Transport Layer

Compiled by :- Er. Nagendra karn

Transport layer

- The data link layer is responsible for delivery of frames between two neighboring nodes over a link. This is called node-to-node delivery.
- The network layer is responsible for delivery of datagrams between two hosts. This is called host-to-host delivery.
- The transport layer is responsible for process-to-process delivery. The delivery of packet, part of a message, from one process to another.
- The basic function of transport layer is to accept data from above, split it up into smaller units if need be, pass these to the network layer, and ensure that the pieces all arrive correctly at the other end.
- Furthermore, all this must be done efficiently and in a way that isolates the upper layers from the inevitable changes in the hardware technology.

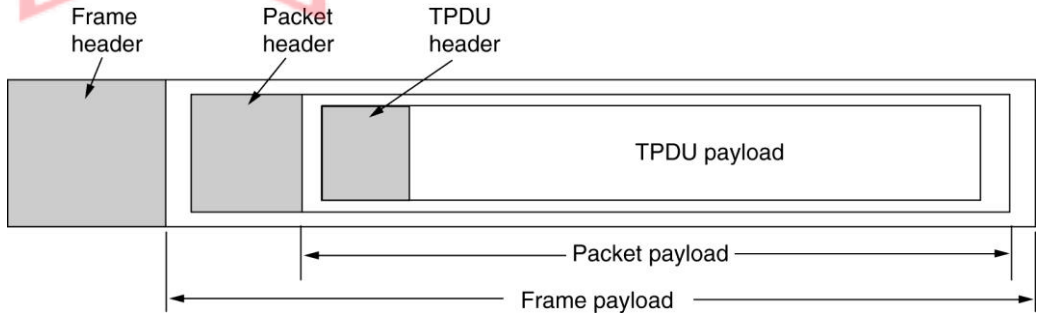
Introduction

- Fourth layer of OSI reference model
- Protocol Data Unit (PDU) is called **Segment**
- “port number” is added to identify different processes and applications running in the single host (ie. Only one IP).
- Why we need Transport Layer?
  - Allows different processes or applications to run simultaneously in same host
  - Users have no control over the Network Services. So, if something goes wrong, a user can do nothing.
  - The transport service is what users can add to improve the reliability of the service provided by the network.

Services provided

- Error Handling
- Flow Control
- Multiplexing
- Connection Set-up and Release
- Congestion Handling
- Segmentation and Reassembly
- Addressing

- **Data Encapsulation at different layers**

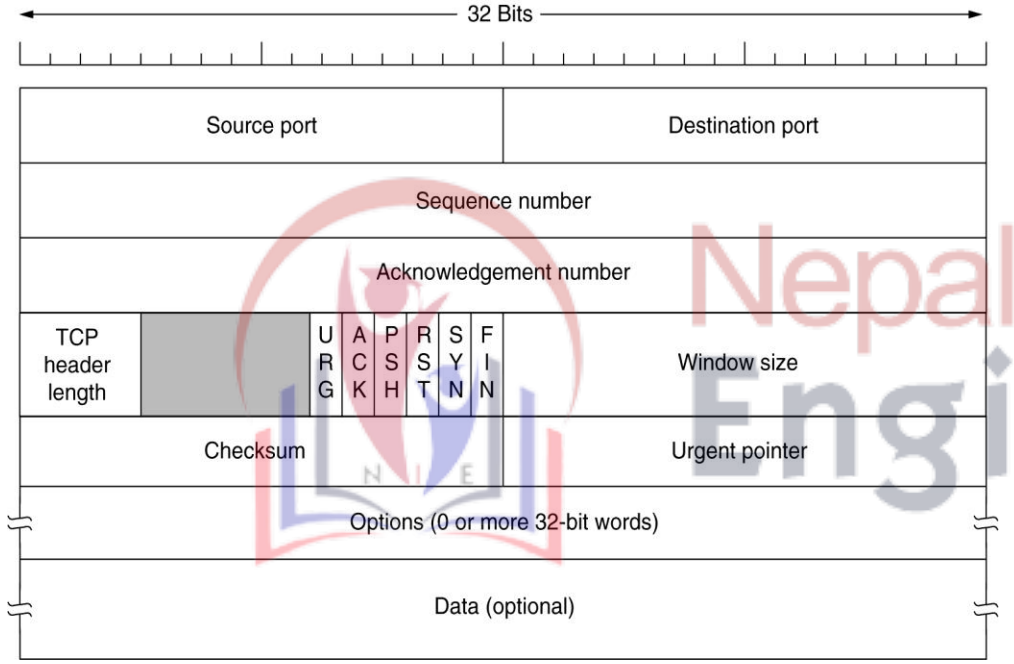


Transport Layer Protocols

- TCP(Transmission Control Protocol)
  - Reliable, connection oriented byte-stream protocol
  - In-sequence segment delivery
  - Flow control(with seq. and ACK with sliding window)
  - Error detection with checksum
  - Very reliable, usually used for file transfers.
- UDP (User Datagram Protocol)
  - Simple (asynchronous) multiplexing protocol
  - no flow control, not reliable
  - Connection less, no ack. and no seq. no.
  - Error detection is optional and no retransmission at all
  - Main Advantage is Speed
  - Usually used for real time traffics like live video streaming, video or voice chatting etc

Transport Layer Protocols

- **TCP(Transmission Control Protocol)**

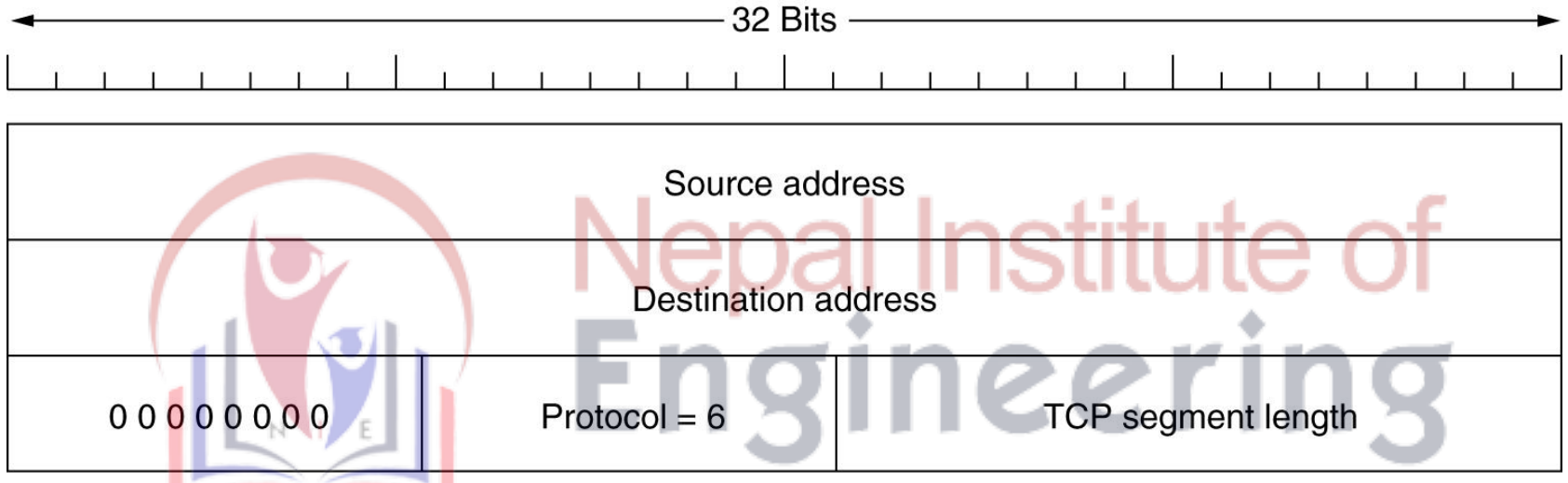


TCP Segment Format

- Seq. No and ACK are used for flow control
- TCP Header Length>=20 B
- URG:1=Urgent pointer in use
- ACK:1=Ack. No. is valid
- PSH: Push data (send to app. Layer, do not buffer)
- RST: Reset Connection
- SYN: used to estb. connection
- FIN: used to release connect^{n}
- Window Size: used for sliding window protocol(works on bytes)
- Checksum: computed with TCP header, data and pseudo header
- Urgent pointer: byte offset of urg. data
- Options: Extra facilities eg. Specify maximum TCP Load

Transport Layer Protocols

- TCP(Transmission Control Protocol)

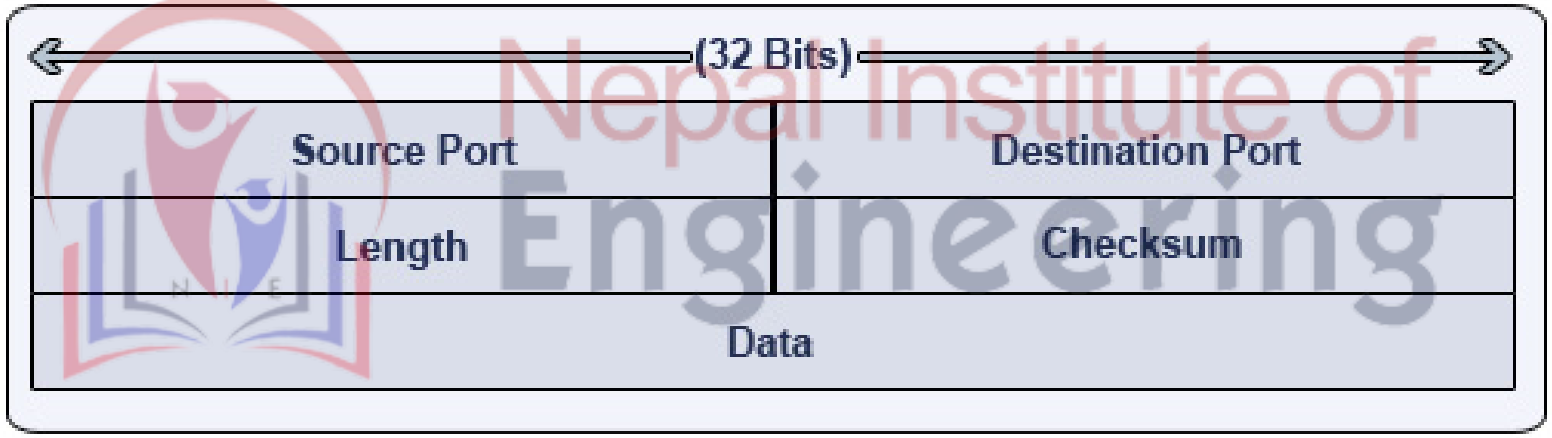


- IP Addresses are also used to compute TCP Checksum for extra reliability.

The pseudoheader included in the TCP checksum.

Transport Layer Protocols

- UDP (User Datagram Protocol)



UDP Segment Format

- Length : Header + Data length (min. 8 when no data)
- Checksum: computed with TCP header, data and pseudo header

Transport Layer Protocols

TCP vs UDP

TCP	UDP
Reliable	Unreliable
Connection-oriented	Connectionless
Segment retransmission and flow control through windowing	No windowing or retransmission
Segment sequencing	No sequencing
Acknowledge segments	No acknowledgement

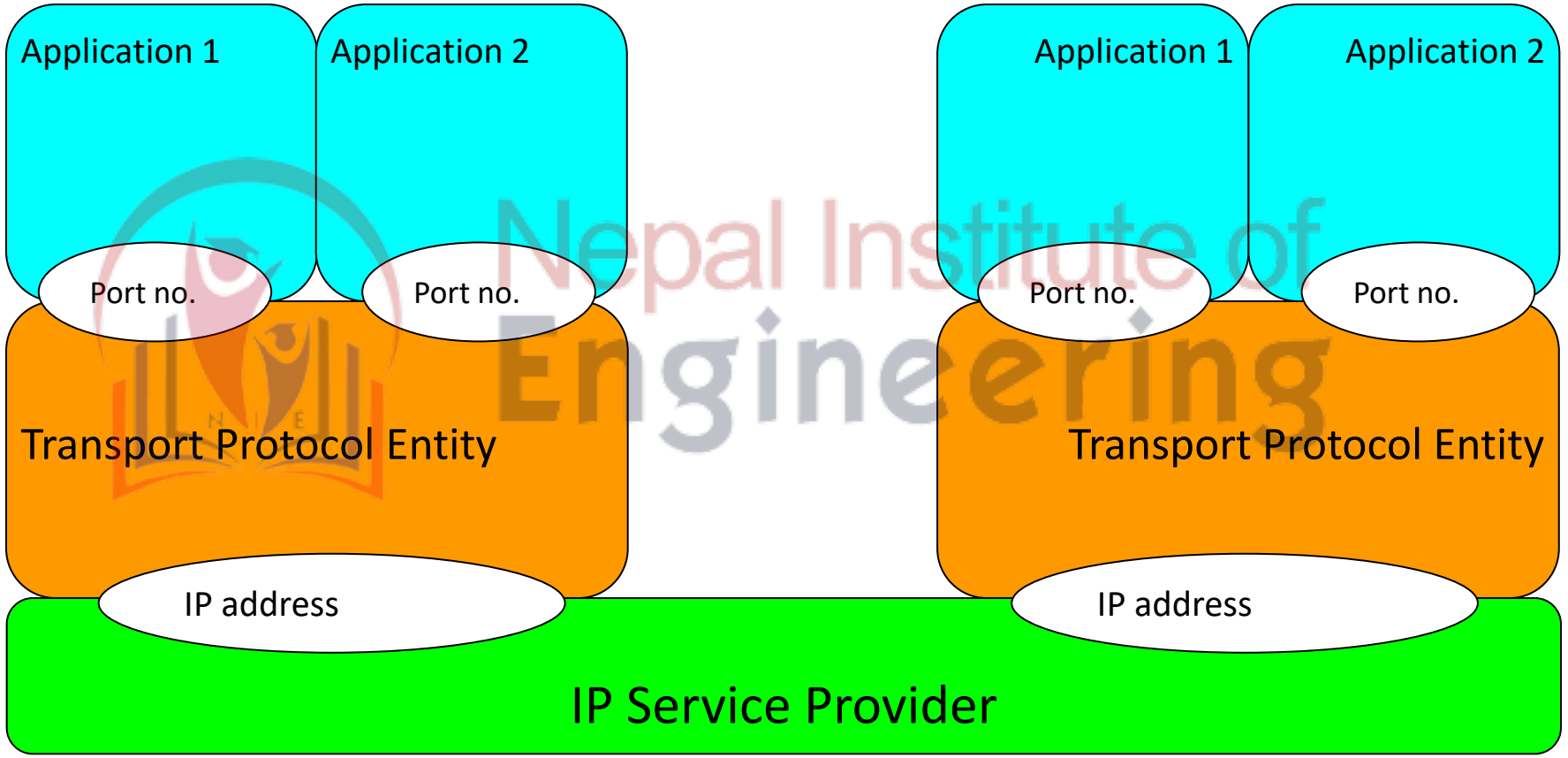
Port and Socket

- Port
  - An application-specific or process-specific software construct to identify different processes of same host
  - Analogous to telephone extension inside an organization
  - When you open *yahoo.com* and *hotmail.com* on same PC at same time, they will have different port numbers.

Port and Socket

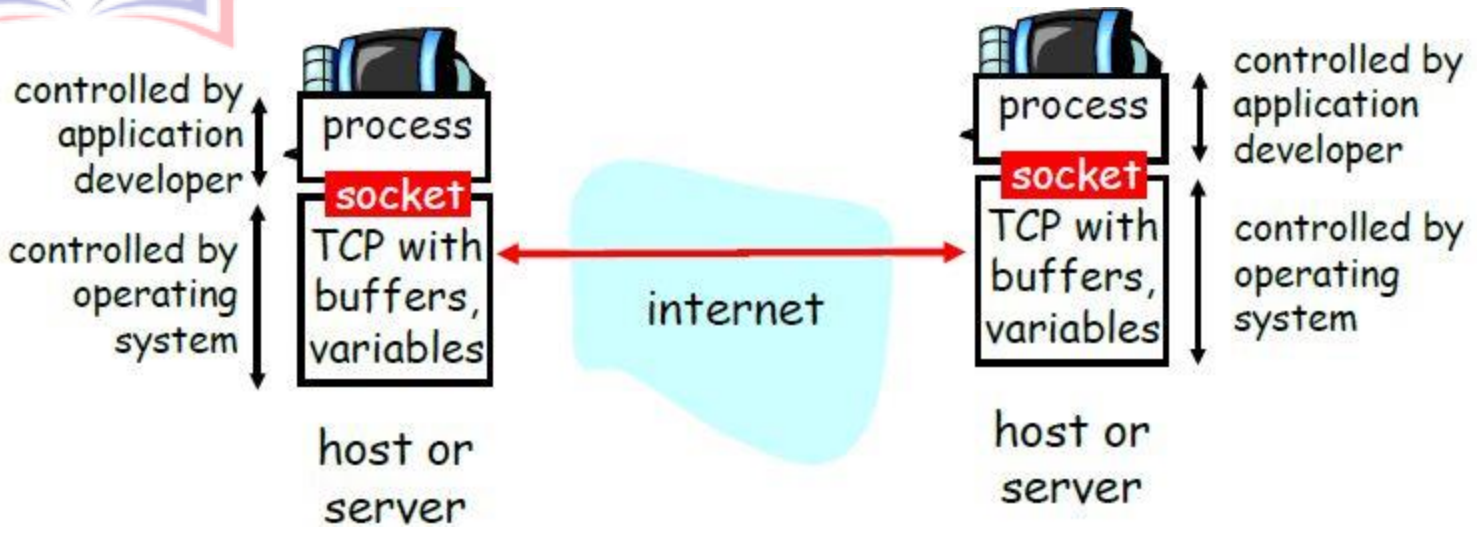
- Types of Port Number
  - Well-Known Ports
    - Port numbers 0 to 1023
    - Used for most common network applications
    - Eg. HTTP - 80, FTP data - 20, FTP control - 21, DHCP Client - 68, DHCP Server - 67, DNS - 53, SMTP – 25, POP3 – 110 etc
  - Registered Ports
    - Port Numbers 1024 to 49151
    - Assigned by IANA for specific services eg. VLC – 1234, Skype - 23399
    - In most systems, these ports can also be used by ordinary users
  - Dynamic/Private Ports
    - Port Numbers 49152 to 65535
    - Cannot be registered with IANA
    - Used for temporary purposes

Port and Socket



Port and Socket

- Socket
  - Endpoint of an inter-process communication flow across a computer network.
  - Socket address is a combination of IP and Port no. according to which it delivers data to appropriate application or process
  - Integrated into the operating system
  - Explicitly created, used, released by applications

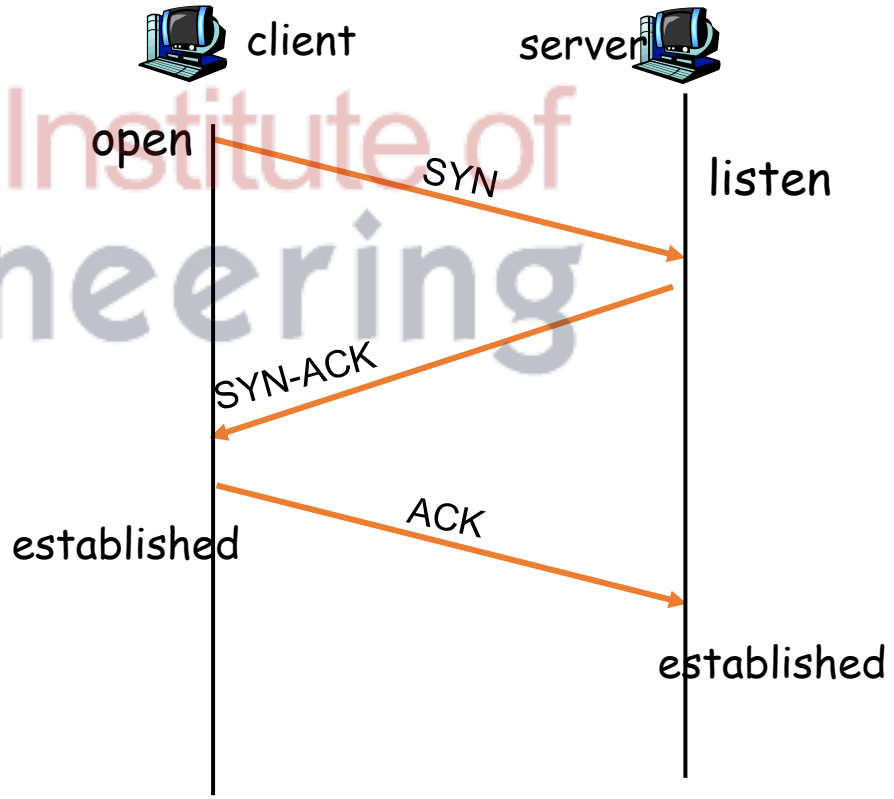


Transport Control Protocol (TCP)

- **TCP Protocol Functions**
  - Multiplexing
  - Error handling
  - Flow control
  - Congestion Handling
  - Connection set-up and release
- **TCP Transport service**
  - Connection oriented (full duplex point-to-point connection between processes).
  - Reliable
  - In-sequence segment delivery.
- In TCP, connection-oriented transmission **requires three phases** : connection establishment, data transfer, and connection termination.
- TCP identifies connections on the basis of endpoints :
  - IP address + Port number
  - Often written as : IP-address : port-number, for instance : 130.89.17.3:80

Connection Establishment

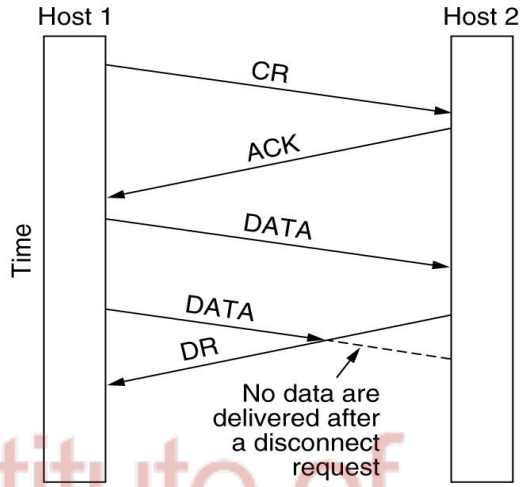
- 3-way handshake:
  - Step 1:
    Client end system sends TCP SYN control segment to server
  - Step 2:
    Server end system receives SYN, replies with SYN-ACK
    - Allocates buffers
    - ACKs received SYN
  - Step 3:
    Client receives SYN-ACK
    - connection is now set up
    - client starts the “real work”



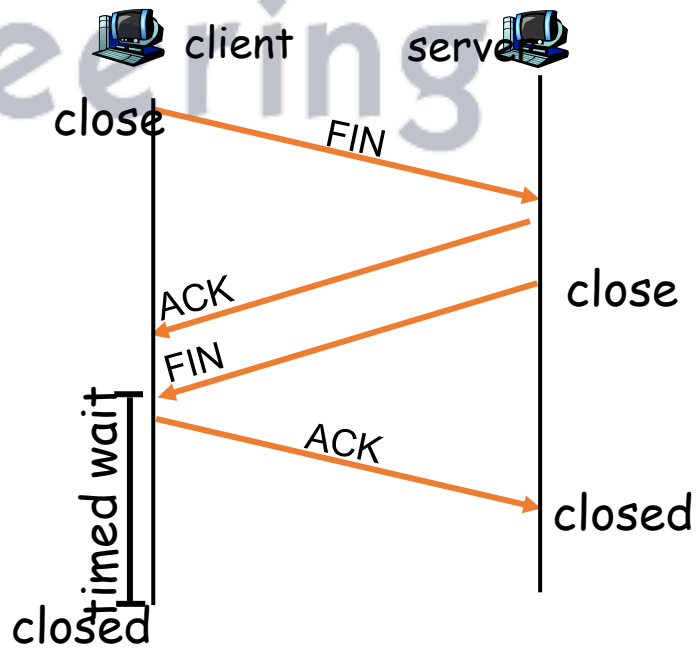
Connection Release

Release can be asymmetric(one sided, where data may loss) or symmetric (both side agreed)
Steps for symmetric Release:

- Step 1: Client end system sends TCP FIN control segment to server
- Step 2: Server receives FIN, replies with ACK. Closes connection, sends FIN.
- Step 3: Client receives FIN, replies with ACK.
  - Enters “timed wait” - will respond with ACK to received FINs
- Step 4: server, receives ACK. Connection closed.



Abrupt disconnection with loss of data.

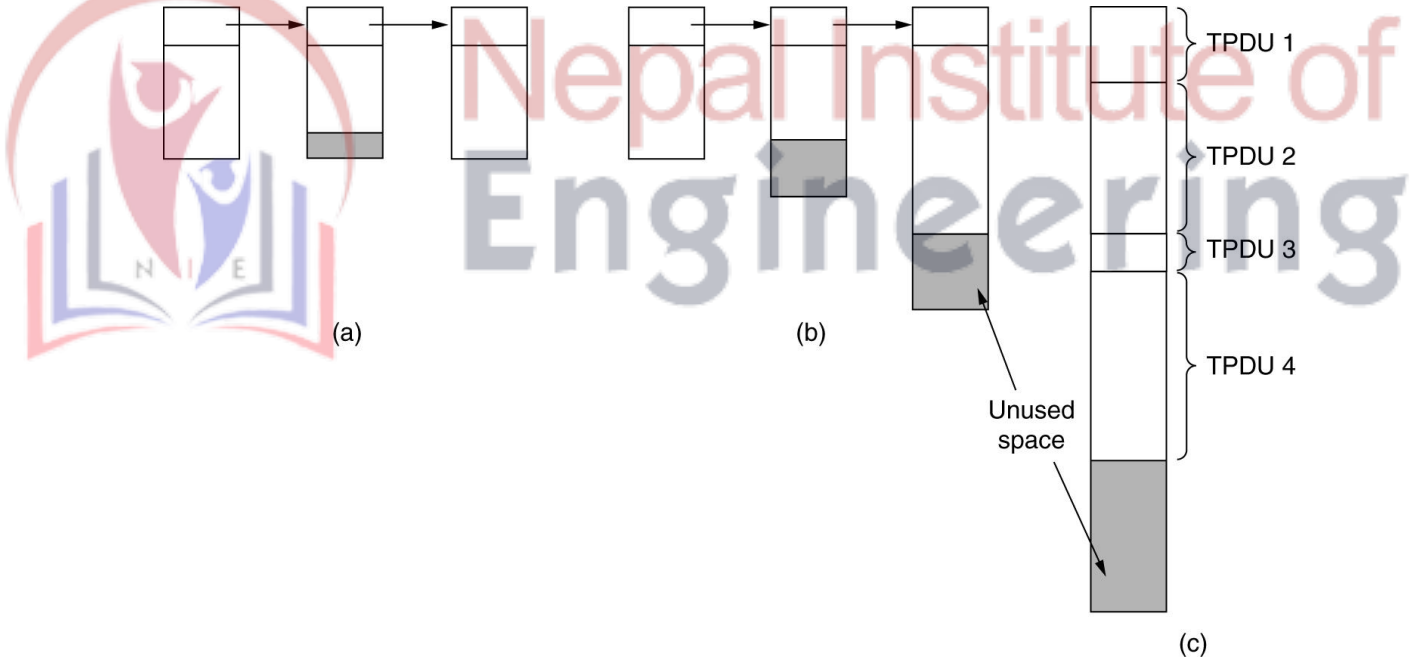


Symmetric Release

Flow Control and Buffering

- Flow Control, similar to data link layer (sliding window), except that transport layer has to deal with large number of connections
- Flow control at this layer is performed **end-to-end** rather than across a single link.
- Allocating dedicated buffers per connection at sender and receiver may not be practical
- Different approaches for buffer organization
  - Chained fixed-size buffers
  - Chained variable-sized buffers
  - One large circular buffer per connection
- Type of traffic carried by a connection also influences buffering strategy
- As connections come and go, and traffic types changes, need to able to adjust buffer allocations dynamically
  - Using variable-sized sliding windows
  - Decouple window management from acknowledgements

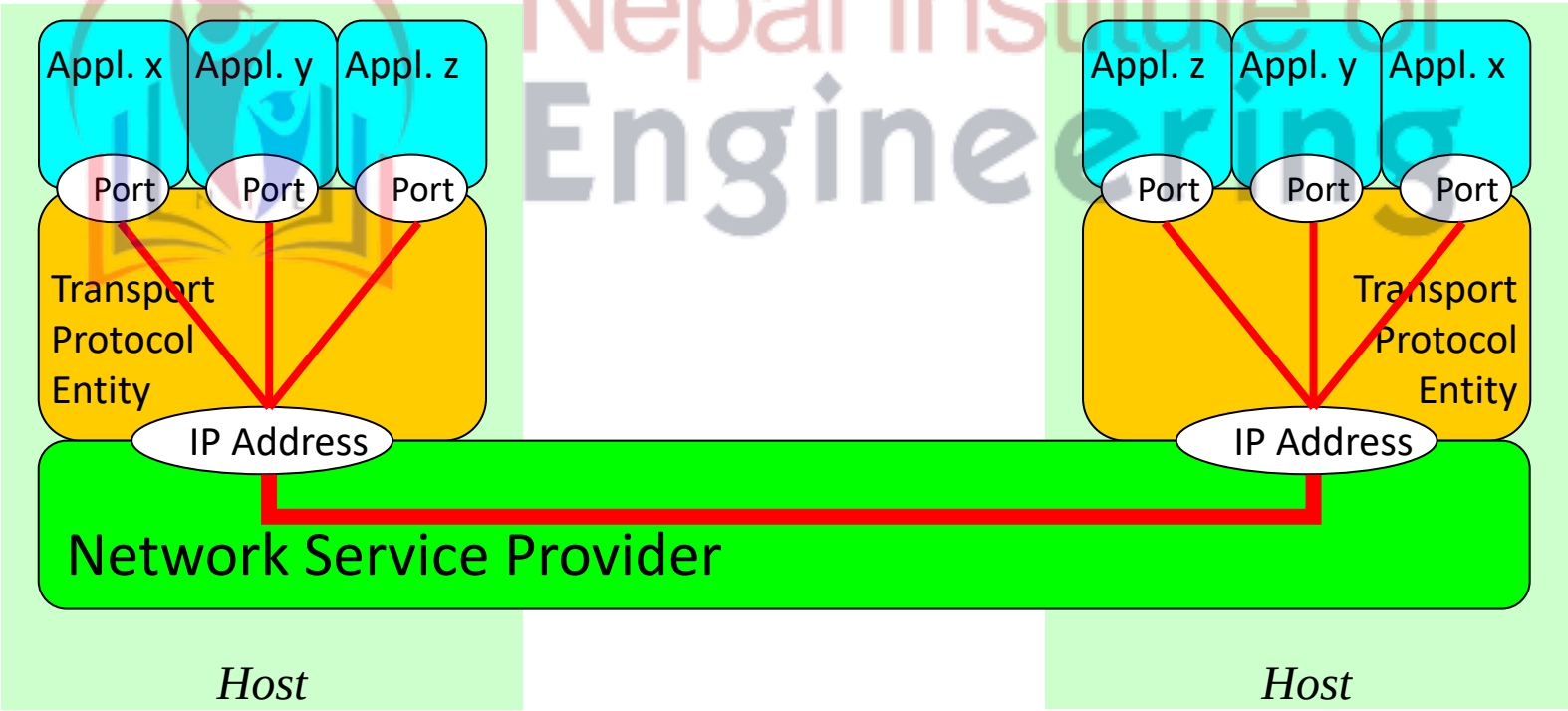
Flow Control and Buffering



(a) Chained fixed-size buffers. (b) Chained variable-sized buffers.
(c) One large circular buffer per connection.

Multiplexing and De-multiplexing

- Different transport connections(ports) uses same network connection(IP Address) to remote host
- Multiplexing at Sender
  - Gathering data from multiple sockets, enveloping data with header (later used for demultiplexing)
- De-Multiplexing at Receiver
  - Delivering received segments to correct socket



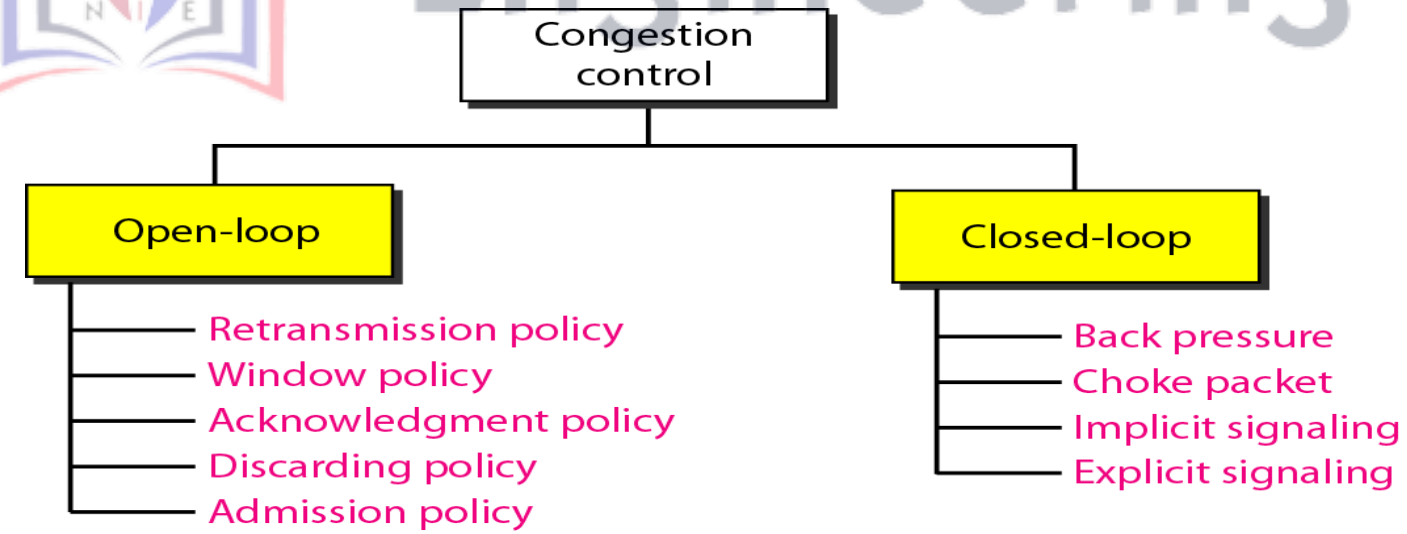
Congestion

- Overloading of routers with packets
- Routers receiving packets faster than they can forward
- Factors Causing Congestion
  - Packet arrival rate exceeds the outgoing link capacity.
  - Insufficient memory to store arriving packets
  - Bursty traffic
  - Slow processor
- Congestion control Techniques
  - Warning bit – Special Bit in Packet Header to notify sender about the congestion, piggy-backed with ACK
  - Choke packets – Control Packet sent by congested node. Sources that get choke packet must reduce transmission rate
  - Load shedding – buffer full=>simply discard packets.
  - Random early discard – discard packets before buffer is full
  - **Traffic shaping -** controls the *rate* at which packets are sent

CONGESTION CONTROL

- Congestion control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened.
- In general, we can divide congestion control mechanisms into two broad categories:
  - open-loop congestion control (prevention) and
  - closed-loop congestion control (removal)

Congestion control categories

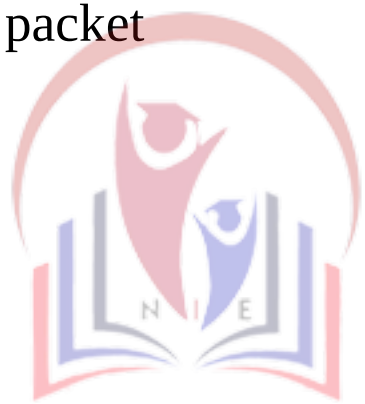


Open-loop congestion control

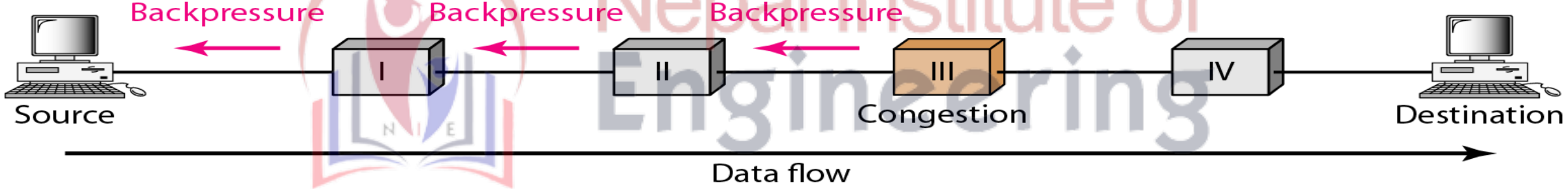
- Retransmission policy
  - The retransmission policy and the retransmission timers must be designed to optimize efficiency and at the same time prevent congestion.
- Window policy
  - The type of window at the sender may also affect congestion. The selective repeat window is better than go back n window for congestion.
- Acknowledgment policy
  - If the receiver does not acknowledge every packet it receives, it may slow down the sender and help prevent congestion
- Discard policy
  - In audio transmission, if the policy is to discard less sensitive packets when congestion is likely, the quality of sound is still preserved and congestion is prevented.
- Admission policy
  - Switches first check the resource requirement of a flow before admitting it to the network. A router can deny establishing a virtual circuit connection if there is congestion in the network or if there is a possibility of future congestion.

Closed-loop congestion control

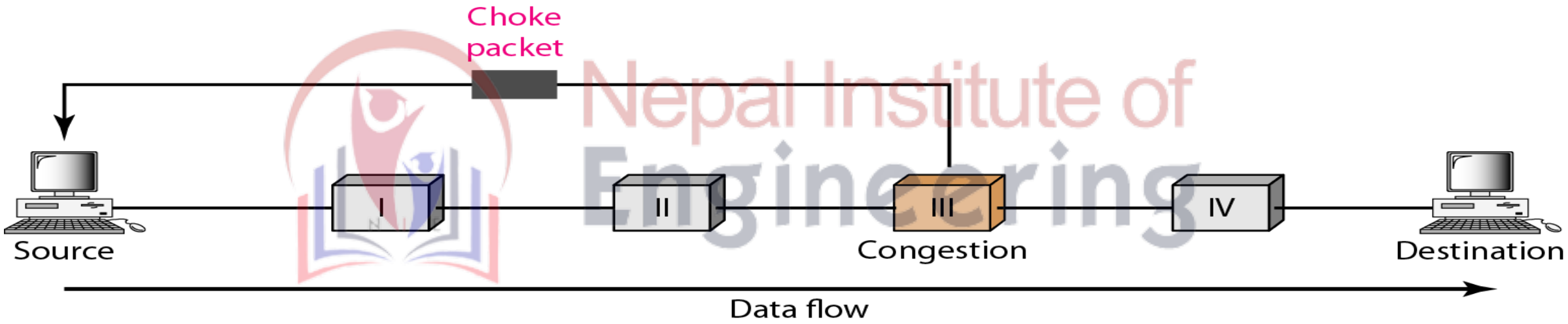
- Back pressure
  - informing the previous upstream router to reduce the rate of outgoing packets
- Choke point
  - is a packet sent by a router to the source to inform it of congestion is similar to ICMP's source quench packet



Backpressure method for alleviating congestion



Choke packet



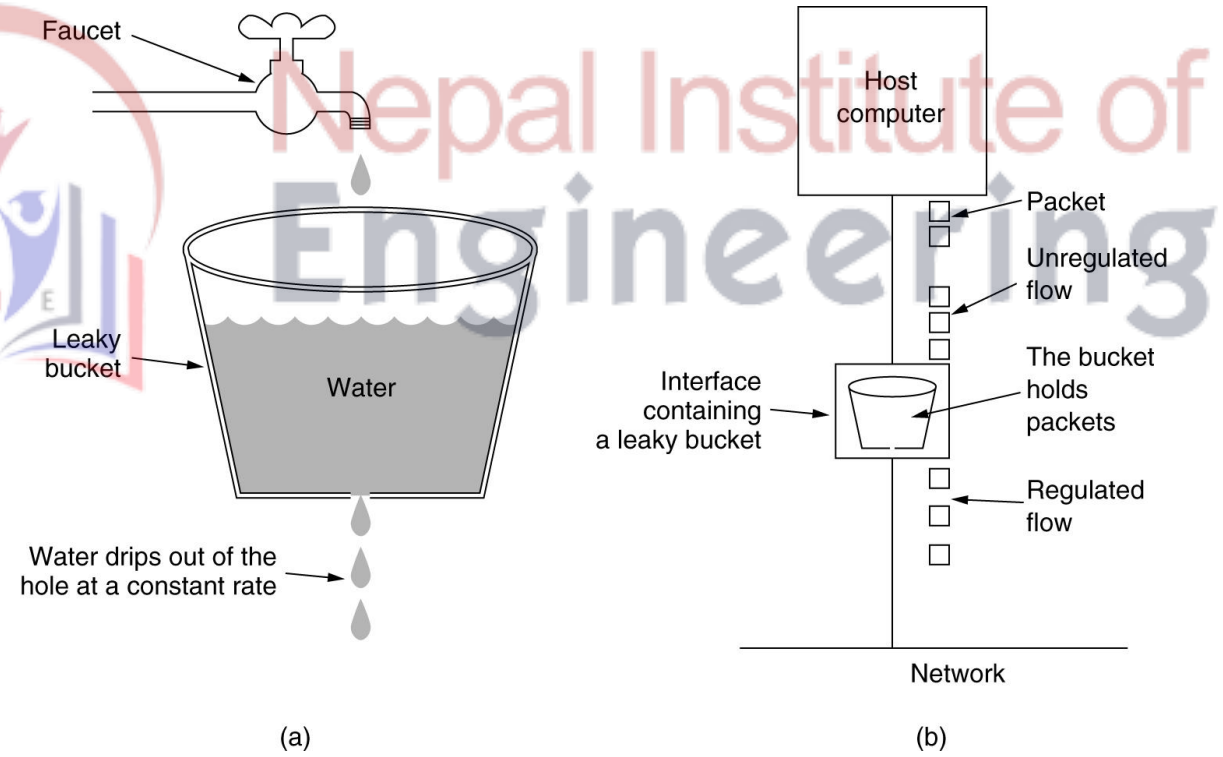
- **In implicit signaling:**
  - There is no communication between the congested node or nodes and the source.
  - The source guesses that there is a congestion somewhere in the network from other symptoms.
  - For example, when a source sends several packets and there is no acknowledgment for a while, one assumption is that the network is congested.
  - The delay in receiving an acknowledgment is interpreted as congestion in the network; the source should slow down.
- ***Explicit Signaling:***
  - The node that experiences congestion can explicitly send a signal to the source or destination.
  - The explicit signaling method, however, is different from the choke packet method.
  - In the choke packet method, a separate packet is used for this purpose; in the explicit signaling method, the signal is included in the packets that carry data.

Quality of Service-Traffic Shaping

- **Leaky Bucket Algorithm**
  - Single-server queue with constant service time. If the bucket (buffer) overflows then packets are discarded.
  - Enforces a constant output rate (average rate) regardless of the burstiness of the input.
  - The host injects one packet per clock tick onto the network. This results in a uniform flow of packets, smoothing out bursts and reducing congestion.

Traffic Shaping

- Leaky Bucket Algorithm



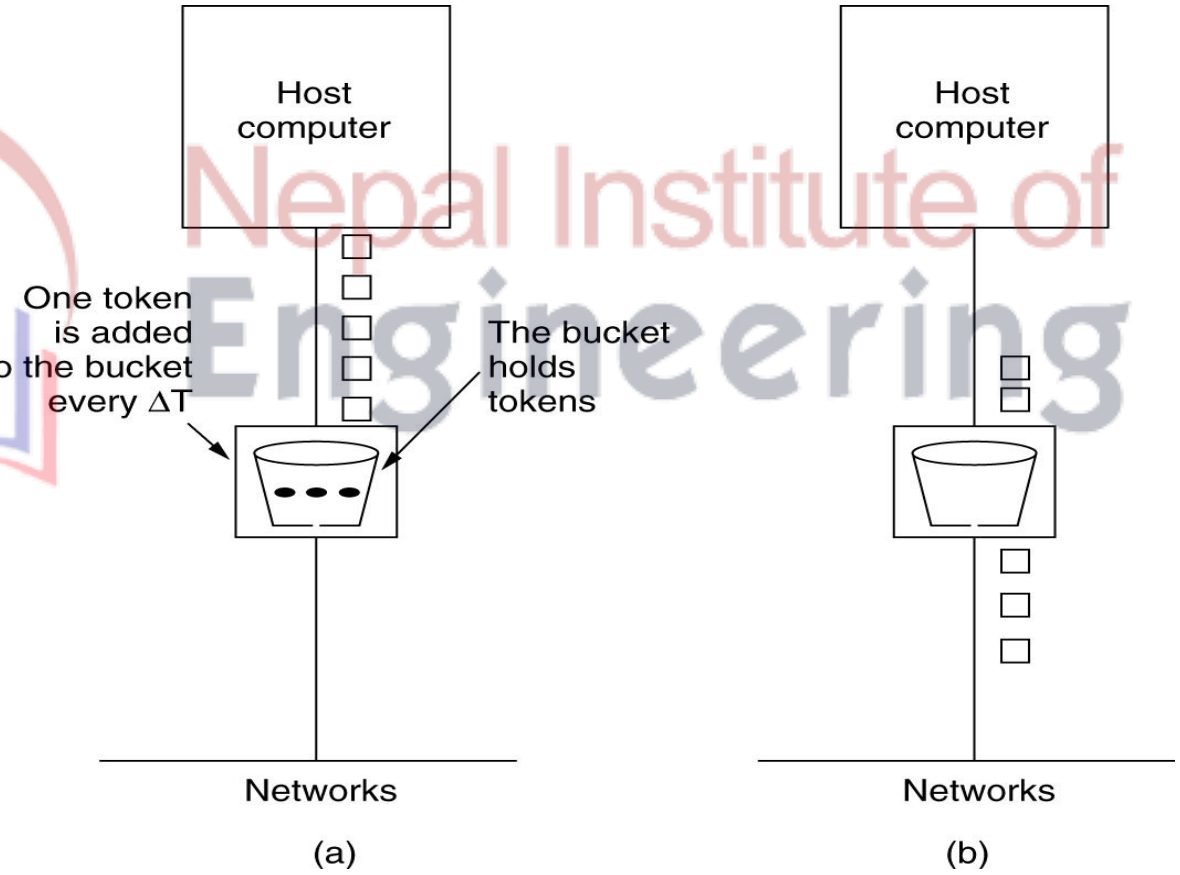
(a) A leaky bucket with water. (b) a leaky bucket with packets.

Traffic Shaping

- Traffic Shaping...
  - **Token Bucket Algorithm**
    - In contrast to the Leaky Bucket, the Token Bucket Algorithm, allows the output rate to vary, depending on the size of the burst.
    - The bucket holds tokens and to transmit a packet, the host must capture and destroy one token.
    - Tokens are generated by a clock at the rate of one token every Δt sec
    - Idle hosts can capture and save up tokens (upto max. bucket size) in order to send larger bursts later.

Traffic Shaping

- Token Bucket Algorithm



(a) Before. (b) After.

Traffic Shaping

- Leaky Bucket vs Token Bucket Algorithm
  - When bucket is full, LB discards packets; where as TB discards tokens.
  - With TB, a packet can only be transmitted if there is a token
  - LB sends packets at an average rate. TB allows for large bursts to be sent faster by speeding up the output.
  - TB allows saving up tokens (permissions) to send large bursts. LB does not allow saving.

Reliable Data Transfer (RDT)

- RDT is the mechanism where no transferred data bits are corrupted (flipped from o to 1, or vice versa) or lost, and all are delivered in the order in which they were sent.

Building a RDT protocol

- **1) Reliable Data transfer over a Perfectly Reliable Channel**
- We first consider the simplest case, in which the underlying channel is completely reliable.
- It is called finite-state machine (FSM).
- **2) Reliable Data Transfer over a channel with Bit Errors**
- A more realistic model of underlying channel is one in which bits in a packet may be corrupted.
- If receiver receives the packet, the receiver must acknowledge it to the sender whether the packet has received with error-free or not through these:
  - Positive Acknowledgement (ACK)
  - Negative Acknowledgement (NAK)
- If NAK provided, the sender should retransmit the packet.
- Such protocol is called ARQ (Automatic Repeat Request) protocols.

- Fundamentally, three additional protocol capabilities are required in ARQ protocols to handle the presence of bit errors :
- **Error Detection**
- Internet checksum field
- Error-detection and correction techniques
- Require extra bits (beyond the bits of original data to be transferred) to be sent from the sender to the receiver, these bits will be gathered into the packet checksum field.
- **Receiver Feed Back**
- Receiver provides feed back
- Positive (ACK) -1 value
- Negative (NAK) – 0 values
- **Retransmission**
- A packet that is received in error at the receiver will be retransmitted by the sender.
- These phenomena are called **stop-and-wait protocols**.

- An amazing case will occur if ACK or NAK is corrupted i.e. the sender could not get the feedback from sender.
- Consider different probabilities for handling corrupted ACKs or NAKs.
  - An alternative is to add enough checksum bits to allow the sender not only to detect, but also to receiver from bit errors. This solves immediate problem for a channel that can corrupt packets but not lose them.
  - Another approach is for the sender simply to resend the current data packet when it receives a garbled ACK or NAK packet. This introduces duplicate packets into the sender-to-receiver channel. The receiver doesn't know whether the ACK or NAK it last sent was received correctly at the ender. Thus, it cannot know whether an arriving packet contains new data or is a retransmission.
- A solution to this problem is to a new field called "sequence number" to the data packet. For this stop-and-wait protocol, a 1-bit sequence number will be ok.

- **3)Reliable Data Transfer Over A lossy Channel with Bit Errors**
- Suppose now that in addition to corrupting bits, the underlying channel can lose packets as well.
- The sender must get information of packet loss on the way from the receiver so that the sender can retransmit.
- The sender must clearly wait at least as long as a round-trip delay between the sender and the receiver.
- If ACK is not received within this time, the packet is retransmitted.
- If a packet experiences a particularly large delay, the sender may retransmit the packet even though neither the data packet nor its ACK have been lost.
- This introduces the possibility of duplicate data packets in the sender-to-receiver channel.
- For all this, we can do is retransmit.

- But implementing a time-based retransmission mechanism requires a “countdown timer” that an interrupt the sender after a given amount of time has expired.
- The sender will thus need to be able to
  - Start the timer each time a packet (either a first time packet or a retransmission) is sent.
  - Respond to a timer interrupt (taking appropriate actions)
  - Stop the timer.

Reliable Data Transfer Protocol

- **1. Pipelined**
- **2. Go-Back-N(GBN)**
- **3. Selective Repeat (SR)**

Pipelined Reliable Data Transfer Protocol

- Instead of sending a single packet in stop and wait manner, the sender is allowed to send multiple packet without waiting for acknowledgements, as illustrate in fib (b).
- fig (b) shows that if the sender is allowed to transmit three packets before having to wait for acknowledgement the utilization of the sender is essentially tripled.
- Since the many in-transmit sender-to-receiver packets can be visualized as a filling a pipeline, this technique is known as pipelining.

- **Consequences of pipelined protocol**
- Increment in the range of sequence numbers.
- Sender and receiver have to buffer more than one packet.
- Range of sequence numbers and the buffering requirements will depend on the manner in which a data transfer protocol responds to lost, corrupted and overly delayed packets.

Go-Back-N (GBN):

- In a GBN, protocol, the sender is allowed to transmit multiple packets (when available) without waiting for an acknowledgement but is allowed to have no more than some maximum allowable number, N, of an un acknowledged packets in the pipeline.
- GBN protocol itself as a sliding window protocol.

Selective Repeat (SR):

- GBN itself suffers from performance problems.
- Many packets can be in the pipeline when the window size and bandwidth-delay product are both large.
- A single packet error can thus cause GBN to retransmit a large number of packets, many unnecessarily.
- As the probability of channel error increased, the pipeline can become filled with these unnecessary retransmissions.