to Run

```
ns exp2.tcl 0.5Mb 5
awk '$1 == "d" {drop++} END {print "Ping packets dropped:", drop}' exp2.tr
```

Sample Output

```
Ping packets dropped: 5
```

Viva Questions

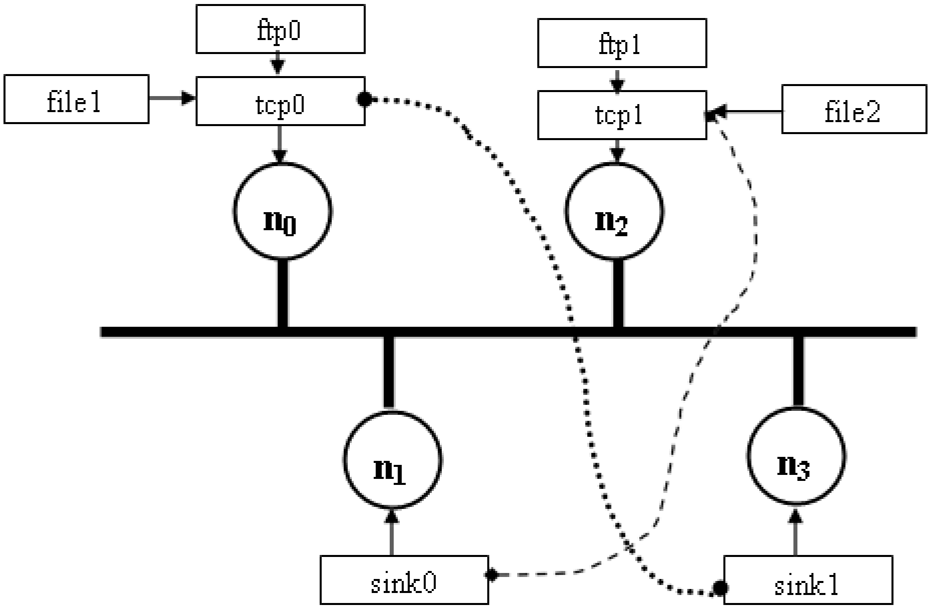
- What is the purpose of ping and traceroute?
- How does network congestion affect ICMP packets?
- How can you identify congestion in a trace file?

Experiment 3

Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.

Basic Explanation

This simulates an Ethernet LAN with multiple nodes and TCP flows, letting you visualize how the congestion window changes with network load.



Program (exp3.tcl)

```
if {$argc != 1} {
    puts "Usage: ns exp3.tcl <num_nodes>"
    exit 1
}
set n [lindex $argv 0]
if {$n < 2} {
    puts "Number of nodes must be at least 2"
    exit 1
}
set ns [new Simulator]
set nodes ""
for {set i 0} {$i < $n} {incr i} {
    append nodes " n$i"
    set n$i [$ns node]
}
eval "$ns make-lan \"$nodes\" 10Mb 10ms LL Queue/DropTail MAC/Csma/Cd"

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n[expr $n-1] $sink0
$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

set cwndfile [open cwnd.tr w]
$tcp0 trace cwnd_ $cwndfile

$ns at 0.5 "$ftp0 start"
$ns at 4.5 "$ftp0 stop"

$ns at 5.0 "finish"
proc finish {} {
    global ns cwndfile
    close $cwndfile
    exit 0
}
$ns run
```

How to Run

```
ns exp3.tcl 4
xgraph cwnd.tr
```

Sample Output

- A graph showing the congestion window size over time for each TCP flow.

Viva Questions

- What is the congestion window in TCP?
- How does TCP congestion control work?
- Why does the congestion window fluctuate?

Experiment 4

Develop a program for error detecting code using CRC-CCITT (16- bits).

Basic Explanation

CRC is used to detect errors in data transmission. CRC-CCITT (16-bit) is a standard polynomial used in many protocols.

Program (CRC16CCITT.java)

```
import java.util.Scanner;
public class CRC16CCITT {
    public static int crc16(byte[] data) {
        int crc = 0xFFFF;
        for (byte b : data) {
            crc ^= (b << 8) & 0xFFFF;
            for (int i = 0; i < 8; i++) {
                if ((crc & 0x8000) != 0)
                    crc = ((crc << 1) ^ 0x1021) & 0xFFFF;
                else
                    crc = (crc << 1) & 0xFFFF;
            }
        }
        return crc & 0xFFFF;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter data: ");
        String input = sc.nextLine();
        if (input.isEmpty()) {
            System.out.println("Input cannot be empty.");
            return;
        }
        int crc = crc16(input.getBytes());
        System.out.printf("CRC: %04X\n", crc);
        System.out.print("Enter received data: ");
        String recv = sc.nextLine();
        int recvCrc = crc16(recv.getBytes());
        if (crc == recvCrc) System.out.println("No Error Detected.");
        else System.out.println("Error Detected.");
    }
}
```

How to Run

```
javac CRC16CCITT.java
java CRC16CCITT
```

Sample Output

```
Enter data: hello
CRC: 34D2
Enter received data: hello
No Error Detected.
```

Viva Questions

- What is the purpose of CRC?
- How does CRC detect errors?
- What is the CRC-CCITT polynomial?

Experiment 5

Develop a program to implement a sliding window protocol in the data link layer.

Basic Explanation

Sliding window protocol allows multiple frames to be in transit, improving efficiency over stop-and-wait by using a window of frames.

Program (SlidingWindow.java)

```
import java.util.Scanner;
public class SlidingWindow {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter total number of frames: ");
        int totalFrames = sc.nextInt();
        System.out.print("Enter window size: ");
        int windowSize = sc.nextInt();
        if (totalFrames <= 0 || windowSize <= 0) {
            System.out.println("Total frames and window size must be positive.");
            return;
        }
        int sent = 0, ack = 0;
        while (ack < totalFrames) {
            while (sent < ack + windowSize && sent < totalFrames) {
                System.out.println("Sent frame: " + sent);
                sent++;
            }
            System.out.print("Enter ACK for frame (or -1 for lost): ");
            int ackInput = sc.nextInt();
            if (ackInput == ack) {
                System.out.println("ACK received for frame: " + ack);
                ack++;
            } else if (ackInput == -1) {
                System.out.println("Frame lost. Retransmitting from frame: " + ack);
                sent = ack;
            } else {
                System.out.println("Invalid ACK. Try again.");
            }
        }
    }
}
```

How to Run

```
javac SlidingWindow.java
java SlidingWindow
```

Sample Output

```
Enter total number of frames: 5
Enter window size: 3
Sent frame: 0
Sent frame: 1
Sent frame: 2
Enter ACK for frame (or -1 for lost): 0
ACK received for frame: 0
Sent frame: 3
Enter ACK for frame (or -1 for lost): -1
Frame lost. Retransmitting from frame: 1
...
```

Viva Questions

- What is the advantage of sliding window over stop-and-wait?
- How does the protocol handle lost frames?
- What is the window size?

Experiment 6

Develop a program to find the shortest path between vertices using the Bellman-Ford and path vector routing algorithm.

Basic Explanation

Bellman-Ford finds shortest paths in a graph, even with negative weights, and forms the basis for distance vector routing.

Program (BellmanFord.java)

```
import java.util.*;
class Edge {
    int src, dest, weight;
    Edge(int s, int d, int w) { src = s; dest = d; weight = w; }
}
public class BellmanFord {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of vertices: ");
        int V = sc.nextInt();
        System.out.print("Enter number of edges: ");
        int E = sc.nextInt();
        Edge[] edges = new Edge[E];
        for (int i = 0; i < E; i++) {
            System.out.print("Edge " + i + " (src dest weight): ");
            int s = sc.nextInt(), d = sc.nextInt(), w = sc.nextInt();
            edges[i] = new Edge(s, d, w);
        }
        System.out.print("Enter source vertex: ");
        int src = sc.nextInt();
        int[] dist = new int[V];
        Arrays.fill(dist, Integer.MAX_VALUE);
        dist[src] = 0;
        for (int i = 1; i < V; i++)
            for (Edge e : edges)
                if (dist[e.src] != Integer.MAX_VALUE && dist[e.src] + e.weight <
dist[e.dest])
                    dist[e.dest] = dist[e.src] + e.weight;
        System.out.println("Shortest distances from source:");
        for (int i = 0; i < V; i++)
            System.out.println(src + " to " + i + ": " + (dist[i] == Integer.MAX_VALUE ?
"INF" : dist[i]));
    }
}
```

How to Run

```
javac BellmanFord.java
java BellmanFord
```

Sample Output

```
Enter number of vertices: 4
Enter number of edges: 4
Edge 0 (src dest weight): 0 1 1
Edge 1 (src dest weight): 1 2 2
Edge 2 (src dest weight): 2 3 3
Edge 3 (src dest weight): 0 3 10
Enter source vertex: 0
Shortest distances from source:
0 to 0: 0
0 to 1: 1
0 to 2: 3
0 to 3: 6
```

Viva Questions

- How does Bellman-Ford differ from Dijkstra's algorithm?
- What is the count-to-infinity problem?
- What is a path vector?

Experiment 7

Using TCP/IP sockets, write a client–server program to make the client send the file name and to make the server send back the contents of the requested file if present.

Basic Explanation

This experiment demonstrates file transfer using TCP sockets, with the client requesting a file and the server responding with its contents.

Server (FileServer.java)

```java
import java.io.*;
import java.net.*;
public class FileServer {
    public static void main(String[] args) throws IOException {
        ServerSocket ss = new ServerSocket(1234);
        while (true) {
            Socket s = ss.accept();
            DataInputStream in = new DataInputStream(s.getInputStream());
            DataOutputStream out = new DataOutputStream(s.getOutputStream());
            String fname = in.readUTF();
            File f = new File(fname);
            if (f.exists() && f.isFile()) {
                BufferedReader br = new BufferedReader(new FileReader(f));
                String line;
                while ((line = br.readLine()) != null) out.writeUTF(line);
                br.close();
            } else {
                out.writeUTF("ERROR: File not found or not a regular file.");
            }
            s.close();
        }
    }
}
```

Client (FileClient.java)

```java
import java.io.*;
import java.net.*;
import java.util.Scanner;
```

```
public class FileClient {
    public static void main(String[] args) throws IOException {
        Socket s = new Socket("localhost", 1234);
        DataOutputStream out = new DataOutputStream(s.getOutputStream());
        DataInputStream in = new DataInputStream(s.getInputStream());
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter file name: ");
        String fname = sc.nextLine();
        out.writeUTF(fname);
        try {
            while (true) {
                String str = in.readUTF();
                System.out.println(str);
            }
        } catch (EOFException e) {}
        s.close();
    }
}
```

How to Run

- Start server: `java FileServer`
- Start client: `java FileClient`

Sample Output

```
Enter file name: test.txt
Hello, this is a test file.
```

Viva Questions

- Why is TCP used for file transfer?
- How does the server handle a missing file?
- What are the advantages of client-server architecture?

Experiment 8

Develop a program on a datagram socket for client/server to display the messages on client side, typed at the server side.

Basic Explanation

This experiment uses UDP sockets for message exchange between client and server, demonstrating connectionless communication.

Server (UDPServer.java)

```
import java.net.*;
public class UDPServer {
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket(9876);
        byte[] receive = new byte[65535];
        while (true) {
            DatagramPacket DpReceive = new DatagramPacket(receive, receive.length);
            ds.receive(DpReceive);
            String msg = new String(receive, 0, DpReceive.getLength());
            System.out.println("Client: " + msg);
            if (msg.equalsIgnoreCase("exit")) break;
        }
        ds.close();
    }
}
```

Client (UDPClient.java)

```
import java.net.*;
import java.util.Scanner;
public class UDPClient {
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket();
        InetAddress ip = InetAddress.getLocalHost();
        Scanner sc = new Scanner(System.in);
        while (true) {
            System.out.print("Enter message (type 'exit' to quit): ");
            String str = sc.nextLine();
            DatagramPacket dp = new DatagramPacket(str.getBytes(), str.length(), ip, 9876);
            ds.send(dp);
            if (str.equalsIgnoreCase("exit")) break;
        }
        ds.close();
    }
}
```

How to Run

- Start server: `java UDPServer`
- Start client: `java UDPClient`

Sample Output

```
Enter message (type 'exit' to quit): Hello
Client: Hello
Enter message (type 'exit' to quit): exit
Client: exit
```

Viva Questions

- What is the difference between TCP and UDP?
- Why is UDP suitable for real-time communication?
- How does the server identify different clients?

Experiment 9

Develop a program for a simple RSA algorithm to encrypt and decrypt the data.

Basic Explanation

RSA is a public-key cryptosystem. Here, you manually implement encryption and decryption using modular arithmetic, not using any inbuilt crypto libraries.

Program (SimpleRSA.java)

```
import java.math.BigInteger;
import java.util.Random;
import java.util.Scanner;

public class SimpleRSA {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Key generation (for demonstration, small primes are used)
        BigInteger p = BigInteger.probablePrime(8, new Random());
        BigInteger q = BigInteger.probablePrime(8, new Random());
        BigInteger n = p.multiply(q);
        BigInteger phi = (p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE));

        // Choose e
        BigInteger e = BigInteger.valueOf(3);
        while (phi.gcd(e).intValue() > 1) {
            e = e.add(BigInteger.valueOf(2));
        }

        // Compute d
        BigInteger d = e.modInverse(phi);

        System.out.println("Public key (n, e): (" + n + ", " + e + ")");
        System.out.println("Private key (n, d): (" + n + ", " + d + ")");

        System.out.print("Enter a number to encrypt (as integer < n): ");
        BigInteger message = new BigInteger(sc.nextLine());
        if (message.compareTo(n) >= 0) {
            System.out.println("Message must be less than n.");
            return;
        }

        // Encryption: c = m^e mod n
        BigInteger cipher = message.modPow(e, n);
        System.out.println("Encrypted: " + cipher);

        // Decryption: m = c^d mod n
        BigInteger decrypted = cipher.modPow(d, n);
        System.out.println("Decrypted: " + decrypted);
    }
}
```

How to Run

```
javac SimpleRSA.java
java SimpleRSA
```

Sample Output

```
Public key (n, e): (24169, 3)
Private key (n, d): (24169, 16067)
Enter a number to encrypt (as integer < n): 1234
Encrypted: 1953125
Decrypted: 1234
```

Viva Questions

- What is the role of public and private keys in RSA?
- Why is modular arithmetic used in RSA?
- What are the limitations of this simple implementation?

Experiment 10

Develop a program for congestion control using a leaky bucket algorithm.

Basic Explanation

The leaky bucket algorithm regulates data flow, ensuring the output rate does not exceed a fixed rate, and drops packets if the bucket overflows.

Program (LeakyBucket.java)

```
import java.util.Scanner;
public class LeakyBucket {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter bucket size: ");
        int bucketSize = sc.nextInt();
        System.out.print("Enter output rate: ");
        int outputRate = sc.nextInt();
        System.out.print("Enter number of packets: ");
        int n = sc.nextInt();
        int[] packets = new int[n];
        for (int i = 0; i < n; i++) {
            System.out.print("Packet " + (i+1) + " size: ");
            packets[i] = sc.nextInt();
        }
        int bucket = 0;
        for (int p : packets) {
            if (p > (bucketSize - bucket)) {
                System.out.println("Packet of size " + p + " dropped");
                bucket = bucketSize;
            } else {
                bucket += p;
                System.out.println("Packet of size " + p + " accepted");
            }
            bucket = Math.max(0, bucket - outputRate);
        }
    }
}
```

How to Run

```
javac LeakyBucket.java
java LeakyBucket
```

Sample Output

```
Enter bucket size: 1000
Enter output rate: 200
Enter number of packets: 3
Packet 1 size: 500
Packet of size 500 accepted
Packet 2 size: 700
Packet of size 700 dropped
Packet 3 size: 300
Packet of size 300 accepted
```

Viva Questions

- How does the leaky bucket algorithm control congestion?
- What is the difference between leaky bucket and token bucket?
- What happens if the input rate exceeds the output rate?