**Unit-IV**

**Transport layer:** Transport Service, Elements of Transport Protocols, Congestion Control, TCP and UDP Protocols, Quality of Service Model, Best Effort Model, Network Performance Issues.

Transport layer

**Introduction**

The **Transport Layer** in the Open System Interconnection (OSI) model is responsible for end-to-end delivery over a network. Whereas the network layer is concerned with the end - to- end delivery of individual packets and it does not recognize any relationship between those packets.

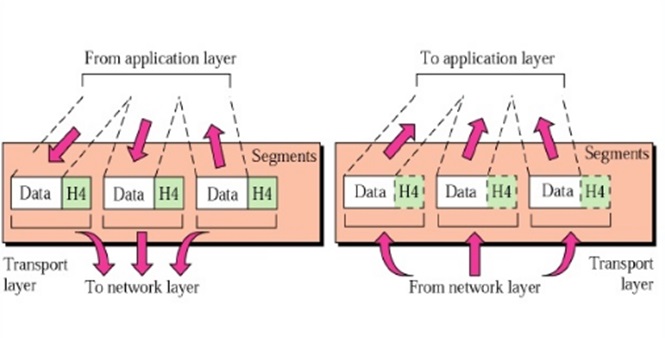
* This layer treats each packet independently because each packet belongs to a different message.
* The **transport layer** ensures that each message should reach its destination completely and in order so that it maintains error and flow control to the source to destination to ensure proper data transmission.
* The **transport layer** establishes a connection between two end ports. A connection is a single logical path from source to destination which is associated with all the packets in a message.
* **Transport Layer** uses some standard protocols to enhance its functionalities are TCP (Transmission Control Protocol), UDP (User Datagram Protocol), DCCP( Datagram Congestion Control Protocol), etc.

**Working of Transport Layer**

The transport layer takes services from the Application layer and provides services to the Network layer.

* **At the sender’s side:**The transport layer receives data (message) from the Application layer and then performs Segmentation, divides the actual message into segments, adds the source and destination’s port numbers into the header of the segment, and transfers the message to the Network layer.
* **At the receiver’s side:**The transport layer receives data from the Network layer, reassembles the segmented data, reads its header, identifies the port number, and forwards the message to the appropriate port in the Application layer.

This figure shows the relationship of the **transport layer** to the network and session layer.



**Functions of the Transport Layer**

Specific **functions** of the transport layer are as follows:

1. **Service-point addressing**

Computers often run many programs at the same time. Due to this, source-to-destination delivery means delivery from a specific job (currently running program) on one computer to a specific job (currently running program) on the other system not only one computer to the next.

For this reason, the transport layer added a specific type of address to its header, it is referred to as a service point address or port address.

By this address each packet reaches the correct computer and also the transport layer gets the complete message to the correct process on that computer.

1. **Segmentation and Reassembly**

In segmentation, a message is divided into transmittable segments; each segment containing a sequence number. This number enables this layer to reassemble the message.

Upon arriving at its destination system message is reassembled correctly, identify and replaces packets that were lost in transmission.

1. **Connection Control**

It can be either of two types:

1. Connectionless Transport Layer
2. Connection Oriented Transport Layer
3. Connectionless Transport Layer

* This Transport Layer treats each packet as an individual and delivers it to the destination machine.
* In this type of transmission, the receiver does not send an acknowledgment to the sender about the receipt of a packet. This is a faster communication technique.

1. Connection Oriented Transport Layer

* This Transport Layer creates a connection with the Transport Layer at the destination machine before transmitting the packets to the destination.
* To Create a connection following three steps are possible:
* Connection establishment
* Data transfer
* Connection termination
* When all the data are transmitted connection is terminated. Connectionless Service is less reliable than connection Oriented Service.

1. **Multiplexing and Demultiplexing**

* Multiple packets from diverse applications are transmitted across a network needs very dedicated control mechanisms, which are found in the transport layer.
* The transport layer accepts packets from different processes. These packets are differentiated by their port numbers and pass them to the network layer after adding proper headers.
* In Demultiplexing, at the receiver's side to obtain the data coming from various processes. It receives the segments of data from the network layer and delivers it to the appropriate process running on the receiver's machine.

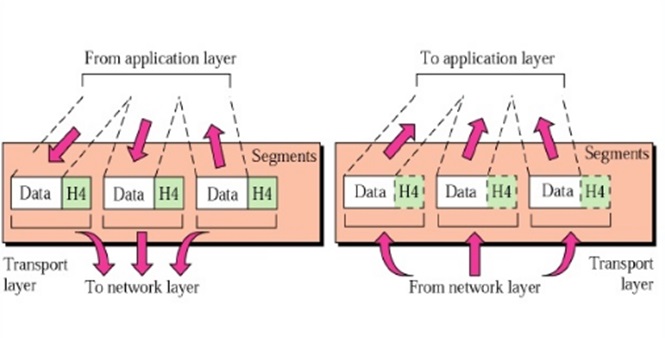
1. **Flow Control**

* The transport layer also responsible for the flow control mechanism between the adjacent layers of the TCP/IP model.
* It does not perform across a single link even it performs an end-to-end node.
* By imposing flow control techniques data loss can be prevented from the cause of the sender and slow receiver.
* For instance, it uses the method of sliding window protocol in this method receiver sends a window back to the sender to inform the size of the data is received.

1. **Error Control**

* Error Control is also performed end to end like the data link layer.
* In this layer to ensure that the entire message arrives at the receiving transport layer without any error(damage, loss or duplication). Error Correction is achieved through retransmission of the packet.
* The data has arrived or not and checks for the integrity of data, it uses the ACK and NACK services to inform the sender.

**Example of Transport Layer**

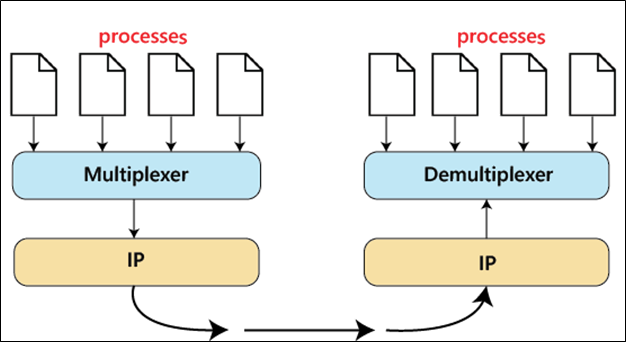


* This figure shows an example data coming from upper layers have service point addresses j and k (j is the address of sending application and k is the address of the receiving application).
* Since the data size is greater than the network layer can occupy. The data are divided into two packets. Each packet containing the service point addresses ( j and k).
* In the network layer, network addresses (A and P) are combined with each packet.
* The packet may travel on different paths and arrive at the destination either in order or out of order.
* The two packets are transmitted to the destination network layer. This is responsible for removing the network layer headers.
* Two packets are now passed to the transport layer, where they are combined for delivery to the upper layers.

**Responsibilities of a Transport Layer**

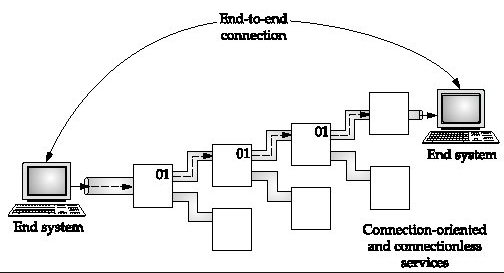
The Process to Process Delivery

1. End-to-End Connection between Hosts
2. Multiplexing and Demultiplexing
3. Congestion Control
4. Data integrity and Error correction
5. Flow control
6. **The Process to Process Delivery -** While Data Link Layer requires the MAC address (48 bits address contained inside the Network Interface Card of every host machine) of source-destination hosts to correctly deliver a frame and the Network layer requires the IP address for appropriate routing of packets, in a similar way Transport Layer requires a Port number to correctly deliver the segments of data to the correct process amongst the multiple processes running on a particular host. A port number is a 16-bit address used to identify any client-server program uniquely.



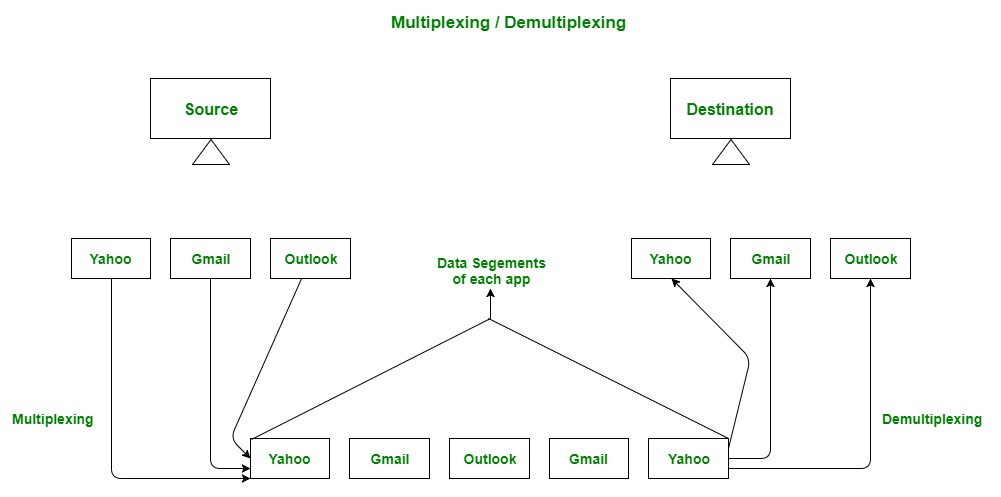
*Process to Process Delivery*

1. **End-to-end Connection between Hosts -** The transport layer is also responsible for creating the end-to-end Connection between hosts for which it mainly uses TCP and UDP. TCP is a secure, connection-orientated protocol that uses a handshake protocol to establish a robust connection between two end hosts. TCP ensures the reliable delivery of messages and is used in various applications. UDP, on the other hand, is a stateless and unreliable protocol that ensures best-effort delivery. It is suitable for applications that have little concern with flow or error control and requires sending the bulk of data like video conferencing. It is often used in multicasting protocols.



*End to End Connection*

1. **Multiplexing and Demultiplexing –** Multiplexing (many to one) is when data is acquired from several processes from the sender and merged into one packet along with headers and sent as a single packet. Multiplexing allows the simultaneous use of different processes over a network that is running on a host.  The processes are differentiated by their port numbers. Similarly, Demultiplexing(one to many is required at the receiver side when the message is distributed into different processes. Transport receives the segments of data from the network layer distributes and delivers it to the appropriate process running on the receiver’s machine.



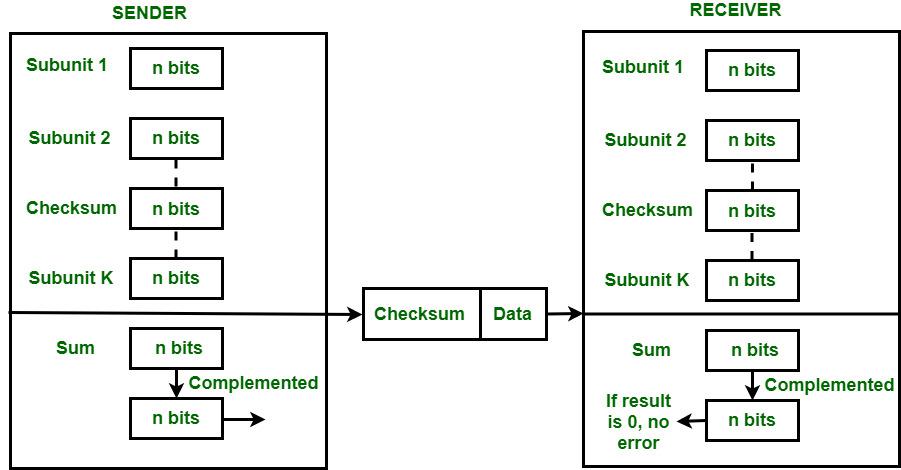
*Multiplexing and Demultiplexing*

1. **Congestion Control -** Congestion is a situation in which too many sources over a network attempt to send data and the router buffers start overflowing due to which loss of packets occurs. As a result, the retransmission of packets from the sources increases the congestion further. In this situation, the Transport layer provides Congestion Control in different ways. It uses open-loop congestion control to prevent congestion and closed-loop congestion control to remove the congestion in a network once it occurred. TCP provides AIMD – additive increases multiplicative decrease and leaky bucket technique for congestion control.



*Leaky Bucket Congestion Control Technique*

1. **Data integrity and Error Correction** - The transport layer checks for errors in the messages coming from the application layer by using error detection codes, and computing checksums, it checks whether the received data is not corrupted and uses the ACK and NACK services to inform the sender if the data has arrived or not and checks for the integrity of data.



*Error Correction using Checksum*

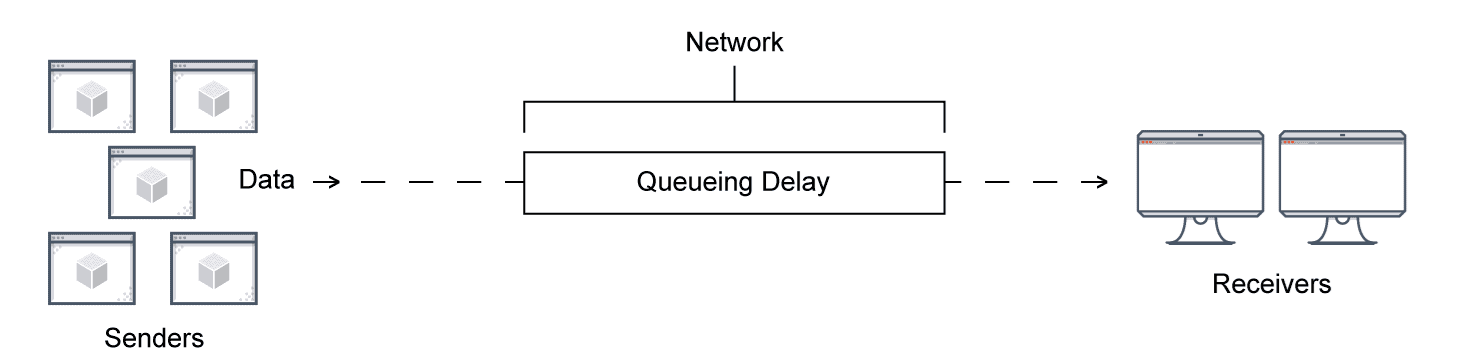
1. **Flow Control -** The transport layer provides a flow control mechanism between the adjacent layers of the TCP/IP model. TCP also prevents data loss due to a fast sender and slow receiver by imposing some flow control techniques. It uses the method of sliding window protocol which is accomplished by the receiver by sending a window back to the sender informing the size of data it can receive.

Congestion

Network congestion refers to a reduction in quality of service (QOS) that causes packet loss, queueing delay, or the blocking of new connections. Typically, network congestion occurs in cases of traffic overloading when a link or network node is handling data in excess of its capacity.

To avoid collapse and reduce the effects of congestion in the network, organizations use various congestion avoidance and congestion control methods. These include:

* TCP/IP window reduction
* Fair queueing in network devices such as routers, switches, and other devices
* Priority schemes which transmit higher priority packets ahead of other traffic
* Explicit network resource allocation via admission controls toward specific flows



**Identification of Network Congestion**

There are five ways to identify network congestion:

* **Bandwidth:** The most common cause of network congestion is bandwidth. Bandwidth refers to the ideal capacity of the network to transfer a certain amount of data from a source to a destination in a specific amount of time. A lack of bandwidth can lead to network outages.
* **Latency**: Latency is the time taken to transmit, capture, transmit, and process data from source to destination. It refers to the speed of your network traffic measured in milliseconds. High latency can lead to a slower network. Latency numbers may vary based on the application and network connection usage.
* **Jitter**: Jitter refers to the time delay while sending data packets to a destination from a source over a network. When traffic becomes unpredictable, it causes jitter and network congestion. Jitter can also impact the quality of audio and video quality on your network. Network equipment and devices try to adjust the changes caused by traffic patterns, which creates jitter and leads to congestion as a cascading effect.
* **Packet Retransmission**: Packet retransmission is required when the movement of data packets stops due to packet loss, packet damage, or more. In such cases, data packets are resent from source to destination, increasing network congestion.
* **Collision:** Packet collision occurs when two or more network nodes try to send data simultaneously. This leads to packet loss and requires resending of packets, which can negatively impact network performance. Collision is a back-off process where all the packets have to wait to clear the network congestion. It can be due to an inappropriate connection, poor cabling, and more.

**Causes of Network Congestion**

The first step to troubleshoot network issues such as congestion is to understand and identify their root cause. There can be several causes of network congestion:

* **Unneeded Traffic:** The most common cause of network congestion is unneeded traffic. It may include streaming video content, advertisements, or junk VoIP phone calls that consume bandwidth. It’s important to identify unneeded traffic before it slows down your network.
* **Misconfigured Traffic:** Business traffic typically comes from multiple sources.
  + Unicast traffic to support video functions, voice calls, or data transfer
  + Broadcast traffic for network operations
  + Multicast traffic for real-time media streams

All these can be business-critical traffic; however, you need to prioritize them to eliminate network congestion. Network devices treat this intermixed traffic equally, which can cause network issues and outages. Organizations can use Quality of Service (QoS) protocols to manage misconfigured traffic.

* **Business – Critical Traffic:** In a business network, the network manager decides which type of network is business-critical to ensure the required bandwidth is reserved. The remaining bandwidth can be used for traffic coming from other sources.
* **Outdated or Non-Compatible Hardware:** Enterprises must look for outdated and non-compatible devices and try upgrading network capacity to speed up the enterprise's network demands. A hardware upgrade is critical to have an optimal layout. If hardware assets such as switches, routers, servers, and cable connections can’t handle the data speeds the network requires, they can slow down the network and lead to network congestion.
* **Overused and too many devices:** Overused devices can also contribute to network congestion. Pushing devices to their maximum capacity can often result in over-utilization. Excessive volumes of device usage can also cause network congestion as they can provide a surplus of requests for data.
* **Bandwidth hogs:** A bandwidth hog is the over-usage of data on a particular device. The difference between average data usage and a hog's usage depends on the user or device. It’s important to monitor bandwidth in real-time to detect a bandwidth hog.

**How to solve network congestion problems**

* Traffic and bandwidth monitoring: The first step to resolve network congestion is to identify issues such as over-utilization of devices, insufficient bandwidth, and more. Monitoring networks also provide sufficient insights to identify problematic areas. You can also use network performance monitoring tools to identify such issues quickly. Once you get insights into your network performance and how data traffic flows, you can upgrade your devices, bandwidth, or network hardware to maximize the benefit.
* Segmenting and prioritizing: Segmenting your network into small subnets increases efficiency by letting you prioritize traffic. This also helps you accurately monitor network traffic. Segmenting networks can reduce data traffic, producing a more viable network. Prioritization refers to the capacity to minimize traffic. Critical network traffic areas need more attention than others.

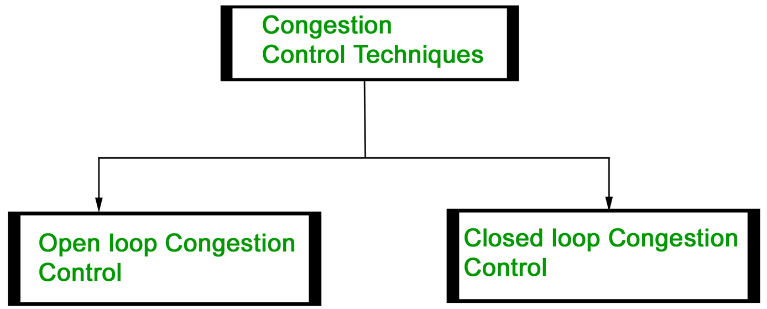
**Helpful tools for monitoring and preventing network congestion**

Several network performance monitoring tools can detect bandwidth hogs and monitor network traffic to help ensure there is no congestion. These tools help to:

* Monitor and analyze network traffic patterns down to the interface level and help IT teams identify endpoints, protocols, and applications that consume more bandwidth.
* Create reports to analyze and monitor network bandwidth more precisely. These tools convert raw numbers into easy-to-interpret charts, tables, and utilization reports to better understand how the network is being used.
* Get instant alerts, so you can quickly act if traffic increases or decreases and efficiently remediate the problem.

**Congestion Control techniques in Computer Networks**

Congestion control refers to the techniques used to control or prevent congestion. Congestion control techniques can be broadly classified into two categories:



* Retransmission Policy **-** Back Pressure
* Window Policy **-** Choke Packet
* Acknowledgement Policy **-** Implicit Signaling
* Discording Policy **-** Explicit Signaling
* Admission Policy

**Open Loop Congestion Control**

Open loop congestion control policies are applied to prevent congestion before it happens. The congestion control is handled either by the source or the destination.

**Policies adopted by open loop congestion control –**

* **Retransmission Policy:** It is the policy in which retransmission of the packets are taken care of. If the sender feels that a sent packet is lost or corrupted, the packet needs to be retransmitted. This transmission may increase the congestion in the network.

To prevent congestion, retransmission timers must be designed to prevent congestion and also able to optimize efficiency.

* **Window Policy:** The type of window at the sender’s side may also affect the congestion. Several packets in the Go-back-n window are re-sent, although some packets may be received successfully at the receiver side. This duplication may increase the congestion in the network and make it worse.

Therefore, Selective repeat window should be adopted as it sends the specific packet that may have been lost.

* **Discarding Policy:** A good discarding policy adopted by the routers is that the routers may prevent congestion and at the same time partially discard the corrupted or less sensitive packages and also be able to maintain the quality of a message.

In case of audio file transmission, routers can discard less sensitive packets to prevent congestion and also maintain the quality of the audio file.

* **Acknowledgment Policy:** Since acknowledgements are also the part of the load in the network, the acknowledgment policy imposed by the receiver may also affect congestion. Several approaches can be used to prevent congestion related to acknowledgment.

The receiver should send acknowledgement for N packets rather than sending acknowledgement for a single packet. The receiver should send an acknowledgment only if it has to send a packet or a timer expires.

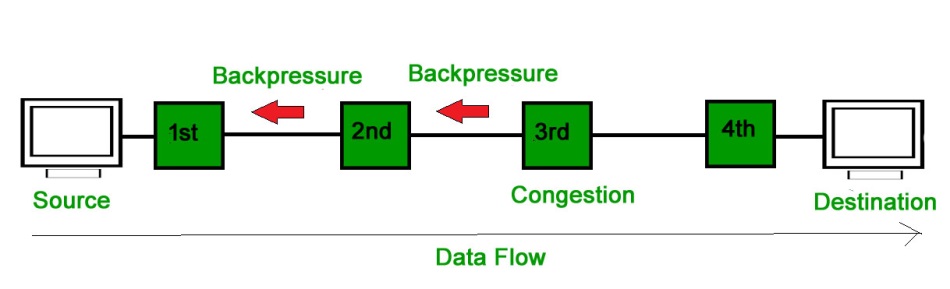
* **Admission Policy:** In admission policy a mechanism should be used to prevent congestion. Switches in a flow should first check the resource requirement of a network flow before transmitting it further. If there is a chance of a congestion or there is a congestion in the network, router should deny establishing a virtual network connection to prevent further congestion.

All the above policies are adopted to prevent congestion before it happens in the network. 

**Closed Loop Congestion Control**

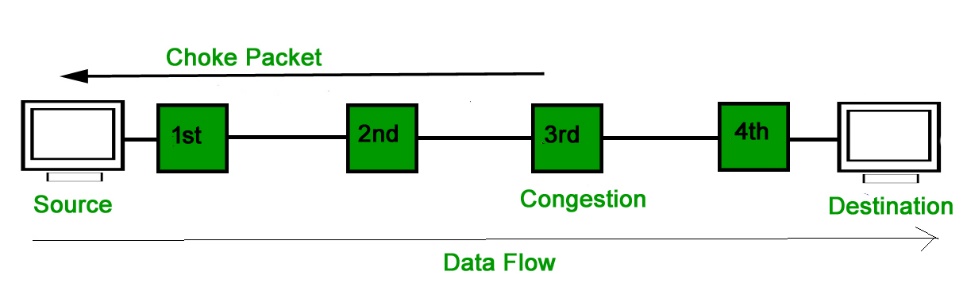
Closed loop congestion control techniques are used to treat or alleviate congestion after it happens. Several techniques are used by different protocols; some of them are:

* **Backpressure:**  Backpressure is a technique in which a congested node stops receiving packets from upstream node. This may cause the upstream node or nodes to become congested and reject receiving data from above nodes. Backpressure is a node-to-node congestion control technique that propagate in the opposite direction of data flow. The backpressure technique can be applied only to virtual circuit where each node has information of its above upstream node.



In above diagram the 3rd node is congested and stops receiving packets as a result 2nd node may be get congested due to slowing down of the output data flow. Similarly 1st node may get congested and inform the source to slow down.

* **Choke Packet Technique:** Choke packet technique is applicable to both virtual networks as well as datagram subnets. A choke packet is a packet sent by a node to the source to inform it of congestion. Each router monitors its resources and the utilization at each of its output lines. Whenever the resource utilization exceeds the threshold value which is set by the administrator, the router directly sends a choke packet to the source giving it a feedback to reduce the traffic. The intermediate nodes through which the packets has traveled are not warned about congestion



* **Implicit Signaling:**In implicit signaling, there is no communication between the congested nodes and the source. The source guesses that there is congestion in a network. For example when sender sends several packets and there is no acknowledgment for a while, one assumption is that there is a congestion.
* **Explicit Signaling:** In explicit signaling, if a node experiences congestion it can explicitly sends a packet to the source or destination to inform about congestion. The difference between choke packet and explicit signaling is that the signal is included in the packets that carry data rather than creating a different packet as in case of choke packet technique.

Explicit signaling can occur in either forward or backward direction.

* **Forward Signaling:** In forward signaling, a signal is sent in the direction of the congestion. The destination is warned about congestion. The receiver in this case adopt policies to prevent further congestion.
* **Backward Signaling:** In backward signaling, a signal is sent in the opposite direction of the congestion. The source is warned about congestion and it needs to slow down.

**Leaky bucket algorithm**

In the network layer, before the network can make Quality of service guarantees, it must know what traffic is being guaranteed. One of the main causes of congestion is that traffic is often bursty.

To understand this concept first we have to know little about traffic shaping. **Traffic Shaping** is a mechanism to control the amount and the rate of traffic sent to the network. Approach of congestion management is called Traffic shaping. Traffic shaping helps to regulate the rate of data transmission and reduces congestion.

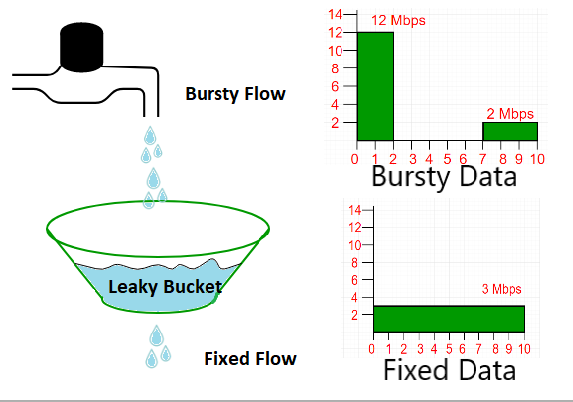
There are 2 types of traffic shaping algorithms:

1. Leaky Bucket
2. Token Bucket

**Leaky Bucket**

Suppose we have a bucket in which we are pouring water, at random points in time, but we have to get water at a fixed rate, to achieve this we will make a hole at the bottom of the bucket. This will ensure that the water coming out is at some fixed rate, and also if the bucket gets full, then we will stop pouring water into it.

The input rate can vary, but the output rate remains constant. Similarly, in networking, a technique called leaky bucket can smooth out bursty traffic. Bursty chunks are stored in the bucket and sent out at an average rate.



In the above figure, we assume that the network has committed a bandwidth of 3 Mbps for a host. The use of the leaky bucket shapes the input traffic to make it conform to this commitment. In the above figure, the host sends a burst of data at a rate of 12 Mbps for 2s, for a total of 24 Mbits of data. The host is silent for 5 s and then sends data at a rate of 2 Mbps for 3 s, for a total of 6 Mbits of data. In all, the host has sent 30 Mbits of data in 10 s. The leaky bucket smooths out the traffic by sending out data at a rate of 3 Mbps during the same 10 s.

Without the leaky bucket, the beginning burst may have hurt the network by consuming more bandwidth than is set aside for this host. We can also see that the leaky bucket may prevent congestion.

A simple leaky bucket algorithm can be implemented using FIFO queue. A FIFO queue holds the packets. If the traffic consists of fixed-size packets (e.g., cells in ATM networks), the process removes a fixed number of packets from the queue at each tick of the clock. If the traffic consists of variable-length packets, the fixed output rate must be based on the number of bytes or bits.

The following is an algorithm for variable-length packets:

1. Initialize a counter to n at the tick of the clock.
2. Repeat until n is smaller than the packet size of the packet at the head of the queue.
   1. Pop a packet out of the head of the queue, say P.
   2. Send the packet P, into the network
   3. Decrement the counter by the size of packet P.
3. Reset the counter and go to step 1.

***Note:****In the below examples, the head of the queue is the rightmost position and the tail of the queue is the leftmost position.*

**Example:**  Let n=1000

Packet=

leaky_algorithm_2

Since n > size of the packet at the head of the Queue, i.e. n > 200

Therefore, n = 1000-200 = 800

Packet size of 200 is sent into the network.

leaky_algorithm_2

Now, again n > size of the packet at the head of the Queue, i.e. n > 400

Therefore, n = 800-400 = 400

Packet size of 400 is sent into the network.

leaky_algorithm_2

Since, n < size of the packet at the head of the Queue, i.e.  n < 450

Therefore, the procedure is stopped.

Initialise n = 1000 on another tick of the clock.

This procedure is repeated until all the packets are sent into the network.

Below is the implementation of above explained approach:

**In C++**

// cpp program to implement leakybucket

#include <bits/stdc++.h>

using namespace std;

int main()

{

    int no\_of\_queries, storage, output\_pkt\_size;

    int input\_pkt\_size, bucket\_size, size\_left;

    // initial packets in the bucket

    storage = 0;

    // total no. of times bucket content is checked

    no\_of\_queries = 4;

    // total no. of packets that can

    // be accommodated in the bucket

    bucket\_size = 10;

    // no. of packets that enters the bucket at a time

    input\_pkt\_size = 4;

    // no. of packets that exits the bucket at a time

    output\_pkt\_size = 1;

    for (int i = 0; i < no\_of\_queries; i++) // space left

    {

        size\_left = bucket\_size - storage;

        if (input\_pkt\_size <= size\_left) {

            // update storage

            storage += input\_pkt\_size;

        }

        else

{

             printf("Packet loss = %d\n", input\_pkt\_size);

         }

        printf("Buffer size= %d out of bucket size= %d\n",

               storage, bucket\_size);

        storage -= output\_pkt\_size;

    }

    return 0;

}

**Output**   
Buffer size= 4 out of bucket size= 10

Buffer size= 7 out of bucket size= 10

Buffer size= 10 out of bucket size= 10

Packet loss = 4

Buffer size= 9 out of bucket size= 10

**In Python3**

# initial packets in the bucket

storage = 0

# total no. of times bucket content is checked

no\_of\_queries = 4

# total no. of packets that can

# be accommodated in the bucket

bucket\_size = 10

# no. of packets that enters the bucket at a time

input\_pkt\_size = 4

# no. of packets that exits the bucket at a time

output\_pkt\_size = 1

for i in range(0, no\_of\_queries):  # space left

    size\_left = bucket\_size - storage

    if input\_pkt\_size <= size\_left:

      # update storage

        storage += input\_pkt\_size

    else:

        print("Packet loss = ", input\_pkt\_size)

    print(f"Buffer size= {storage} out of bucket size = {bucket\_size}")

    # as packets are sent out into the network, the size of the storage decreases

    storage -= output\_pkt\_size

# This code is contributed by Arpit Jain

# Improved by: rishitchaudhary

**Output**Buffer size= 4 out of bucket size= 10

Buffer size= 7 out of bucket size= 10

Buffer size= 10 out of bucket size= 10

Packet loss = 4

Buffer size= 9 out of bucket size= 10

**Token Bucket Algorithm**

The leaky bucket algorithm enforces output patterns at the average rate, no matter how busy the traffic is. So, to deal with the more traffic, we need a flexible algorithm so that the data is not lost. One such approach is the token bucket algorithm.

Let us understand this algorithm step wise as given below −

**Step 1** − in regular intervals tokens are thrown into the bucket f.

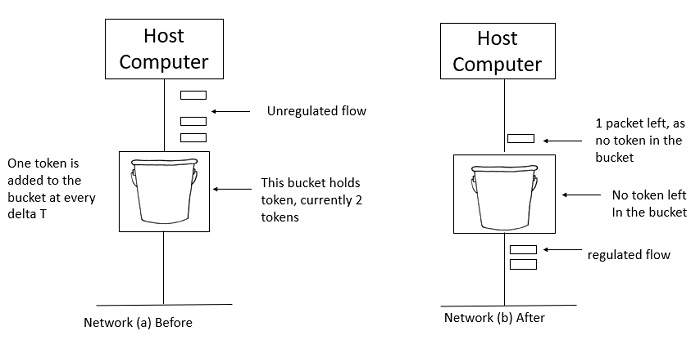
**Step 2** − the bucket has a maximum capacity f.

**Step 3** − If the packet is ready, then a token is removed from the bucket, and the packet is sent.

**Step 4** − suppose, if there is no token in the bucket, the packet cannot be sent.

**Example**

Let us understand the Token Bucket Algorithm with an example –



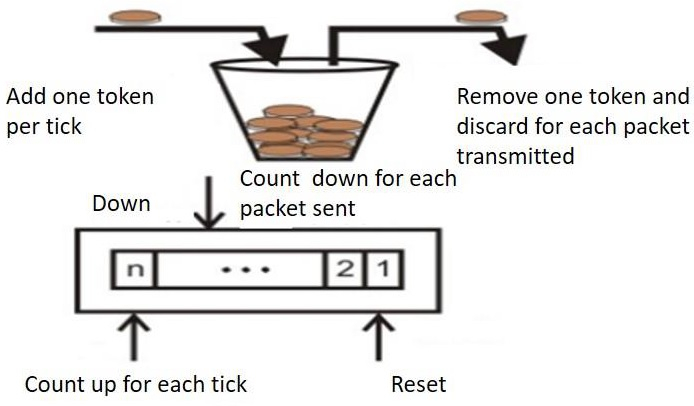
In figure (a) the bucket holds two tokens, and three packets are waiting to be sent out of the interface.

In Figure (b) two packets have been sent out by consuming two tokens, and 1 packet is still left.

When compared to Leaky bucket the token bucket algorithm is less restrictive that means it allows more traffic. The limit of busyness is restricted by the number of tokens available in the bucket at a particular instant of time.

The implementation of the token bucket algorithm is easy − a variable is used to count the tokens. For every t seconds the counter is incremented and then it is decremented whenever a packet is sent. When the counter reaches zero, no further packet is sent out.

This is shown in below given diagram −



**Difference between Leaky and Token buckets –**

| **Leaky Bucket** | **Token Bucket** |
| --- | --- |
| When the host has to send a packet, packet is thrown in bucket. | In this, the bucket holds tokens generated at regular intervals of time. |
| Bucket leaks at constant rate | Bucket has maximum capacity. |
| Bursty traffic is converted into uniform traffic by leaky bucket. | If there is a ready packet, a token is removed from Bucket and packet is send. |
| In practice bucket is a finite queue outputs at finite rate | If there is no token in the bucket, then the packet cannot be sent. |

**Some advantage of token Bucket over leaky bucket**

* If a bucket is full in tokens Bucket, tokens are discard not packets. While in leaky bucket, packets are discarded.
* Token Bucket can send large bursts at a faster rate while leaky bucket always sends packets at constant rate.
* **Predictable Traffic Shaping:**Token Bucket offers more predictable traffic shaping compared to leaky bucket. With token bucket, the network administrator can set the rate at which tokens are added to the bucket, and the maximum number of tokens that the bucket can hold. This allows for better control over the network traffic and can help prevent congestion.
* **Better Quality of Service (QoS):** Token Bucket provides better QoS compared to leaky bucket. This is because token bucket can prioritize certain types of traffic by assigning different token arrival rates to different classes of packets. This ensures that important packets are sent first, while less important packets are sent later, helping to ensure that the network runs smoothly.
* **More efficient use of network bandwidth:** Token Bucket allows for more efficient use of network bandwidth as it allows for larger bursts of data to be sent at once. This can be useful for applications that require high-speed data transfer or for streaming video content.
* **More granular control:**Token Bucket provides more granular control over network traffic compared to leaky bucket. This is because it allows the network administrator to set the token arrival rate and the maximum token count, which can be adjusted according to the specific needs of the network.
* **Easier to implement:** Token Bucket is generally considered easier to implement compared to leaky bucket. This is because token bucket only requires the addition and removal of tokens from a bucket, while leaky bucket requires the use of timers and counters to determine when to release packets.

**Some Disadvantage of token Bucket over leaky bucket**

* **Tokens may be wasted:** In Token Bucket, tokens are generated at a fixed rate, even if there is no traffic on the network. This means that if no packets are sent, tokens will accumulate in the bucket, which could result in wasted resources. In contrast, with leaky bucket, the network only generates packets when there is traffic, which helps to conserve resources.
* **Delay in packet delivery:** Token Bucket may introduce delay in packet delivery due to the accumulation of tokens. If the token bucket is empty, packets may need to wait for the arrival of new tokens, which can lead to increased latency and packet loss.
* **Lack of flexibility:**Token Bucket is less flexible compared to leaky bucket in terms of shaping network traffic. This is because the token generation rate is fixed, and cannot be changed easily to meet the changing needs of the network. In contrast, leaky bucket can be adjusted more easily to adapt to changes in network traffic.
* **Complexity:** Token Bucket can be more complex to implement compared to leaky bucket, especially when different token generation rates are used for different types of traffic. This can make it more difficult for network administrators to configure and manage the network.
* **Inefficient use of bandwidth:** In some cases, Token Bucket may lead to inefficient use of bandwidth. This is because Token Bucket allows for large bursts of data to be sent at once, which can cause congestion and lead to packet loss. In contrast, leaky bucket helps to prevent congestion by limiting the amount of data that can be sent at any given time.

**TCP Congestion Control**

TCP congestion control is a method used by the TCP protocol to manage data flow over a network and prevent congestion. TCP uses a congestion window and congestion policy that avoids congestion. Previously, we assumed that only the receiver could dictate the sender’s window size. We ignored another entity here, the network. If the network cannot deliver the data as fast as it is created by the sender, it must tell the sender to slow down. In other words, in addition to the receiver, the network is a second entity that determines the size of the sender’s window

**Congestion Policy in TCP**

1. **Slow Start Phase:** Starts slow increment is exponential to the threshold.
2. **Congestion Avoidance Phase:** After reaching the threshold increment is by 1.
3. **Congestion Detection Phase:** The sender goes back to the Slow start phase or the Congestion avoidance phase.

**Slow Start Phase**

**Exponential increment**: In this phase after every RTT the congestion window size increments exponentially.

**Example:-** If the initial congestion window size is 1 segment, and the first segment is successfully acknowledged, the congestion window size becomes 2 segments. If the next transmission is also acknowledged, the congestion window size doubles to 4 segments. This exponential growth continues as long as all segments are successfully acknowledged.

Initially cwnd = 1

After 1 RTT, cwnd = 2^(1) = 2

2 RTT, cwnd = 2^(2) = 4

3 RTT, cwnd = 2^(3) = 8

**Congestion Avoidance Phase**

**Additive increment:** This phase starts after the threshold value also denoted as ssthresh. The size of cwnd(congestion window) increases additive. After each RTT cwnd = cwnd + 1.

**Example:-** if the congestion window size is 20 segments and all 20 segments are successfully acknowledged within an RTT, the congestion window size would be increased to 21 segments in the next RTT. If all 21 segments are again successfully acknowledged, the congestion window size would be increased to 22 segments, and so on.

Initially cwnd = i

After 1 RTT, cwnd = i+1

2 RTT, cwnd = i+2

3 RTT, cwnd = i+3

**Congestion Detection Phase**

**Multiplicative decrement:** If congestion occurs, the congestion window size is decreased. The only way a sender can guess that congestion has happened is the need to retransmit a segment. Retransmission is needed to recover a missing packet that is assumed to have been dropped by a router due to congestion. Retransmission can occur in one of two cases: when the RTO timer times out or when three duplicate ACKs are received.

**Case 1:** Retransmission due to Timeout – In this case, the congestion possibility is high.

(a) ssthresh is reduced to half of the current window size.

(b) set cwnd = 1

(c) start with the slow start phase again.

**Case 2:**Retransmission due to 3 Acknowledgement Duplicates – The congestion possibility is less.

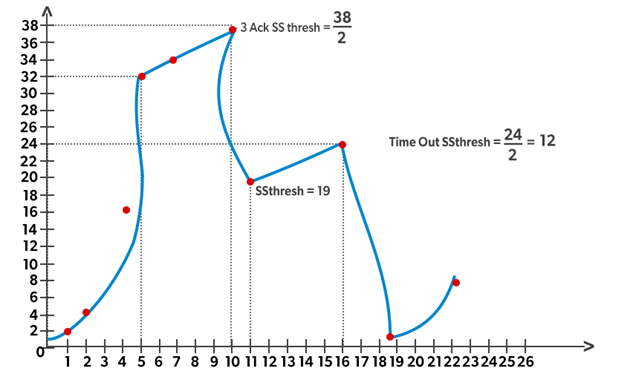
(a) ssthresh value reduces to half of the current window size.

(b) set cwnd= ssthresh

(c) start with congestion avoidance phase

**Example**

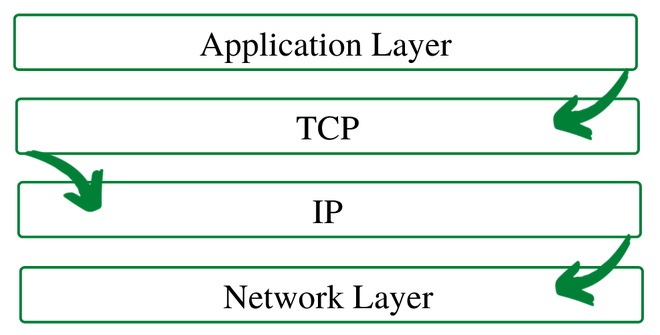
Assume a TCP protocol experiencing the behavior of slow start. At the 5th transmission round with a threshold (ssthresh) value of 32 goes into the congestion avoidance phase and continues till the 10th transmission. At the 10th transmission round, 3 duplicate ACKs are received by the receiver and entered into additive increase mode. Timeout occurs at the 16th transmission round. Plot the transmission round (time) vs congestion window size of TCP segments.



**Protocols of Transport Layer**

1. Transmission Control Protocol (TCP)
2. User Datagram Protocol (UDP)
3. Stream Control Transmission Protocol (SCTP)
4. Datagram Congestion Control Protocol (DCCP)
5. AppleTalk Transaction Protocol (ATP)
6. Fibre Channel Protocol (FCP)
7. Reliable Data Protocol (RDP)
8. Reliable User Data Protocol (RUDP)
9. Structured Steam Transport (SST)
10. Sequenced Packet Exchange (SPX)
11. Transmission Control Protocol (TCP):

TCP (Transmission Control Protocol) is one of the main protocols of the Internet protocol suite. It lies between the Application and Network Layers which are used in providing reliable delivery services. It is a connection-oriented protocol for communications that helps in the exchange of messages between different devices over a network. The Internet Protocol (IP), which establishes the technique for sending data packets between computers, works with TCP.

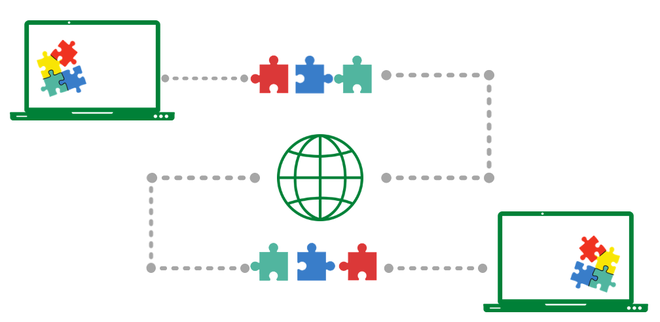


*TCP/IP Layer*

**Working of TCP**

To make sure that each message reaches its target location intact, the TCP/IP model breaks down the data into small bundles and afterward reassembles the bundles into the original message on the opposite end. Sending the information in little bundles of information makes it simpler to maintain efficiency as opposed to sending everything in one go.

After a particular message is broken down into bundles, these bundles may travel along multiple routes if one route is jammed but the destination remains the same.



*We can see that the message is being broken down, and then reassembled from a different order at the destination*

*For* ***example****,* when a user requests a web page on the internet, somewhere in the world, the server processes that request and sends back an HTML Page to that user. The server makes use of a protocol called the HTTP Protocol. The HTTP then requests the TCP layer to set the required connection and send the HTML file.

Now, the TCP breaks the data into small packets and forwards it toward the Internet Protocol (IP) layer. The packets are then sent to the destination through different routes.

The TCP layer in the user’s system waits for the transmission to get finished and acknowledges once all packets have been received.

**Features of TCP/IP**

Some of the most prominent features of Transmission control protocol are

1. **Segment Numbering System**

* TCP keeps track of the segments being transmitted or received by assigning numbers to each and every single one of them.
* A specific *Byte Number* is assigned to data bytes that are to be transferred while segments are assigned *sequence numbers*.
* *Acknowledgment Numbers* are assigned to received segments.

1. **Connection Oriented**

* It means sender and receiver are connected to each other till the completion of the process.
* The order of the data is maintained i.e. order remains same before and after transmission.

1. **Full Duplex**

* In TCP data can be transmitted from receiver to the sender or vice – versa at the same time.
* It increases efficiency of data flow between sender and receiver.

1. **Flow Control**

* Flow control limits the rate at which a sender transfers data. This is done to ensure reliable delivery.
* The receiver continually hints to the sender on how much data can be received (using a sliding window)

1. **Error Control**

* TCP implements an error control mechanism for reliable data transfer
* Error control is byte-oriented
* Segments are checked for error detection
* Error Control includes – *Corrupted Segment & Lost Segment Management, Out-of-order segments, Duplicate segments*, etc.

1. **Congestion Control**

* TCP takes into account the level of congestion in the network
* Congestion level is determined by the amount of data sent by a sender

**Advantages**

1. It is a reliable protocol.
2. It provides an error-checking mechanism as well as one for recovery.
3. It gives flow control.
4. It makes sure that the data reaches the proper destination in the exact order that it was sent.
5. Open Protocol, not owned by any organization or individual.
6. It assigns an IP address to each computer on the network and a domain name to each site thus making each device site to be distinguishable over the network.

**Disadvantages**

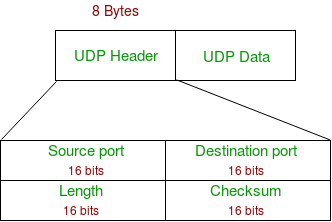
1. TCP is made for Wide Area Networks, thus its size can become an issue for small networks with low resources.
2. TCP runs several layers so it can slow down the speed of the network.
3. It is not generic in nature. Meaning, it cannot represent any protocol stack other than the TCP/IP suite. E.g., it cannot work with a Bluetooth connection.
4. No modifications since their development around 30 years ago.
5. User Datagram Protocol (UDP)

**User Datagram Protocol (UDP)** is a Transport Layer protocol. UDP is a part of the Internet Protocol suite, referred to as UDP/IP suite. Unlike TCP, it is an **unreliable and connectionless protocol.** So, there is no need to establish a connection prior to data transfer. The UDP helps to establish low-latency and loss-tolerating connections establish over the network. The UDP enables process to process communication.

Though Transmission Control Protocol (TCP) is the dominant transport layer protocol used with most of the Internet services; provides assured delivery, reliability, and much more but all these services cost us additional overhead and latency. Here, UDP comes into the picture. For real-time services like computer gaming, voice or video communication, live conferences; we need UDP. Since high performance is needed, UDP permits packets to be dropped instead of processing delayed packets. There is no error checking in UDP, so it also saves bandwidth.  User Datagram Protocol (UDP) is more efficient in terms of both latency and bandwidth.

**UDP Header –**

UDP header is an **8-bytes** fixed and simple header, while for TCP it may vary from 20 bytes to 60 bytes. The first 8 Bytes contains all necessary header information and the remaining part consist of data. UDP port number fields are each 16 bits long, therefore the range for port numbers is defined from 0 to 65535; port number 0 is reserved. Port numbers help to distinguish different user requests or processes.



1. **Source Port:** Source Port is a 2 Byte long field used to identify the port number of the source.
2. **Destination Port:** It is a 2 Byte long field, used to identify the port of the destined packet.
3. **Length:** Length is the length of UDP including the header and the data. It is a 16-bits field.
4. **Checksum:** Checksum is 2 Bytes long field. It is the 16-bit one’s complement of the one’s complement sum of the UDP header, the pseudo-header of information from the IP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

**Notes –** Unlike TCP, the Checksum calculation is not mandatory in UDP. No Error control or flow control is provided by UDP. Hence UDP depends on IP and ICMP for error reporting. Also UDP provides port numbers so that is can differentiate between users requests.

**Applications of UDP:**

* Used for simple request-response communication when the size of data is less and hence there is lesser concern about flow and error control.
* It is a suitable protocol for multicasting as UDP supports packet switching.
* UDP is used for some routing update protocols like RIP(Routing Information Protocol).
* Normally used for real-time applications which can not tolerate uneven delays between sections of a received message.
* UDP is widely used in online gaming, where low latency and high-speed communication is essential for a good gaming experience. Game servers often send small, frequent packets of data to clients, and UDP is well suited for this type of communication as it is fast and lightweight.
* Streaming media applications, such as IPTV, online radio, and video conferencing, use UDP to transmit real-time audio and video data. The loss of some packets can be tolerated in these applications, as the data is continuously flowing and does not require retransmission.
* VoIP (Voice over Internet Protocol) services, such as Skype and WhatsApp, use UDP for real-time voice communication. The delay in voice communication can be noticeable if packets are delayed due to congestion control, so UDP is used to ensure fast and efficient data transmission.
* DNS (Domain Name System) also uses UDP for its query/response messages. DNS queries are typically small and require a quick response time, making UDP a suitable protocol for this application.
* DHCP (Dynamic Host Configuration Protocol) uses UDP to dynamically assign IP addresses to devices on a network. DHCP messages are typically small, and the delay caused by packet loss or retransmission is generally not critical for this application.
* Following implementations uses UDP as a transport layer protocol:
* NTP (Network Time Protocol)
* DNS (Domain Name Service)
* BOOTP, DHCP.
* NNP (Network News Protocol)
* Quote of the day protocol
* TFTP, RTSP, RIP.
* The application layer can do some of the tasks through UDP-
* Trace Route
* Record Route
* Timestamp
* UDP takes a datagram from Network Layer, attaches its header, and sends it to the user. So, it works fast.
* Actually, UDP is a null protocol if you remove the checksum field.
* Reduce the requirement of computer resources.
* When using the Multicast or Broadcast to transfer.
* The transmission of Real-time packets, mainly in multimedia applications.

**Advantages of UDP:**

1. Speed: UDP is faster than TCP because it does not have the overhead of establishing a connection and ensuring reliable data delivery.
2. Lower latency: Since there is no connection establishment, there is lower latency and faster response time.
3. Simplicity: UDP has a simpler protocol design than TCP, making it easier to implement and manage.
4. Broadcast support: UDP supports broadcasting to multiple recipients, making it useful for applications such as video streaming and online gaming.
5. Smaller packet size: UDP uses smaller packet sizes than TCP, which can reduce network congestion and improve overall network performance.

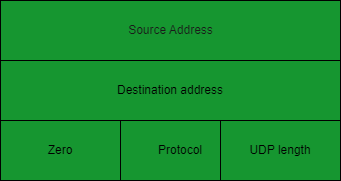
**Disadvantages of UDP:**

1. No reliability: UDP does not guarantee delivery of packets or order of delivery, which can lead to missing or duplicate data.
2. No congestion control: UDP does not have congestion control, which means that it can send packets at a rate that can cause network congestion.
3. No flow control: UDP does not have flow control, which means that it can overwhelm the receiver with packets that it cannot handle.
4. Vulnerable to attacks: UDP is vulnerable to denial-of-service attacks, where an attacker can flood a network with UDP packets, overwhelming the network and causing it to crash.
5. Limited use cases: UDP is not suitable for applications that require reliable data delivery, such as email or file transfers, and is better suited for applications that can tolerate some data loss, such as video streaming or online gaming.

**UDP PSEUDO HEADER:**

The purpose of using a pseudo-header is to verify that the UDP packet has reached its correct destination

The correct destination consist of a specific machine and a specific protocol port number within that machine



**UDP pseudo header details:**

* The UDP header itself specify only protocol port number.thus , to verify the destination UDP on the sending machine computes a checksum that covers the destination IP address as well as the UDP packet.
* At the ultimate destination, UDP software verifies the checksum using the destination IP address obtained from the header of the IP packet that carried the UDP message.
* If the checksum agrees, then it must be true that the packet has reached the intended destination host as well as the correct protocol port within that host.

**User Interface:**

A user interface should allow the creation of new receive ports, receive operations on the receive ports that returns the data octets and an indication of source port and source address, and an operation that allows a datagram to be sent, specifying the data, source and destination ports and address to be sent.

**IP Interface:**

1. The UDP module must be able to determine the source and destination internet address and the protocol field from internet header
2. one possible UDP/IP interface would return the whole internet datagram including the entire internet header in response to a receive operation
3. such an interface would also allow the UDP to pass a full internet datagram complete with header to the IP to send. the IP would verify certain fields for consistency and compute the internet header checksum.
4. The IP interface allows the UDP module to interact with the network layer of the protocol stack, which is responsible for routing and delivering data across the network.
5. The IP interface provides a mechanism for the UDP module to communicate with other hosts on the network by providing access to the underlying IP protocol.
6. The IP interface can be used by the UDP module to send and receive data packets over the network, with the help of IP routing and addressing mechanisms.
7. The IP interface provides a level of abstraction that allows the UDP module to interact with the network layer without having to deal with the complexities of IP routing and addressing directly.
8. The IP interface also handles fragmentation and reassembly of IP packets, which is important for large data transmissions that may exceed the maximum packet size allowed by the network.
9. The IP interface may also provide additional services, such as support for Quality of Service (QoS) parameters and security mechanisms such as IPsec.
10. The IP interface is a critical component of the Internet Protocol Suite, as it enables communication between hosts on the internet and allows for the seamless transmission of data packets across the network.
11. Stream Control Transmission Protocol (SCTP)

SCTP stands for **Stream Control Transmission Protocol**.

It is a connection- oriented protocol in computer networks which provides a full-duplex association i.e., transmitting multiple streams of data between two end points at the same time that have established a connection in network. It is sometimes referred to as next generation TCP or TCPng, SCTP makes it easier to support telephonic conversation on Internet. A telephonic conversation requires transmitting of voice along with other data at the same time on both ends, SCTP protocol makes it easier to establish reliable connection.

SCTP is also intended to make it easier to establish connection over wireless network and managing transmission of multimedia data. SCTP is a standard protocol (RFC 2960) and is developed by Internet Engineering Task Force (IETF).

**Characteristics of SCTP**

1. **Unicast with Multiple properties** –  It is a point-to-point protocol which can use different paths to reach end host.
2. **Reliable Transmission** – It uses SACK and checksums to detect damaged, corrupted, discarded, duplicate and reordered data. It is similar to TCP but SCTP is more efficient when it comes to reordering of data.
3. **Message oriented –** Each message can be framed and we can keep order of datastream and tabs on structure. For this, In TCP, we need a different layer for abstraction.
4. **Multi-homing** – It can establish multiple connection paths between two end points and does not need to rely on IP layer for resilience.
5. **Security** – Another characteristic of SCTP that is  security. In SCTP, resource allocation for association establishment only takes place following cookie exchange identification verification for the client (INIT ACK). Man-in-the-middle and denial-of-service attacks are less likely as a result. Furthermore, SCTP doesn’t allow for half-open connections, making it more resistant to network floods and masquerade attacks.

**Advantages of SCTP:**

* It is a full- duplex connection i.e. users can send and receive data simultaneously.
* It allows half- closed connections.
* The message’s boundaries are maintained and application doesn’t have to split messages.
* It has properties of both TCP and UDP protocol.
* It doesn’t rely on IP layer for resilience of paths.

**Disadvantages of SCTP:**

* One of key challenges is that it requires changes in transport stack on node.
* Applications need to be modified to use SCTP instead of TCP/UDP.
* Applications need to be modified to handle multiple simultaneous streams.

1. Datagram Congestion Control Protocol (DCCP)

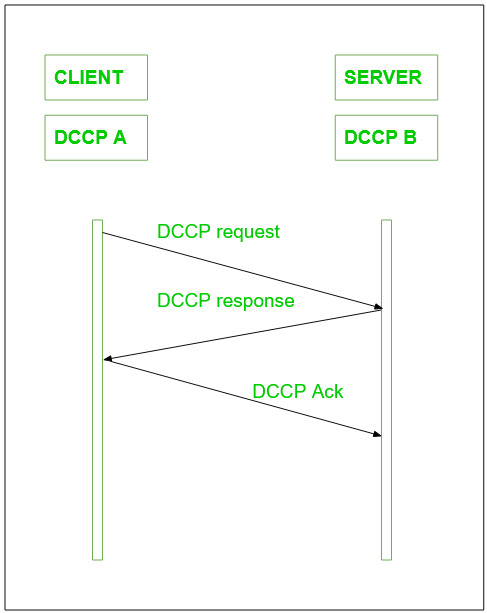
Congestion in network means deterioration of network or services which are caused due to overloading of network nodes, basically, this problem is primarily associated with large networks, in which a large amount of data and information is being transmitted. Congestion can be caused by several reasons: either the routers which are being used are not fast enough, the CPUs which are being used are not fast enough and they do not manage to quit queues in OS in a timely manner, buffers are not large enough as our requirements or they are lost from the packets. Also in the case of very high traffic, the situation can be worse enough that no packages are delivered at all.

**DCCP** is basically a message-based transport-level protocol. The setting of a secure connection is easily maintained using it, its closure i.e. ECN (Explicit Congestion Notification), congestion control, and negotiation of features. DCCP is a great technique to access congestion control mechanisms, also we don’t need to implement them at the application level also.

DCCP basically allows similar Transfer Control Protocol feeds also, but delivery in the order of transmission cannot be done. Sequential delivery of multiple streams (as in SCTP- Stream Control Transmission Protocol) is not available in DCCP.

DCCP is widely used in applications package delivery is composed of time constraints. The examples that come under this category include multiplayer online games, internet telephony, streaming media (video, audio), etc. The most important feature of these applications is that old messages quickly become expired automatically, lose their usefulness by default.

DCCP connection setup can be explained through the below image, it is basically similar to TCP connection setup:



*DCCP Connection setup*

On the other hand, the higher priority is given to new messages, so to resend the packets is not very much useful here, it would eventually consume time and unnecessary network resources as well. Datagram Congestion Control Protocol can also be used as a general congestion control technique for those types of applications that are based on the UDP protocol as well. A safety mechanism can also be added and possibly one for packet delivery in the order of transmission. In other cases, DCCP helps to use various congestion control mechanisms, generally Transmission Control Protocol-friendly. Confirmation traffic and data traffic are both contained in a DCCP connection.

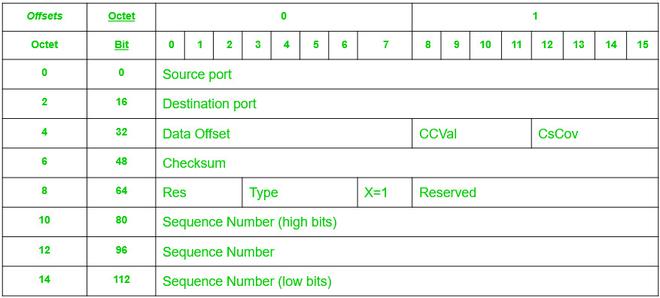
The transmitter gets to know with the help of confirmations that his packages have arrived at the destination or have been marked by ECN. Confirmations are used with the purpose of safety demanded by the congestion control mechanism. Its primary aim is to reach 100% safely.

**DCCP Packet structure:**

The DCCP generic header has various forms according to the value given to X i.e. the Extended Sequence Numbers bit.

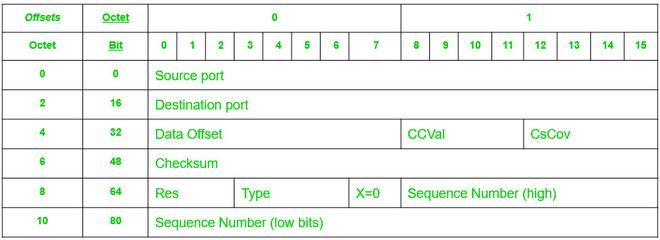
Let X = 1,

the Sequence Number field is 48 bits long, and the generic header takes 16 bytes, which is clearly explained in the below image.



*DCCP generic header when X=1*

If we change the value of X = 0, only the low 24 bits of the Sequence Number are transmitted, and the generic header is 12 bytes long which is shown in the below image :



*DCCP generic header when X=0*

**Features of DCCP:**

1. DCCP is a non-reliable datagram stream, with a good feature of confirmation.
2. DCCP helps to secure negotiation of options, including negotiation of the most suitable mechanism for congestion control.
3. It provides a secure handshake protocol with the purpose of initializing and closing the connection of DCCP.
4. It plays a vital role in the discovery of the maximum transmitting unit on the chosen path by the user.
5. It provides techniques that allow servers to avoid storing states for attempted unconnected, unconfirmed disconnections, or for already closed connections as well.
6. Confirmation mechanisms are a very good feature of DCCP which helps to communicate packet loss and ECN information.
7. Optional mechanisms are also some good techniques that communicate to the emitting application with high security, which packets have reached the receiver and which are not, also whether they have been marked by ECN or not, or they are corrupted or removed in the receiver buffer.
8. DCCP can support multiple concurrent streams within a single connection, which enables applications to transmit multiple data flows over the same connection.
9. It provides a mechanism for applications to prioritize their data flows, which helps in achieving better Quality of Service (QoS).
10. DCCP supports both connection-oriented and connectionless communication modes.
11. It offers a congestion control mechanism that is more flexible than the TCP congestion control mechanism.
12. DCCP can be used over both IP version 4 and IP version 6 networks.

**Advantages**

* **Congestion control:**Unlike UDP, which has no built-in mechanism for controlling congestion, DCCP includes congestion control algorithms that help to prevent network overload and ensure reliable delivery of data.
* **Quality of Service (QoS) support:**DCCP provides support for QoS, which allows network administrators to prioritize different types of traffic based on their importance. This can be useful for applications such as video streaming or voice over IP, where low latency and high reliability are essential.
* **Flexibility:** DCCP is designed to be a flexible protocol, allowing network administrators to choose from a variety of congestion control algorithms based on the specific requirements of their network and applications.
* **Compatibility:** DCCP is designed to work with existing IP networks and is compatible with traditional IP protocols like TCP and UDP.

**Real-World Applications**

* **Streaming media:**DCCP is often used in streaming media applications, such as video conferencing, where low latency and high reliability are important. The congestion control algorithms built into DCCP help to ensure that these applications can run smoothly and effectively, even in networks with high levels of congestion.
* **Gaming:**Some online gaming applications also make use of DCCP, as its congestion control algorithms can help to prevent network slowdowns and ensure that game data is delivered quickly and reliably.
* **Telemetry:**DCCP is also used in telemetry applications, where large amounts of data need to be transmitted from remote devices back to a central control center. The congestion control algorithms built into DCCP help to ensure that this data is delivered reliably and efficiently, even in congested networks.
* **Remote Access:**DCCP can be used for remote access applications, as it provides a reliable and secure connection for remote access to servers and other resources.

1. AppleTalk Transaction Protocol (ATP)

ATP stands for AppleTalk Transaction Protocol. It is a part of a series of networking protocols called the ‘AppleTalk’ developed by Apple Inc. It is a transport layer protocol that lets you transfer small amounts of data across a network. It provides an error-free and reliable means of communication in a client-server setup. It performs most of the transport layer functions like segmentation, packet sequencing, etc.

ATP works on top of an architecture that is similar to a client-server architecture. There is a requester that makes a request to another endpoint which is called a responder. The responder performs the needful and reverts back with a response.

**Transactions in ATP:**

The following diagram describes how a transaction occurs:



*Transactions in ATP*

1. The requester invokes the ATP interface. The interface then uses the ATP driver to establish a dedicated connection with the responder’s ATP driver.
2. The driver at the responder side invokes the responder application which processes the incoming request and returns a response to be sent back to the requester.
3. In order to send the response, the responder invokes its ATP interface which uses the pre-established connection to send the response back to the requester.

As mentioned before, ATP can be used to send **small** amounts of data. There is a restriction on the amount of data that can be transmitted by both the requester and the responder. For the requester, the data that can be transmitted is 578 bytes and for the responder, the limit is 4624 bytes.

**Types of ATP Transactions:**

There are two types of ATP transactions:

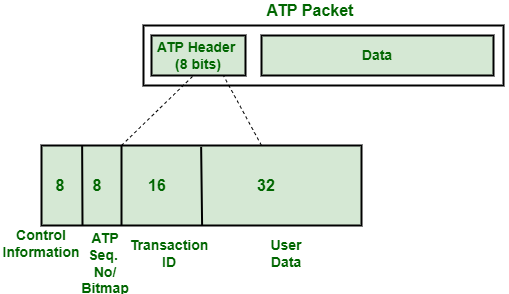
1. **At-Least-Once Transactions**: In an ALO transaction, it is guaranteed that the responder will receive all the requests sent to it at least once. It offers no guarantee that the same request won’t be sent again to the responder.
2. **eXactly-Once Transactions**: In an XO transaction, it is guaranteed that the responder will receive all the requests sent to it exactly once. It guarantees that a request that has been received once by the responder won’t be sent again.

The nature of the transaction to be used is determined by the kind of processing required by the request. If multiple executions of the same request are likely to cause some kind of inconsistency in the results obtained, then XO transactions are used. If multiple executions of the request are not likely to cause problems then ALO transactions are used.

**ATP Packet Format:**

An ATP packet consists of an ATP header followed by the data. The size of the ATP header is 8 bytes. The entire packet is stored inside a DDP (Datagram Delivery Protocol) datagram and is transported as a data link layer frame.

Following is the structure of an ATP packet:



*Structure of an ATP Packet*

The data part follows the ATP header which can be up to 578 bytes or up to 4624 bytes depending on whether the packet is being sent by the requester side or the responder side.

Following is a detailed description of the fields of the **ATP header:**

**Control Information Byte:**

It is used to store control-related information about a request/response like the type of transaction, whether this packet is the last packet to be received or not etc. The control information byte stores the following information –

|  |  |
| --- | --- |
| **Bit No.** | **Description** |
| **0** | Specifies whether to use the DDP checksum or not. |
| **1** | Specifies whether the transaction ID has been assigned for this request or not |
| **2** | Specifies whether this request uses an extended parameter block or not |
| **3** | Specifies whether the current packet is to be retransmitted immediately or not. |
| **4** | Specifies whether this is the last packet of the response (end-of-message) or not. |

**Bitmap/Sequence Number:**

The Bitmap/Sequence Number is 8 bits in size. ATP is well-known for the reliability it offers during data transmission. It guarantees that the requester will receive all the packets sent to it in response to its request. In order to do so, it needs to track which packets have been received by the requester and which are yet to be received. The approach used by ATP for the same is to assign sequence numbers to all the packets which are sent in response and to mark the last packet of response as “end-of-message” (indicated by bit 4 of the control information byte in ATP header). The requester assembles all the packets on receiving them, checks the sequence number and the control information byte of the ATP header, and determines whether it has received all the packets or not. If not, then the ATP driver on the receiver’s end initiates retransmission of the lost packets.

**Transaction ID:**

The transaction ID is used to associate each request with its corresponding response. It is 16 bits in size. The requester can make multiple requests at any given time. Transaction ID helps ATP to keep track of all the requests made and to ensure that the response delivered for every request is correct. It is carried by the 3rd and the 4th byte of the ATP header.

**User Data:**

The last 4 bytes of the ATP header are not used by ATP. They are reserved for use by the requester and responder applications. The ATP driver supports pre-defined functions that let you set the values of these headers to what you want.

The following table summarizes the fields that are part of the ATP header:

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte  No.** | **Field Stored** | **Size     (Bits)** | **Description** |
| **1** | Control Information | 8 | Is used to store control-related information about a request/response like the type of transaction, whether this packet is the last packet to be received or not etc. |
| **2** | Transaction Bitmap/ ATP Sequence Number | 8 | It is used to find the sequence number of the current packet if the packet was sent in response to a request(called ATP sequence number). In case the current packet is a request, it represents the number of buffers supported by the application (called Transaction Bitmap). |
| **3-4** | Transaction ID | 16 | It stores the transaction ID of every request. It is used by the responder to keep track of all the requests. |
| **5-8** | User Data | 32 | Carries application-specific information like checksum etc. |

1. [Fibre Channel Protocol (FCP)](https://www.geeksforgeeks.org/fcp-fibre-channel-protocol/)

Protocol means a set of instructions or rules and regulations. So, instead of first learning about fiber channel protocol, let’s get into what a fiber channel is:

**Fibre Channel** is a high-speed networking technology primarily used for transmitting data among data centers, computer servers, switches, and storage at data rates of up to 128 Gbps with distances up to 10Km.

Fibre Channel Protocol (FCP) is the SCSI (Small Computer System Interface) interface protocol operating on an established Fibre Channel connection. As Fibre Channel provides us with a high-speed data transfer service it can be used to connect workstations, mainframes, displays, storage devices, and supercomputers. The FCP provides one standardized way for storage, data transfer, and networking as the main task of the FCP is to ensure the successful transfer of large and bulky information/ data so that the manufacturers can easily support a variety of channels and networks.

The Fibre Channel protocol, also known as FC, is a method for transferring data serially over copper or optical fiber in order to achieve lower latency and faster speeds. It is a SCSI interface protocol that utilizes Fibre Channel connections. This protocol is used to connect high-performance computers, storage devices, mainframes, big data workstations, and displays as virtual big data structured screens.

So, Fibre Channel is primarily used to connect computer data storage to servers in storage area networks (SAN) in commercial data centers. Though fiber channel mainly runs on optical fiber cables it is also capable of transmitting over copper cables. As mentioned earlier, fiber channels can transmit data up to 128 Gbps (Gigabits per second) hence the alternate name Gigabit Fibre Channel (GFC).

* **FCP Topologies:**
* DAS (Direct Attached Storage)
* NAS (Network Attached Storage)
* SAN (Storage Area Network)
* **FCP Ports:**
* N Port (The Node Port)
* F Port (The Fabric Port)
* L Port (The Loop Port)
* FL Port (The Fabric Loop Port)
* E Port (The Extension Port)
* G Port (The Generic Port)
* GL Port (The Generic Loop Port)
* SL Port (The Segmented Loop Port)
* TL Port (The Translated Loop Port)
* T Port (The Trunk Port)

There are two main protocols for fiber channels with regard to block storage:

1. Fibre channel protocol (FCP): covered in the article
2. FICon (Fibre Connection) is a protocol that transports ESCon (Enterprise Systems Connection)  commands, used by IBM mainframe computers, over Fibre Channel.

**FCP Features Fibre Channel:**

* Data transfer speed of up to 128Gbps over a distance of 10Km.
* Both Fiber optic cable and copper cables can be used.
* FCP is used to transmit SCSI (Small Computer System Interface) commands over a Fibre Channel Network (FCN)
* The Fibre Channel Protocol (FCP) is an original protocol used in Storage Area Network (SAN).
* SFP (Small Form-factor Pluggable) connectors are used to facilitate a reliable, wired, high-speed connection.
* The Fibre Channel Protocol (FCP) offers a bandwidth range of 100 MB/s to 1.6 GB/s and can support distances of up to 500 meters to 10 kilometers.
* FCP operates similarly to both TCP and UDP protocols.
* The Fibre Channel Protocol (FCP) is both reliable and stable, with a balanced design.
* In Fibre Channel Protocol (FCP), World Wide Names (WWN) are used for addressing.
* These 8-byte addresses consist of 16 hexadecimal characters.
* The Fibre Channel Protocol (FCP) uses a format such as 15:00:00:f0:8c:95:de.
* Ability to carry multiple existing interface command sets, including Internet Protocol (IP), SCSI, IPI, HIPPI-FP, and audio/video.
* Support for multiple cost/performance levels, from small systems to supercomputers.
* In Fibre Channel Protocol (FCP), a dedicated host bus adapter, specialized cables, and switches are used. It is distinct from Ethernet at all layers of the OSI model, including the physical layer

**World Wide Name (WWN):**

A **World Wide Name (WWN) or World Wide Identifier (WWID)** is a unique identifier used in storage technologies like Fibre Channel. It is a unique identifier that is hard-coded into each Fibre similar to how devices have MAC Addresses. It is a 64-bit or 128-bit name and is assigned by the**Institute of Electrical and Electronics Engineers IEEE**. Each network storage device that a manufacturer produces must include the manufacturer’s WWN, in order to help system administrators (SA) uniquely categorize and identify storage segments.

The WWN looks for example:

**15:00:00:f0:8c:08:95:de**

**Types of World Wide Name (WWN):**

There are majorly two types of WWNs implemented in an FC Storage Area Network (SAN):

1. **World Wide Node Name (WWNN):**A World Wide Node Name, WWNN, or WWnN, is a World Wide Name assigned to a node (an endpoint, a device) in a Fibre Channel fabric.
2. **World Wide Port Name (WWPN):**a World Wide Port Name, WWPN, or WWpN, is a World Wide Name assigned to a port in a Fibre Channel fabric. In order to behave as a unique identifier in the network, it works similarly to the MAC address in Ethernet protocol.

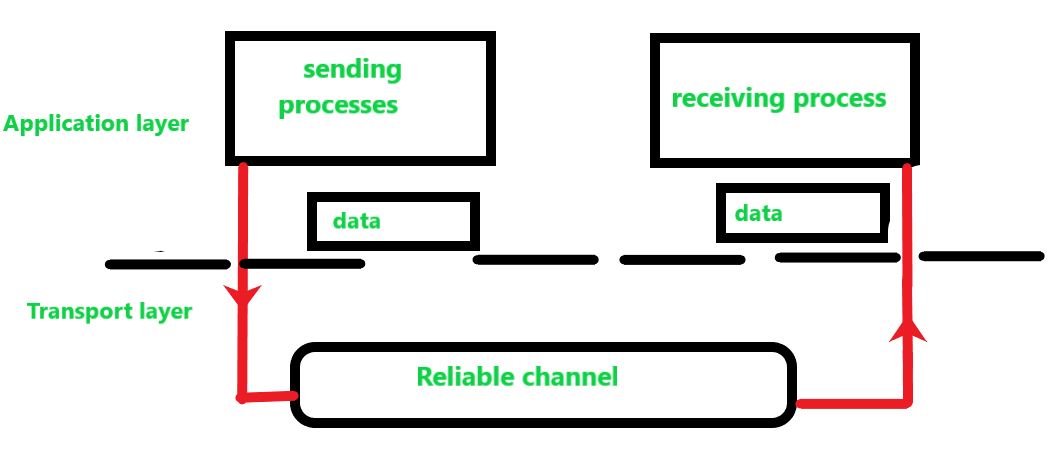
Fibre Channel Protocol (FCP) uses World Wide Node Names (WWNN) to identify nodes in data storage networks. These names can identify multiple network interfaces on a single node. The WWPN (World Wide Port Name) can also be derived from the WWNN.

* In Fibre Channel Protocol (FCP), a unique World Wide Port Name (WWPN) is assigned to every individual port on a node.
* In Fibre Channel Protocol (FCP), a multiport Host Bus Adapter (HBA) will have a different number of World Wide Port Names (WWPNs) for each port. WWPNs are similar to the MAC address in Ethernet networks.
* In Fibre Channel Protocol (FCP), World Wide Port Names (WWPN) are burn-in by the manufacturer. They are validated to be globally unique.
* In Fibre Channel Protocol (FCP), World Wide Port Names (WWPNs) are assigned to Host Bus Adapters (HBAs) on both client and storage systems.
* In Fibre Channel Protocol (FCP), World Wide Port Names (WWPNs) are given more importance when configuring Fibre Channel Networks, compared to World Wide Node Names (WWNNs).
* In Fibre Channel Protocol (FCP), World Wide Port Names (WWPNs) cannot be changed once assigned.
* **Advantages of Fibre Channel:**
* FCP has high performance
* provides good backup and restoration and simplified consolidation
* also offers congestion-free data flow, Gigabit bandwidth, compatibility with multiple topologies and protocols, flow control, and self-management
* It is providing high-speed data transfer
* It is cost-efficient
* High-speed data can be transferred over a distance of 10km
* supports several fault-tolerant features
* FCP having good bandwidth and speed
* The FCP protocol utilizes data frames for transmitting information over a network, which can be used for both link-level and device-level communications
* FCP having good Flow Control
* The Fibre Channel protocol (FCP) is known for its balanced and reliable nature, providing stable communication between devices
* FCP is a balanced and reliable protocol that is used to transmit SCSI (Small Computer System Interface) commands over Fibre Channel Networks (FCN)
* **Disadvantage of Fibre Channel:**
* FCP can be more expensive in cost compared to iSCSI
* FCP can be more costly and complex to implement
* FCP requires updating the cards within servers
* also purchasing FC cables and switches
* More expensive than SCSI (Small Computer System Interface)
* More complex than SCSI (Small Computer System Interface)
* More equipment/ overhead (like FC cables, switches, etc.,) is required

1. Reliable Data Protocol (RDP)

Transport Layer Protocols are central piece of layered architectures, these provides the logical communication between application processes. These processes uses the logical communication to transfer data from transport layer to network layer and this transfer of data should be reliable and secure. The data is transferred in the form of packets but the problem occurs in reliable transfer of data.

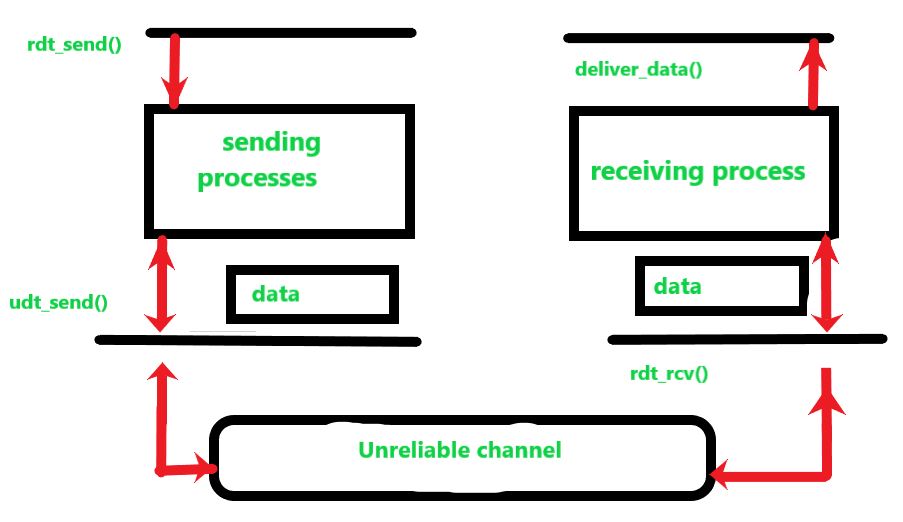
The problem of transferring the data occurs not only at the transport layer, but also at the application layer as well as in the link layer. This problem occur when a reliable service runs on an unreliable service, For example, TCP (Transmission Control Protocol) is a reliable data transfer protocol that is implemented on top of an unreliable layer, i.e., Internet Protocol (IP) is an end to end network layer protocol.



**Figure:** Study of Reliable Data Transfer

In this model, we have design the sender and receiver sides of a protocol over a reliable channel. In the reliable transfer of data the layer receives the data from the above layer breaks the message in the form of segment and put the header on each segment and transfer. Below layer receives the segments and remove the header from each segment and make it a packet by adding to header.

The data which is transferred from the above has no transferred data bits corrupted or lost, and all are delivered in the same sequence in which they were sent to the below layer this is reliable data transfer protocol. This service model is offered by TCP to the Internet applications that invoke this transfer of data.



**Figure:** Study of Unreliable Data Transfer

Similarly in an unreliable channel we have design the sending and receiving side. The sending side of the protocol is called from the above layer to rdt\_send() then it will pass the data that is to be delivered to the application layer at the receiving side (here rdt-send() is a function for sending data where rdt stands for reliable data transfer protocol and \_send() is used for the sending side).

On the receiving side, rdt\_rcv() (rdt\_rcv() is a function for receiving data where -rcv() is used for receiving side), will be called when a packet arrives from the receiving side of the unreliable channel. When the rdt protocol wants to deliver data to the application layer, it will do so by calling deliver\_data() (where deliver\_data() is a function for delivering data to upper layer).

In reliable data transfer protocol, we only consider the case of unidirectional data transfer, that is transfer of data from the sending side to receiving side(i.e. only in one direction). In case of bidirectional (full duplex or transfer of data on both the sides) data transfer is conceptually more difficult. Although we only consider unidirectional data transfer but it is important to note that the sending and receiving sides of our protocol will needs to transmit packets in both directions, as shown in above figure.

In order to exchange packets containing the data that is needed to be transferred the both (sending and receiving) sides of rdt also need to exchange control packets in both direction (i.e., back and forth), both the sides of rdt send packets to the other side by a call to udt\_send() (udt\_send() is a function used for sending data to other side where udt stands for unreliable data transfer protocol).

1. [Reliable User Data Protocol (RUDP)](https://www.geeksforgeeks.org/reliable-user-datagram-protocol-rudp/)

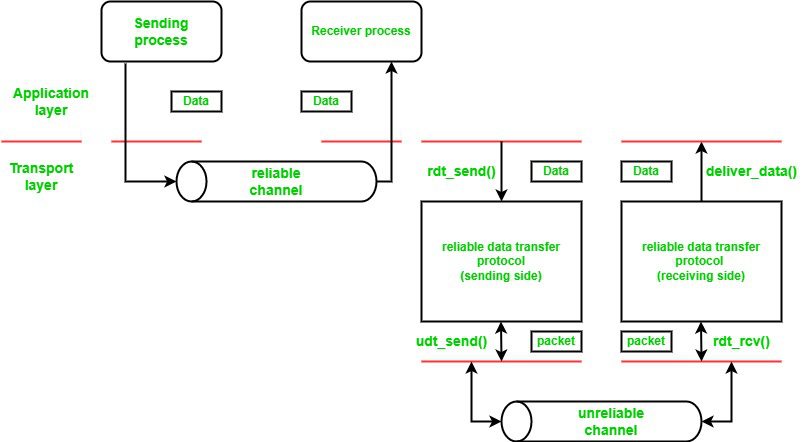
RUDP stands for Reliable user datagram protocol, whereas UDP is one of the core members of IP i.e internet protocol suite. UDP is used as a highly time-sensitive communicative protocol. R in RUDP stands for Reliable, it is a UDP-based data transfer but with higher Reliability. It provides reliability by using the sliding window protocol. RUDP is basically the solution to the UDP where data reliability along with confirmation is needed.

As UDP provides unreliable data transfer protocol which is unreliable services of things and the worst it sometimes goes missing without notice. Reliable UDP uses both positive and negative feedback to provide data reliability which provides reliable data transfer.

In UDP sender’s simply send a message without a piece of prior information about the receiver’s availability which results in a faster rate but it can result in the loss of some packages, also the receiver can receive duplicate packets and UDP also does not provide information that the package has been received or not. RUDP used a sliding window protocol that delivers or transfers the datagram with reliability.

**RUDP Architecture**

The sending process and receiving process of the architecture both stand in the application layer. The sender server sends the data packets to the receiving channel using RUDP protocol; a window size is maintained by both the sender and the receiver, the window consists of some predefined value tending to maximum avoiding the communication errors taking all the edge cases where the packets can be dropped.



*RUDP Protocol Architecture*

**Implementation of RUDP protocol:**

Use synchronized shared buffers using counting semaphores so that only one thread access the buffer at a time to prevent the deadlock situation from occurring.

Let’s keep two-variable called as the base and next to keep the track of the window functioning. If the sender sent’s the packet, the variable next is incremented to 1, this is the way of calculating the number of packets in the buffer.

The timeouts are handled using the timers which are scheduled immediately after sending the packet over the channel, simulate the packet loss rate and random network delay in the protocol.

packets are kept in a queue and assigned a value that values the system’s current time plus the network delay. When the current time of the system is associated with the assigned value the packet is freed from the queue and sent over the network. While the packet is sent over the network it gives an (ACK) acknowledgment. When the ack matches with the sequence number senders send another packet assigning it with the consecutive number.

**Classes in RUDP Protocol:**

Following are the classes that are used to implement the RUDP protocol:

* **RUDP:** This contains the send() and receives () functions called by the client and server.
* **Buffer\_RUDP:**contains the implementation shared buffer using counting semaphores.
* **Receiver\_Thread**: Implements a threaded architecture that simultaneously waits for the packets in the sockets while sending and receiving the data and waits to process the incoming data.
* **Segment\_RUDP:** defines the structure of RUDP packets.
* **Timeout\_Handler:** to handle the timeouts
* **Support\_RUDP:** provides functions like send\_UDP();
* **Client:**It sends the data.
* **Server:** It receives the data.

**Sliding Window Protocol:**

Sliding window protocol also known as RUDP’s SWP on which RUDP is based, is used where the reliability comes into play and it is very much essential or for security reasons where the knowledge of data transferred along with confirmation is required. It works in segments which means it divides the provided data into segments then packets or data packets are prepared and sent over the network, resulting in when the receiver receives the package it checks the order of the packets and discards duplicates and after that, is sent it sends the acknowledgment to the sender for the correctly received packets.

**There are 3 Sliding window Protocols:**

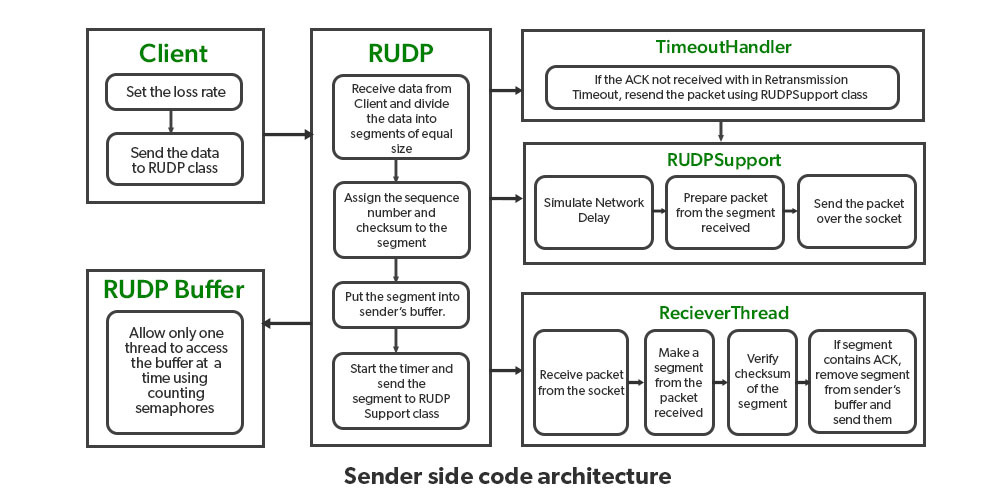
1. One Bit Sliding Window Protocol
2. Go back to N protocol
3. Selective Repeat Protocol

**Piggy-backing Technique:** Sliding window Protocol uses the piggy-backing technique, the technique is based on the acknowledgment that is received but, how is received? So, the answer is it provides the sliding window protocol to attach the acknowledgment in the next frame so that when a receiver receives the data it maintains a set of sequence numbers which is corresponding to the frames that are also can be called acknowledgment frames within a window portion of same or different sizes.

**The mathematics of frame calculation:** When a new packet is sent by the sender to the receiver it is given the highest packet sequence mark and the window’s upper edge is incremented with one for the acknowledged frames and vice versa for the unacknowledged frames. That calculates the mark of unacknowledged frames as well as acknowledged frames and when this number comes to zero that means all the packets have been delivered successfully and it provides feedback to the sender. It is used in data-intensive applications that require a lot of work in a very less amount of time over high-speed networks, where reliability plays a very important role.

**Server-Side Code Architecture:**

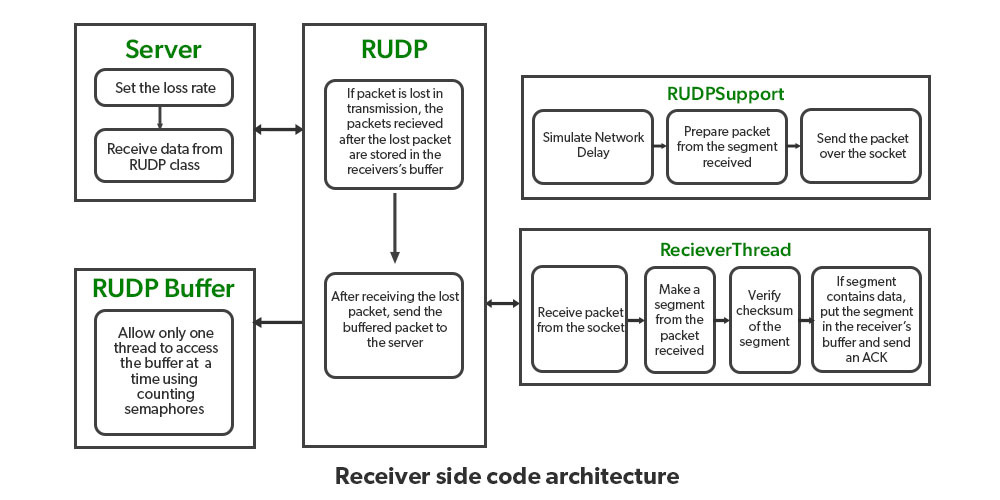
* Client Class sets the rate of loss and sends the data to the RUDP class, where window sizes of both the sender and receiver are set in the class.
* The RUDP class as per the working divides the data into segments of equal sizes and hence assigns the frames i.e. sequence number and checksum to each segment separately then put it in the sender’s window if it has any available slot.
* Support\_RUDP comes into play now, when the data packets in segments are sent by the sender this class collects and prepares the packets from the segments and sends them over the unreliable channel. Once it is sent the timer goes on immediately.
* Thread\_receiver class is the important class for the RUDP structure it provides a thread along with the thread of the data it parallelly sends a thread with the data and waits on the socket for the incoming data, when it receives the packet from the socket it makes a segment and verifies the checksum, if the checksum clarifies it is the sent data then it checks for the acknowledgment if the acknowledgment received then it removes the packet from sender’s window and sent it over the socket; if a timeout happens and a packet is lost it resends the packet and restart the timer immediately, If the RUDP class receives the packet before the timeout then it cancels the timer.



*Server-Side Code Architecture:*

**Receiver Side Code Architecture:**

* Thread\_receiver Class that implemented a thread parallel runs with the thread of the sent data and one of the instances waits at the socket to continue to process the data that is coming. Once the class receives the packet is verified with the segment and made a segment from it and matches with the checksum. If the segment contains the required data then it sends an acknowledgment and puts it in the Buffer\_receiver class and if the packet is not there then it stores all following the lost into the Buffer\_receiver.
* Once the lost packet is received it processes all and then the RUDP class delivers all the packets reordering correctly once it was buffered to the server.
* Once the confirmation is provided that all the data have been received then the connection is closed.



***Receiver Side Code Architecture:***

PseudoCode for RUDP Class:

Class RUDP{

//setting up the window sizes

set Receiver Window size

set Sender Window size

start receiver thread that receives the aka(Acknowledgment) and data for both the receiver and the sender

//sender calls

sent\_data(byte[] data\_gram, int size)

{

//process flow => data -> segments -> store into sender buffer -> send -> start timer

//breaking into segments and sending along with frames

Divide into the segments.

put every segment into the sender's buffer

segment sending using sent\_udp() of support\_RUDP class

timeout scheduling for the data that is divided into segments using frames

}

//receiver calls

receive\_data(byte[] buffer, int size)

{

//receiving of the segments of the data packets

segment receiving once at a time including the frames

}

//call by both the sender and receiver

close()

{

//creation of flag for the indication

creation of a flag segment to indicate the data transfer status

//verification of the data

//if data receiving completed -> close

once the complete data is received close the segment

}

}

//RUDP class ends here

Pseudo Code of Thread\_receiver class:

Class Thread\_receiver

{

while(true)

{

//waiting for the packet -> received

Receive the packet from socket

//numberize or make checksum of the data

individualize a segment from the packet that is received

//verification of the data

checksum verification that is sent along the frames

if segment contains acknowledgment

process -> remove segments from the sender's buffer

if segment contains data

put data->receiver's buffer

send acknowledgment

}

}

//end of Thread\_receiver Class

**Pseudo Code for Support\_RUDP class:**

Class Support\_RUDP

{

//simulate over the conditions that can cause the segments not received completely

random network delay

network loss

any other potential that can effect the packet

//process

packets from the segments received processing

//sent

sent the data packet over the socket

}

//end of Support\_RUDP class

**Pseudo Code for Client-Server Class**

//Sender

Class Client{

RUDP rudp = new RUDP(Host\_name, Port, Local);

//send the data

rudp.send();

//close

rudp.close();

}

//client class end

//Receiver Class start

Class Server{

RUDP rudp = new RUDP(Host\_name, Port, Local);

//receive the data

rudp.receive();

//close

rudp.close();

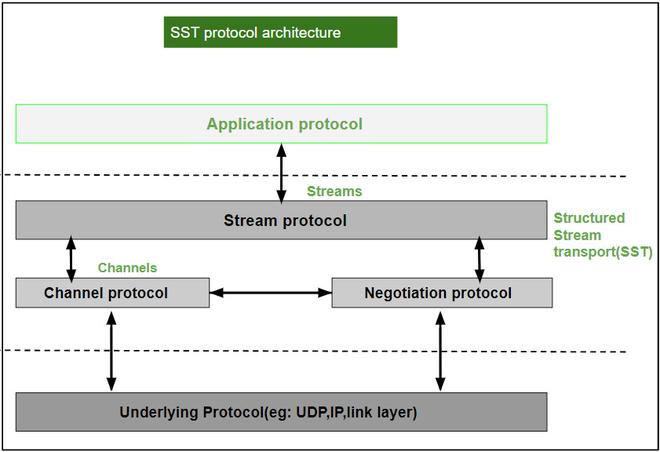
}

1. Structured Steam Transport (SST)

Structured stream transport is an experimental transport protocol that is similar to TCP (Transmission control protocol) that provides an organized, reliable byte-stream abstraction. SST enhances the traditional stream abstraction hierarchical hereditary structure, which allows applications to create lightweight child streams from any existing stream. Datagrams that support small transactions and streams are best suited for long-running conversations but neither of two support applications like HTTP(HyperText Transfer Protocol) which is a mixture of transaction sizes and applications or multiple transport instances(like FTP). TCP streams support 3-way hand-shaking delays on startup whereas Time-Wait streams in SST offer independent data transfer and flow control. All streams share one congestion control context.

**SST Protocol Architecture:**

* The channel protocol provides sequencing, connection security, and congestion control.
* The negotiation protocol provides mechanical of establishing channels between hosts, including symmetric key agreement for channel security and negotiating optional protocol extension.
* The registration protocol provides support for a simple, optional host registration and lookup service. Secure host identification and NAT (Network Access Transaction) traversal.
* The stream protocol is a convenient, high stream abstraction based on the above three protocols to implement the SST application.



*SST Protocol Architecture*

**Features of SST:**

* Over a single SST session, multiple independent streams can run in parallel.
* Relative priority between streams to apply application-specific policies.
* It provides wire efficiency, including header overhead of SST, UDP encapsulation is only 4 bytes larger than TCP.
* It provides built-in communication security.
* It provides efficient support for the short use of reliable steams.
* Streams can be arbitrarily long-running and may preserve internal application-specific record marks.
* Hole punching support for transparent communication across most NATs and Firewalls.

1. Sequenced Packet Exchange (SPX)

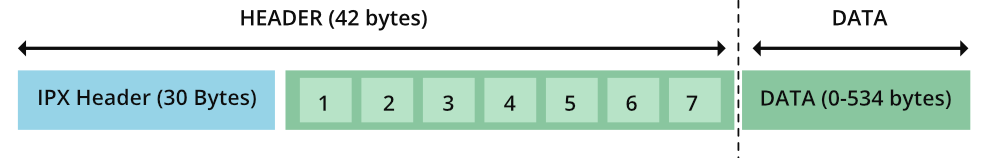
NetWare, developed by Novell is a NOS that expands to Network Operating systems. NOS is a special kind of operating system which helps the personal computers in a network (eg: computers connected in LAN) to share files and resources like printers. They make PCs connected over a network to act in a client-server architecture. The IPX/SPX is the protocol used in the NetWare OS.

The IPX protocol corresponds to the network layer and the SPX protocol on top of it corresponds to the transport layer in the OSI model. The IPX/SPX are similar to TCP/IP where IPX takes up the role of IP and SPX takes up the role of TCP.

**SPX Protocol**

The SPX protocol was developed by Novell for Local Area Networks. It expands to Sequenced Packet Exchange and it is connection-oriented. It enables the exchange of information between the clients and the server in a network with the help of IPX protocol. The IPX/SPX protocol is derived from the SPP (Sequenced Packet Protocol) which was defined in Xerox Network Systems. The SPP is a connectionless protocol while the SPX is connection-oriented.

**Structure of SPX packets**



*Format of SPX data packet*

The SPX data packet contains a 42-byte header and data may vary from 0 to 534 bytes. The minimum length is 42 bytes, ie., without any data. The first 30 bytes are taken from the IPX header and the rest of the 12 bytes are new seven fields (mentioned as 1…7 in the above diagram) belonging to SPX.

The fields considered from the **IPX header** are as follows:

* **Checksum -** The checksums are required to maintain the integrity of the data exchanged. The source and destinations derive the checksums in some methods. The source appends it to the header and sends it to the destination. On the other hand, the destination compares the checksum calculated by itself against the one in the packet’s header. If there is a mismatch then data is said to be corrupt. Usually, 16-bit checksums are used in networking protocols. IPX also uses **2 bytes** for checksum in the header. If the hex value of **FFFF**is set, then the checksum is not used.
* **Length -** This denotes the total length (header + data) of the packet exchanged. It consumes **2 bytes**.
* **Transport Control -** This is a **1-byte**field that is used to count the number of routers the packet has passed through. Starting from the value 0, each router that gets the packet adds ‘1’ to this field. When the packet has gone through 15 routers, the 16th router will discard the packet.
* **Packet type -** This is used to know what service does the packet render. It also takes only **1 byte**. Depending on this byte’s value, either one of the following can be the packet type :

0 – Unknown

1 – Routing Information protocol

2 – Echo packet

3 – Error packet

4 – Packet Exchange Protocol

5 – Sequenced Packet Protocol

17- NetWare Core Protocol

* **Source address -** The address of the source of the packet is mentioned in the IPX addressing structure. It takes **12 bytes**.
* **Destination address -** The address of the destination application to which the packet has to be delivered. It should follow the IPX addressing structure. For sending data to all the computers in the network FF:FF:FF:FF:FF: FF should be specified in the destination. It takes **12 bytes**.
* **IPX Addressing Structure -** The IPX addressing structure contains three fields:

1. Network
2. Node
3. Socket
4. The **network**is used to specify a network’s number in the internetwork communication. Internetwork communication allows data exchange between the nodes present in different cable connections. If the network number is mentioned as 00:00:00:00, then communication is happening within the same network. It takes **32-bits**to specify the number of a network.
5. The **node**is used to uniquely identify the personal computer in that network.
6. The **socket**is used to identify the appropriate application in the destination side to which the packet has to be delivered.

Thus, **2(checksum) + 2(Length) + 1(Transport control) + 1(Packet type) + 12(Source addr) + 12(Destination Addr) = 30 bytes of IPX header. The IPX header along with seven following fields make up the header of a SPX packet.**

**The seven new fields are:**

1. Connection control
2. Datastream type
3. Source connection ID
4. Destination connection ID
5. Sequence number
6. Acknowledge number
7. Allocation number
8. **Connection control -** It is a **1-byte**field, which controls the bidirectional data exchange. There are four flags that will be set depending on whether the packet is for a system or an individual application. The flags are as follows:

END\_OF\_MESSAGE

ATTENTION

ACKNOWLEDGEMENT\_REQUIRED

SYSTEM PACKET

1. **Datastream type -** It is a **1-byte**field that denotes the type of data in the packet. The hex values **0xFE and 0xFF**are reserved for specific purposes. The value 0xFE is used by the client to indicate that this is the last message and the packet delivered is known as the end-of-connection packet. If the value is set to 0xFF, then it is a system packet that acknowledges the termination of connection and this is referred to as an end-of-connection-acknowledgment packet.

**3 & 4) Source connection ID & Destination connection ID -** These refer to the identification number associated with the local source and remote destination. Each takes **2 bytes**, thus both together consume **4 bytes**.

1. **Sequence number -** This is a count managed by the SPX protocol to indicate the number of packets sent in a single direction. This takes **2 bytes** and the maximum value it can reach is 0xFFFF(65,535). After the maximum value, it resets to ‘0’.
2. **Acknowledge number -** This takes **2 bytes**and indicates the sequence number expected. Here also the maximum value is 0xFFFF (65,535). The duplicate packets can be determined using acknowledge number. If the **new packet’s sequence number < acknowledge number**, then it is a duplicate packet and has to be discarded.
3. **Allocation number -**The packets will be sent till the sequence number of the local source is equal to the allocation number of the remote destination. To control the flow of data and to know the remaining buffers that are listening to SPX packets, the (**allocation number of destination – acknowledge number**) formula can be used. It consumes **2 bytes.**

**Difference between TCP/IP and IPX/SPX**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **TCP/IP** | **IPX/SPX** |
| **1** | This is the basis of the Internet. | Mainly used and optimized for LAN connections and not suitable for the internet. |
| **2** | It is a connection-less protocol | It is a connection-oriented protocol. |
| **3** | The addressing logic is a bit complex as it requires both MAC, IP address, and masking techniques for routing the data. | The addressing logic is simpler as only the MAC address is used and is broken down into ethernet card and node addresses. |
| **4** | The network number has to be derived from the IP address. The subnet mask will be used for derivation. | Network numbers are separate and do not depend on the local address of the nodes. |
| **5** | Speed is lesser compared to IPX/SPX stack. | Faster stack compared to TCP/IP. |

Elements of Transport Protocols

To establish a reliable service between two machines on a network, transport protocols are implemented, which somehow resembles the data link protocols implemented at layer 2. The major difference lies in the fact that the data link layer uses a physical channel between two routers while the transport layer uses a subnet.

Following are the issues for implementing transport protocols−

**Types of Service**

The **transport layer** also determines the type of service provided to the users from the **session layer**. An error-free point-to-point communication to deliver messages in the order in which they were transmitted is one of the key functions of the transport layer.

* **Error Control - Error detection** and error recovery are an integral part of reliable service, and therefore they are necessary to perform error control mechanisms on an end-to-end basis. To control errors from lost or duplicate segments, the transport layer enables unique segment sequence numbers to the different packets of the message, creating virtual circuits, allowing only one virtual circuit per session.
* **Flow Control -** The underlying rule of flow control is to maintain a synergy between a fast process and a slow process. The transport layer enables a fast process to keep pace with a slow one. Acknowledgements are sent back to manage end-to-end flow control. Go back N algorithms are used to request retransmission of packets starting with packet number N. Selective Repeat is used to request specific packets to be retransmitted.

**Connection Establishment/Release**

The transport layer creates and releases the connection across the network. This includes a naming mechanism so that a process on one machine can indicate with whom it wishes to communicate. The transport layer enables us to establish and delete connections across the network to multiplex several message streams onto one communication channel.

**Multiplexing/De multiplexing**

The transport layer establishes a separate network connection for each transport connection required by the session layer. To improve throughput, the transport layer establishes multiple network connections. When the issue of throughput is not important, it multiplexes several transport connections onto the same network connection, thus reducing the cost of establishing and maintaining the network connections.

When several connections are multiplexed, they call for demultiplexing at the receiving end. In the case of the transport layer, the communication takes place only between two processes and not between two machines. Hence, communication at the transport layer is also known as peer-to-peer or process-to-process communication.

**Fragmentation and re-assembly**

When the transport layer receives a large message from the session layer, it breaks the message into smaller units depending upon the requirement. This process is called fragmentation. Thereafter, it is passed to the network layer. Conversely, when the transport layer acts as the receiving process, it reorders the pieces of a message before reassembling them into a message.

**Addressing**

Transport Layer deals with addressing or labelling a frame. It also differentiates between a connection and a transaction. Connection identifiers are ports or sockets that label each frame, so the receiving device knows which process it has been sent from. This helps in keeping track of multiple-message conversations. Ports or sockets address multiple conservations in the same location.

Congestion Control

A state occurring in network layer when the message traffic is so heavy that it slows down network response time.

**Effects** of Congestion

* As delay increases, performance decreases.
* If delay increases, retransmission occurs, making situation worse.

**Congestion control algorithms**

* Congestion Control is a mechanism that controls the entry of data packets into the network, enabling a better use of a shared network infrastructure and avoiding congestive collapse.
* Congestive-Avoidance Algorithms (CAA) are implemented at the TCP layer as the mechanism to avoid congestive collapse in a network.
* There are two congestion control algorithm which are as follows:

**Leaky Bucket Algorithm**

* The leaky bucket algorithm discovers its use in the context of network traffic shaping or rate-limiting.
* A leaky bucket execution and a token bucket execution are predominantly used for traffic shaping algorithms.
* This algorithm is used to control the rate at which traffic is sent to the network and shape the burst traffic to a steady traffic stream.
* The disadvantages compared with the leaky-bucket algorithm are the inefficient use of available network resources.
* The large area of network resources such as bandwidth is not being used effectively.

Let us consider an **example** to understand

Imagine a bucket with a small hole in the bottom.No matter at what rate water enters the bucket, the outflow is at constant rate.When the bucket is full with water additional water entering spills over the sides and is lost.

[](https://media.geeksforgeeks.org/wp-content/uploads/leaky.jpg)

Similarly, each network interface contains a leaky bucket and the following **steps** are involved in leaky bucket algorithm:

1. When host wants to send packet, packet is thrown into the bucket.
2. The bucket leaks at a constant rate, meaning the network interface transmits packets at a constant rate.
3. Bursty traffic is converted to a uniform traffic by the leaky bucket.
4. In practice the bucket is a finite queue that outputs at a finite rate.

**Token bucket Algorithm**

* The leaky bucket algorithm has a rigid output design at an average rate independent of the bursty traffic.
* In some applications, when large bursts arrive, the output is allowed to speed up. This calls for a more flexible algorithm, preferably one that never loses information. Therefore, a token bucket algorithm finds its uses in network traffic shaping or rate-limiting.
* It is a control algorithm that indicates when traffic should be sent. This order comes based on the display of tokens in the bucket.
* The bucket contains tokens. Each of the tokens defines a packet of predetermined size. Tokens in the bucket are deleted for the ability to share a packet.
* When tokens are shown, a flow to transmit traffic appears in the display of tokens.
* No token means no flow sends its packets. Hence, a flow transfers traffic up to its peak burst rate in good tokens in the bucket.

**Need** of token bucket Algorithm:-

The leaky bucket algorithm enforces output pattern at the average rate, no matter how bursty the traffic is. So in order to deal with the bursty traffic we need a flexible algorithm so that the data is not lost. One such algorithm is token bucket algorithm.

**Steps** of this algorithm can be described as follows:

1. In regular intervals tokens are thrown into the bucket. ƒ
2. The bucket has a maximum capacity. ƒ
3. If there is a ready packet, a token is removed from the bucket, and the packet is sent.
4. If there is no token in the bucket, the packet cannot be sent.

Let’s understand with an **example**,

 In figure (A) we see a bucket holding three tokens, with five packets waiting to be transmitted. For a packet to be transmitted, it must capture and destroy one token. In figure (B) We see that three of the five packets have gotten through, but the other two are stuck waiting for more tokens to be generated.

**Ways in which token bucket is superior to leaky bucket:** The leaky bucket algorithm controls the rate at which the packets are introduced in the network, but it is very conservative in nature. Some flexibility is introduced in the token bucket algorithm. In the token bucket, algorithm tokens are generated at each tick (up to a certain limit). For an incoming packet to be transmitted, it must capture a token and the transmission takes place at the same rate. Hence some of the busty packets are transmitted at the same rate if tokens are available and thus introduces some amount of flexibility in the system.

**Formula:** M \* s = C + ρ \* s where S – is time taken M – Maximum output rate ρ – Token arrival rate C – Capacity of the token bucket in byte

Let’s understand with an example,

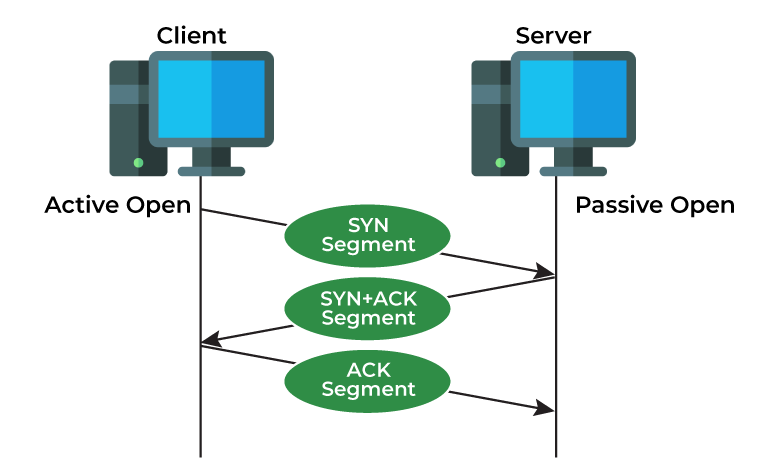
[](https://media.geeksforgeeks.org/wp-content/uploads/leakybuk.jpg)

TCP and UDP Protocols

Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) both are protocols of the Transport Layer. TCP is a connection-oriented protocol where as UDP is a part of the Internet Protocol suite, referred to as the UDP/IP suite. Unlike TCP, it is an unreliable and connectionless protocol.

**Transmission Control Protocol (TCP)**

TCP (Transmission Control Protocol) is one of the main protocols of the Internet protocol suite. It lies between the Application and Network Layers which are used in providing reliable delivery services. It is a connection-oriented protocol for communications that helps in the exchange of messages between different devices over a network. The Internet Protocol (IP), which establishes the technique for sending data packets between computers, works with TCP.



**Features of TCP**

* TCP keeps track of the segments being transmitted or received by assigning numbers to every single one of them.
* Flow control limits the rate at which a sender transfers data. This is done to ensure reliable delivery.
* TCP implements an error control mechanism for reliable data transfer.
* TCP takes into account the level of congestion in the network.

**Advantages of TCP**

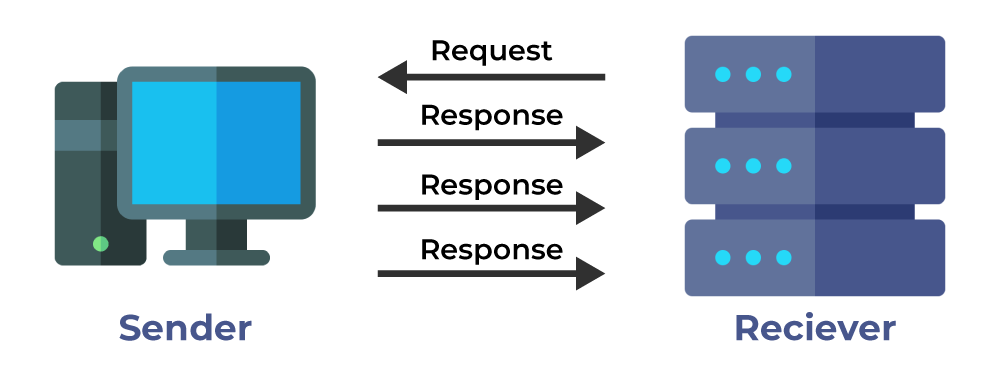
* It is reliable for maintaining a connection between Sender and Receiver.
* It is responsible for sending data in a particular sequence.
* Its operations are not dependent on OS.
* It allows and supports many routing protocols.
* It can reduce the speed of data based on the speed of the receiver.

**Disadvantages of TCP**

* It is slower than UDP and it takes more bandwidth.
* Slower upon starting of transfer of a file.
* Not suitable for LAN and PAN Networks.
* It does not have a multicast or broadcast category.
* It does not load the whole page if a single data of the page is missing.

**User Datagram Protocol (UDP)**

User Datagram Protocol (UDP) is a Transport Layer protocol. UDP is a part of the Internet Protocol suite, referred to as the UDP/IP suite. Unlike TCP, it is an unreliable and connectionless protocol. So, there is no need to establish a connection before data transfer. The UDP helps to establish low-latency and loss-tolerating connections establish over the network. The UDP enables process-to-process communication.



**Features of UDP**

* Used for simple request-response communication when the size of data is less and hence there is lesser concern about flow and error control.
* It is a suitable protocol for multicasting as UDP supports packet switching.
* UDP is used for some routing update protocols like [RIP(Routing Information Protocol)](https://www.geeksforgeeks.org/routing-information-protocol-rip/).
* Normally used for real-time applications which can not tolerate uneven delays between sections of a received message.

**Advantages of UDP**

* It does not require any connection for sending or receiving data.
* Broadcast and Multicast are available in UDP.
* UDP can operate on a large range of networks.
* UDP has live and real-time data.
* UDP can deliver data if all the components of the data are not complete.

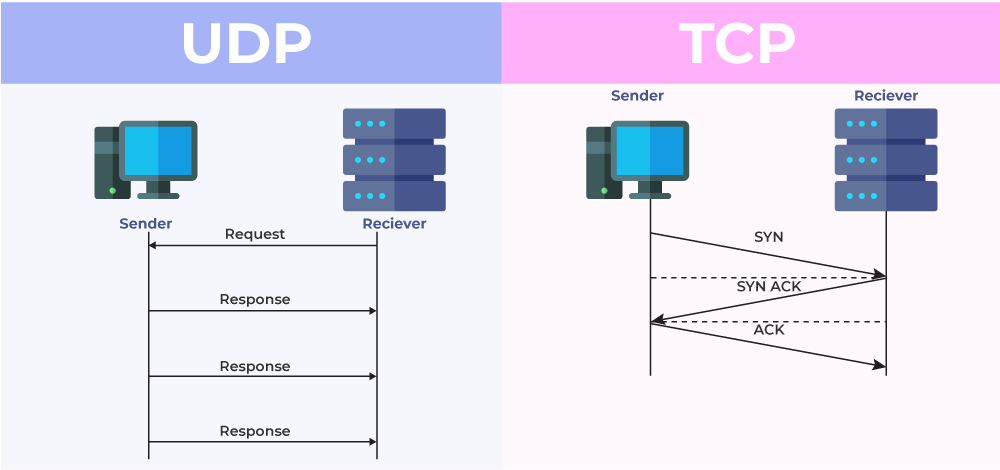
**Disadvantages of UDP**

* We can not have any way to acknowledge the successful transfer of data.
* UDP cannot have the mechanism to track the sequence of data.
* UDP is connectionless, and due to this, it is unreliable to transfer data.
* In case of a Collision, UDP packets are dropped by Routers in comparison to TCP.
* UDP can drop packets in case of detection of errors.

**Which Protocol is Better: TCP or UDP?**

The answer to this question is difficult because it totally depends on what work we are doing and what type of data is being delivered. UDP is better in the case of online gaming as it allows us to work lag-free. TCP is better if we are transferring data like photos, videos, etc. because it ensures that data must be correct has to be sent.

In general, both TCP and UDP are useful in the context of the work assigned by us. Both have advantages upon the works we are performing, that’s why it is difficult to say, which one is better.



*Difference Between TCP and UDP*

* Where TCP is Used?
* Sending Emails
* Transferring Files
* Web Browsing
* Where UDP is Used?
* Gaming
* Video Streaming
* Online Video Chats

**Differences between TCP and UDP**

The main differences between TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are:

| **Basis** | **Transmission Control Protocol (TCP)** | **User Datagram Protocol (UDP)** |
| --- | --- | --- |
| Type of Service | [TCP](https://www.geeksforgeeks.org/what-is-transmission-control-protocol-tcp/) is a connection-oriented protocol. Connection  orientation means that the communicating devices should establish a connection before transmitting data and should close the connection after transmitting the data. | [UDP](https://www.geeksforgeeks.org/user-datagram-protocol-udp/)is the Datagram-oriented protocol. This is because  there is no overhead for opening a connection, maintaining a connection, or terminating a connection. UDP is efficient for broadcast and multicast types of network transmission. |
| Reliability | TCP is reliable as it guarantees the delivery of data to the destination router. | The delivery of data to the destination cannot be guaranteed in UDP. |
| Error checking mechanism | TCP provides extensive error-checking mechanisms.  It is because it provides flow control and acknowledgment of data. | UDP has only the basic error-checking mechanism using checksums. |
| Acknowledgment | An acknowledgment segment is present. | No acknowledgment segment. |
| Sequence | Sequencing of data is a feature of Transmission Control  Protocol (TCP). this means that packets arrive in order at the receiver. | There is no sequencing of data in UDP. If the order is required, it has to be managed by the application layer. |
| Speed | TCP is comparatively slower than UDP. | UDP is faster, simpler, and more efficient than TCP. |
| Retransmission | Retransmission of lost packets is possible in TCP, but not in UDP. | There is no retransmission of lost packets in the User Datagram Protocol (UDP). |
| Header Length | TCP has a (20-60) bytes variable length header. | UDP has an 8 bytes fixed-length header. |
| Weight | TCP is heavy-weight. | UDP is lightweight. |
| Handshaking Techniques | Uses handshakes such as SYN, ACK, SYN-ACK | It’s a connectionless protocol i.e. No handshake |
| Broadcasting | TCP doesn’t support Broadcasting. | UDP supports Broadcasting. |
| Protocols | TCP is used by [HTTP, HTTPs](https://www.geeksforgeeks.org/difference-between-http-and-https-2/),[FTP](https://www.geeksforgeeks.org/file-transfer-protocol-ftp/), [SMTP](https://www.geeksforgeeks.org/simple-mail-transfer-protocol-smtp/) and [Telnet](https://www.geeksforgeeks.org/introduction-to-telnet/). | UDP is used by [DNS](https://www.geeksforgeeks.org/details-on-dns/), [DHCP](https://www.geeksforgeeks.org/dynamic-host-configuration-protocol-dhcp/), TFTP, [SNMP](https://www.geeksforgeeks.org/simple-network-management-protocol-snmp/), [RIP](https://www.geeksforgeeks.org/routing-information-protocol-rip/), and [VoIP](https://www.geeksforgeeks.org/voice-over-internet-protocol-voip/). |
| Stream Type | The TCP connection is a byte stream. | UDP connection is a message stream. |
| Overhead | Low but higher than UDP. | Very low. |
| Applications | This protocol is primarily utilized in situations when a safe and trustworthy communication procedure is necessary, such as in email, on the web surfing, and in military services. | This protocol is used in situations where quick communication is necessary but where dependability is not a concern, such as VoIP, game streaming, video, and music streaming, etc. |

**Example:**Suppose there are two houses, H1 and H2, and a letter has to be sent from H1 to H2. But there is a river in between those two houses. Now how can we send the letter? 

**Solution 1:** Make a bridge over the river and then it can be delivered.

**Solution 2:** Get it delivered by a pigeon.

* Consider the first solution as **TCP**. A connection has to be made (bridge) to get the data (letter) delivered. The data is reliable because it will directly reach another end without loss of data or error.
* The second solution is **UDP**. No connection is required for sending the data. The process is fast as compared to TCP, where we need to set up a connection (bridge). But the data is not reliable: we don’t know whether the pigeon will go in the right direction will drop the letter on the way, or some issue is encountered mid-travel.

Quality of Service Model

**Quality-of-Service (QoS)** refers to traffic control mechanisms that seek to either differentiate performance based on application or network-operator requirements or provide predictable or guaranteed performance to applications, sessions, or traffic aggregates. Basic phenomenon for QoS means in terms of packet delay and losses of various kinds.

**Need for QoS –**

* Video and audio conferencing require bounded delay and loss rate.
* Video and audio streaming requires bounded packet loss rate, it may not be so sensitive to delay.
* Time-critical applications (real-time control) in which bounded delay is considered to be an important factor.
* Valuable applications should be provided better services than less valuable applications.

**QoS Specification –**

QoS requirements can be specified as:

1. Delay
2. Delay Variation(Jitter)
3. Throughput
4. Error Rate

There are two types of QoS Solutions:

1. **Stateless Solutions –** Routers maintain no fine-grained state about traffic, one positive factor of it is that it is scalable and robust. But it has weak services as there is no guarantee about the kind of delay or performance in a particular application which we have to encounter.
2. **Stateful Solutions –** Routers maintain a per-flow state as flow is very important in providing the Quality-of-Service i.e. providing powerful services such as guaranteed services and high resource utilization, providing protection, and is much less scalable and robust.

**Integrated Services(IntServ) –**

1. An architecture for providing QoS guarantees in IP networks for individual application sessions.
2. Relies on resource reservation, and routers need to maintain state information of allocated resources and respond to new call setup requests.
3. Network decides whether to admit or deny a new call setup request.

**IntServ QoS Components –**

1. Resource reservation: call setup signaling, traffic, QoS declaration, per-element admission control.
2. QoS-sensitive scheduling e.g WFQ queue discipline.
3. QoS-sensitive routing algorithm(QSPF)
4. QoS-sensitive packet discard strategy.

**RSVP-Internet Signaling**

It creates and maintains distributed reservation state, initiated by the receiver and scales for multicast, which needs to be refreshed otherwise reservation times out as it is in soft state. Latest paths were discovered through “PATH” messages (forward direction) and used by RESV messages (reserve direction).

**Call Admission**

Session must first declare it’s QoS requirement and characterize the traffic it will send through the network.

* **R-specification:** defines the QoS being requested, i.e. what kind of bound we want on the delay, what kind of packet loss is acceptable, etc.
* **T-specification:** defines the traffic characteristics like bustiness in the traffic.
* A signaling protocol is needed to carry the R-spec and T-spec to the routers where reservation is required.
* Routers will admit calls based on their R-spec, T-spec and based on the current resource allocated at the routers to other calls.

**Diff-Serv –**  
Differentiated Service is a stateful solution in which each flow doesn’t mean a different state. It provides reduced state services i.e. maintaining state only for larger granular flows rather than end-to-end flows tries to achieve the best of both worlds.

Intended to address the following difficulties with IntServ and RSVP:

* **Flexible Service Models:** IntServ has only two classes, want to provide more qualitative service classes: want to provide ‘relative’ service distinction.
* **Simpler signaling:** Many applications and users may only want to specify a more qualitative notion of service.

**Streaming Live Multimedia –**

* **Examples:** Internet radio talk show, Live sporting event.
* **Streaming:** playback buffer, playback buffer can lag tens of seconds after and still have timing constraint.
* **Interactivity:** fast forward is impossible, but rewind and pause is possible.

Best Effort Model

Network Performance Issues

Business networks are complex, and many things can go wrong that disrupt network performance. End users often complain about what appears to be poor application performance, and there can be many possible reasons for these hiccups. Here are nine of the most common network issues to troubleshoot.

1. **Slow network**

Users complain the network is too slow. There can be many reasons why a network that provided adequate performance in the past is now frustrating its users. A failing switch port or link could cause traffic to route around the failure and overload another link.

In other cases, the network could be part of a larger organizational network. As a result, a change in the larger network has resulted in more traffic through the internet connection point, slowing responses to cloud-resident applications.

Another network speed issue could emerge if employees decide to download high-definition videos while at work because downloading in the office is faster than using their home internet connection. A network monitoring tool helps solve any of these common network issues.

1. **Weak Wi-Fi signal**

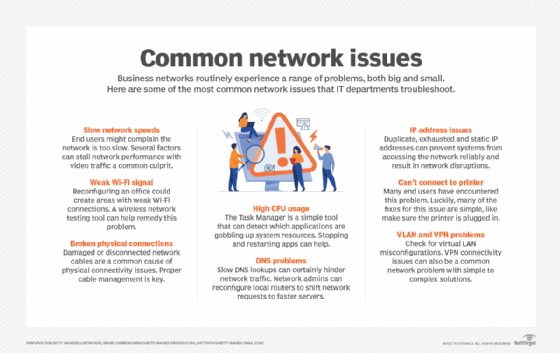
Wi-Fi signal strength may be adequate almost everywhere, but it could be weak or nonexistent in other areas. Rearranging an office area can result in a weak wireless connection, where signal strength had been adequate before the move. For example, a large metal object, like a file cabinet, can block the Wi-Fi signal.

Devices such as microwave ovens, cordless phones and Bluetooth can interfere with Wi-Fi signals, too. A Wi-Fi network test tool can help identify the source of the problem.

1. **Physical connectivity issues**

A network connection can suddenly break because of physical connectivity issues. A common problem is when a network cable becomes damaged or knocked loose. Cables might be added or removed from a switch, and one of the other cables might accidentally get disconnected.

Or a cable was damaged when it was pulled around a sharp edge while work was done on the heating or air conditioning pipes. It should be clear from the segment of the network affected which cable was damaged. But finding the problem along a cable stretching across the ceiling may be time-consuming.



1. **Excessive CPU usage**

Task Manager is the first thing to use to find which application is using a high proportion of system resources, such as CPU, memory or disk space. This basic troubleshooting step may not reveal a problem since some applications may be performing complex calculations, receiving high-speed video or interacting with large databases. A virus may also consume resources, so make sure antivirus software is up to date.

If an application has been running for a long time, it may slowly leak resources. The quickest way to improve performance is to stop and restart the application, although sometimes you may need to stop and restart the entire system. Updating device drivers may also improve performance.

Task Manager also shows applications you didn't know were running in the background. One example would be Windows including games upon system startup. Editing startup files can eliminate this problem.

1. **Slow DNS lookups**

The DNS matches the common name used to match server or service names with the internet address that routes a network request. For commonly used names, the matchup is probably already stored in the system's DNS cache, and the lookup is quick. For less commonly used names, the matchup may be stored in a more distant cache, such as the root server of the top-level name, such as .com, .org or a national root, such as .uk.

Each DNS server along the path checks its cache before making a request to the next server along the path. The next server then checks its cache, repeating the process. If lookup is slow, there may be a slow link along the path or a slow or overloaded server. To address this issue, your local network administrator can reconfigure local routers to shift requests to a faster chain of servers.

1. **Duplicate and static IP addresses**

On a network, no two systems can share the same internet address. If there are duplicate internet addresses, neither system can access the network reliably. The addresses for most network devices are assigned when Dynamic Host Configuration Protocol (DHCP) boots up the systems on the local network. DHCP maintains a pool of addresses assigned to the local network, assigning a different address from the pool to each system.

Workstations are not assigned permanent addresses but receive one for a limited time from DHCP. Systems re-request before the time runs out and usually receive the same address. If the system shuts down without re-requesting and the time runs out, it loses this address and may receive a different one upon startup.

The DHCP administrator may assign a static IP address to some network devices, such as printers or web servers, because external systems won't be updated if an address changes. One issue is users sometimes set up a private web server to support a hobby, allocating a static address without informing the network administrator. Both share a DHCP server in either an organization or home network. So, if the static address matches one assigned by DHCP, it disrupts the network.

Often, these private web servers are set up to upload and download licensed music or video and consume excessive network bandwidth.

1. **Exhausted IP addresses**

Internet addresses are in limited supply. Each service provider is given a supply based on the expected number necessary. Most familiar are the [IPv4 addresses](https://www.techtarget.com/whatis/definition/IPv4-address-class), which were originally thought to be adequate so every system could be allocated one. But, with the proliferation of cellphones and other devices, it's been necessary to move to [IPv6](https://www.techtarget.com/searchnetworking/definition/IPv6-Internet-Protocol-Version-6) with 128-bit addresses for some networks.

A widely used method to stretch the supply of addresses is Network Address Translation ([NAT](https://www.techtarget.com/searchnetworking/definition/Network-Address-Translation-NAT)), a feature often built into routers. Each is assigned a single internet address allocated from the worldwide set of addresses. Its internal DHCP server allocates private addresses to systems on connected local networks -- usually, an Ethernet or wireless network.

Private addresses generally start with either 10 or 192.168 on networks using 32-bit IPv4 addresses. These address ranges can be used many times, which helps to save addresses. The NAT server maps traffic to its global address to communicate with the internet. Responses are mapped back via the private addresses.

1. **Can't connect to printer**

When users can't connect to a printer, the first step is to check simple things like whether the printer is plugged in, turned on and has paper. Also, make sure the printer appears on Devices and Printers on Windows. If it does, click to check whether the file is queued.

Sometimes, you need to stop and restart the print spooler, the software that stores files until the printer is ready to print them. Also, check the printer vendor's website because some brands have a downloadable app that can diagnose and fix problems.

If the OS was just upgraded, scan for other people with similar problems, or check Microsoft.com to see if the company is aware of a problem. Shut off the printer, and turn it back on. Also, shut down your system, and turn it back on.

Finally, update printer drivers and your OS. In some cases, you may need to temporarily shut down your antivirus software. For a wireless printer, make sure it's connected to the signal.

1. **VLAN and VPN problems**

Check for virtual LAN (VLAN) mis-configuration issues. Review the configuration on each switch, carefully comparing configurations to ensure compatibility of switch configuration.

The most common VPN problem is a failure to connect. First, check to see if you're successfully logging in to the service, and make sure your account is up to date and you're entering your correct credentials. Next, check firewall settings. You need to open some ports. Check if that is the problem by temporarily shutting down your firewall. Finally, restart your system.

Try accessing the VPN from a different network, such as switching from Wi-Fi to Ethernet to the router. If there is still a problem, refer to the firewall documentation for other solutions, or contact the VPN vendor support.

In sum, networks are complex, and problems do occur. These are just some of the most common types of network problems. When other types of network issues occur, scan the web for help, or contact network service providers or device vendor support.