BCS502 – COMPUTER NETWORKS

Faculty:

Prof. Ashok Herur

ashok. herur @ eastpoint. ac. in

VTUSYNC.IN

Module 4

Transport Layer

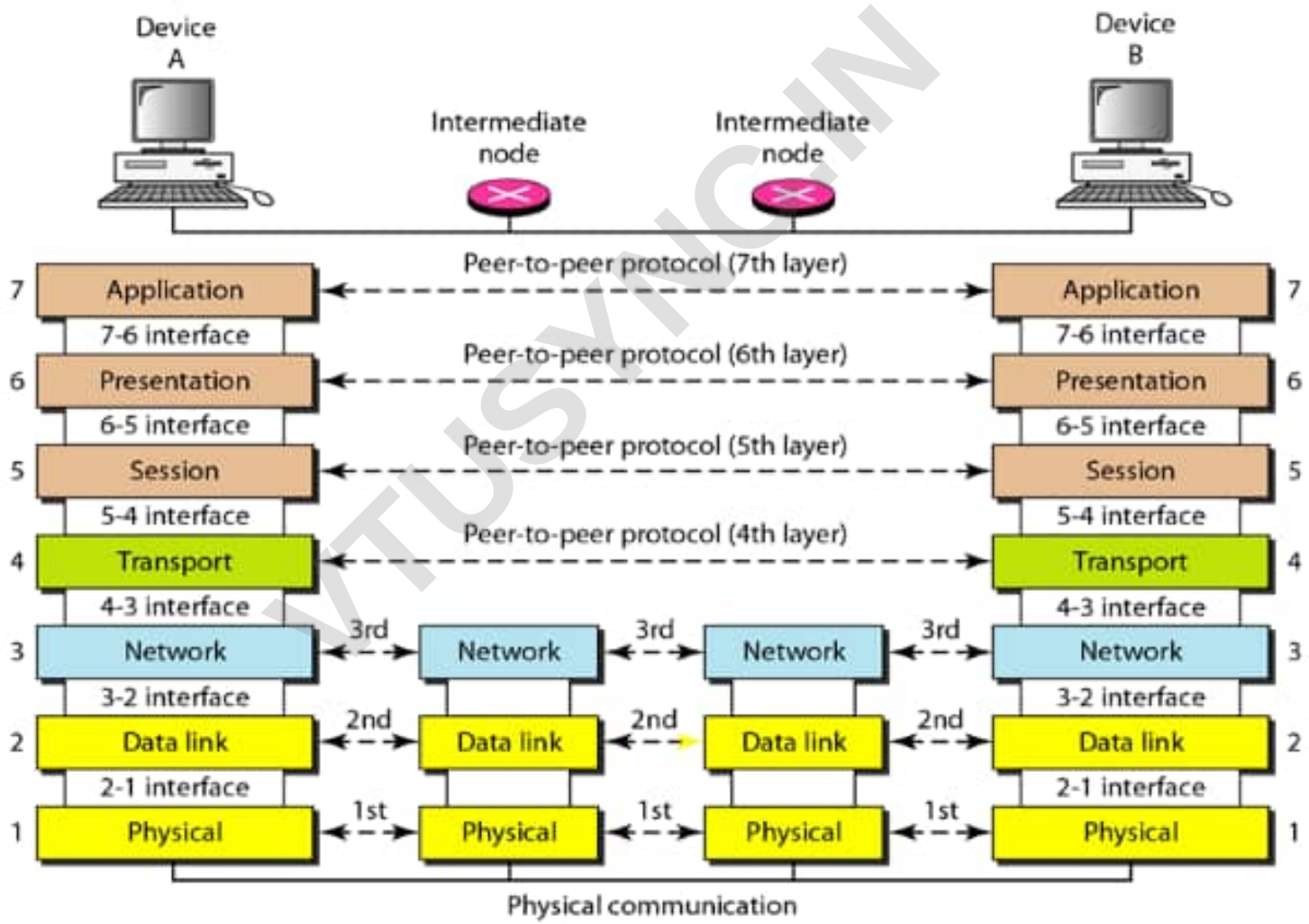
VTUSYNC.IN

Main topics in Module 4

- The Transport service
- Different transport layer protocols – TCP and UDP
- Flow control and Error control
- Congestion control

VTUSYNC.IN

OSI Reference model



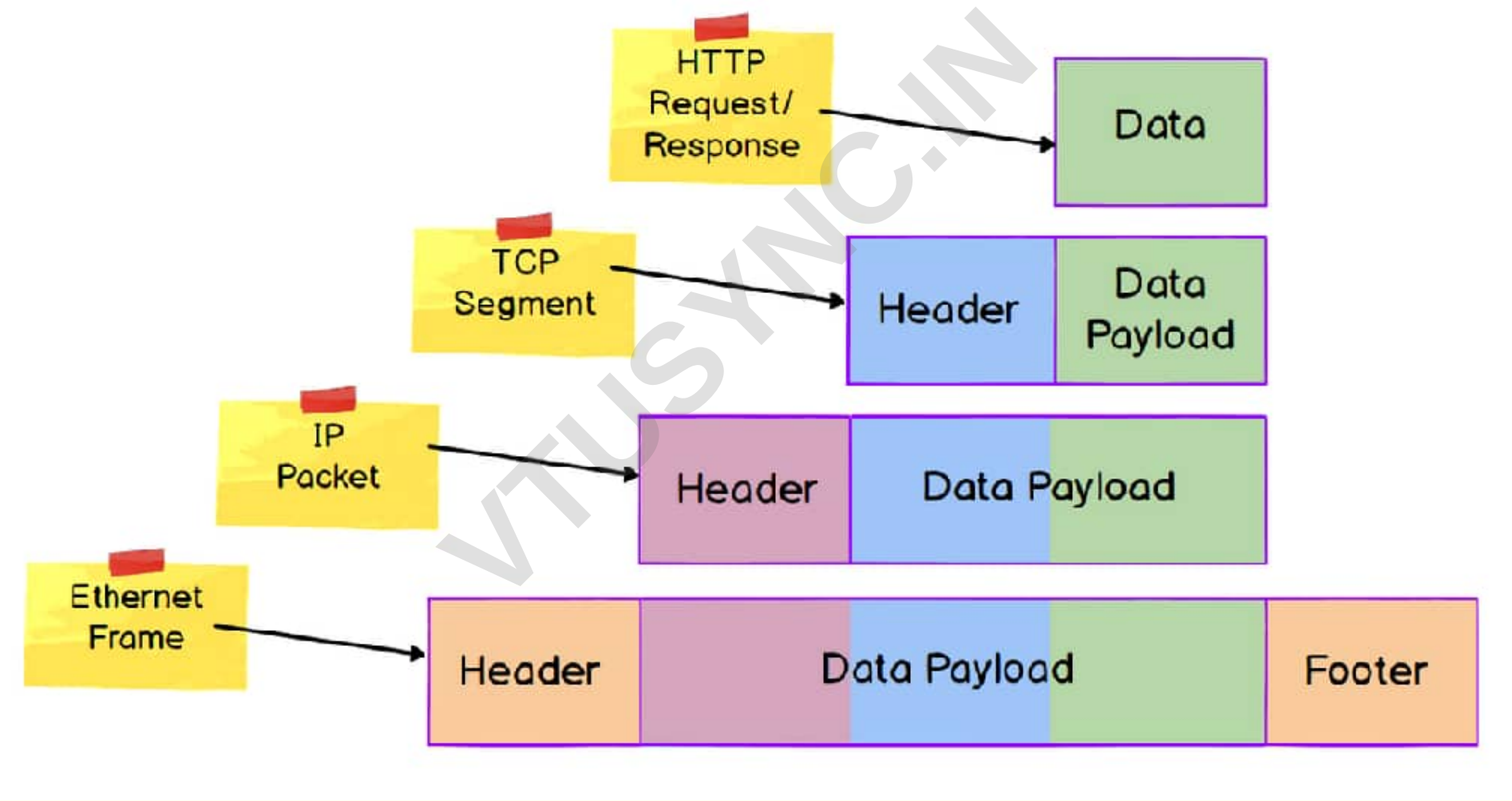
VTUSYNC.IN

Functions of the Transport Layer

- It is the lowest host-to-host protocol.
- Facilitates communication between host processes.
- Divides the message into data segments on the sending side, and reassembles them on the receiving side.
- Takes complete responsibility for a reliable, end-to-end transport of the message.
- Packets can be: Corrupted, or Lost, or Delivered out of sequence.
- Also takes care of Flow control between the hosts.

VTUSYNC.IN

Segments in the Transport Layer



VTUSYNC.IN

Transport entity

- The software and hardware within the Transport layer that does the work is called the Transport entity.
- The Transport entity is generally located (when it comes to Internet) in the Operating System, or in a library package bound into network applications.
- It can also be in a separate user process or even in the Network Interface card.

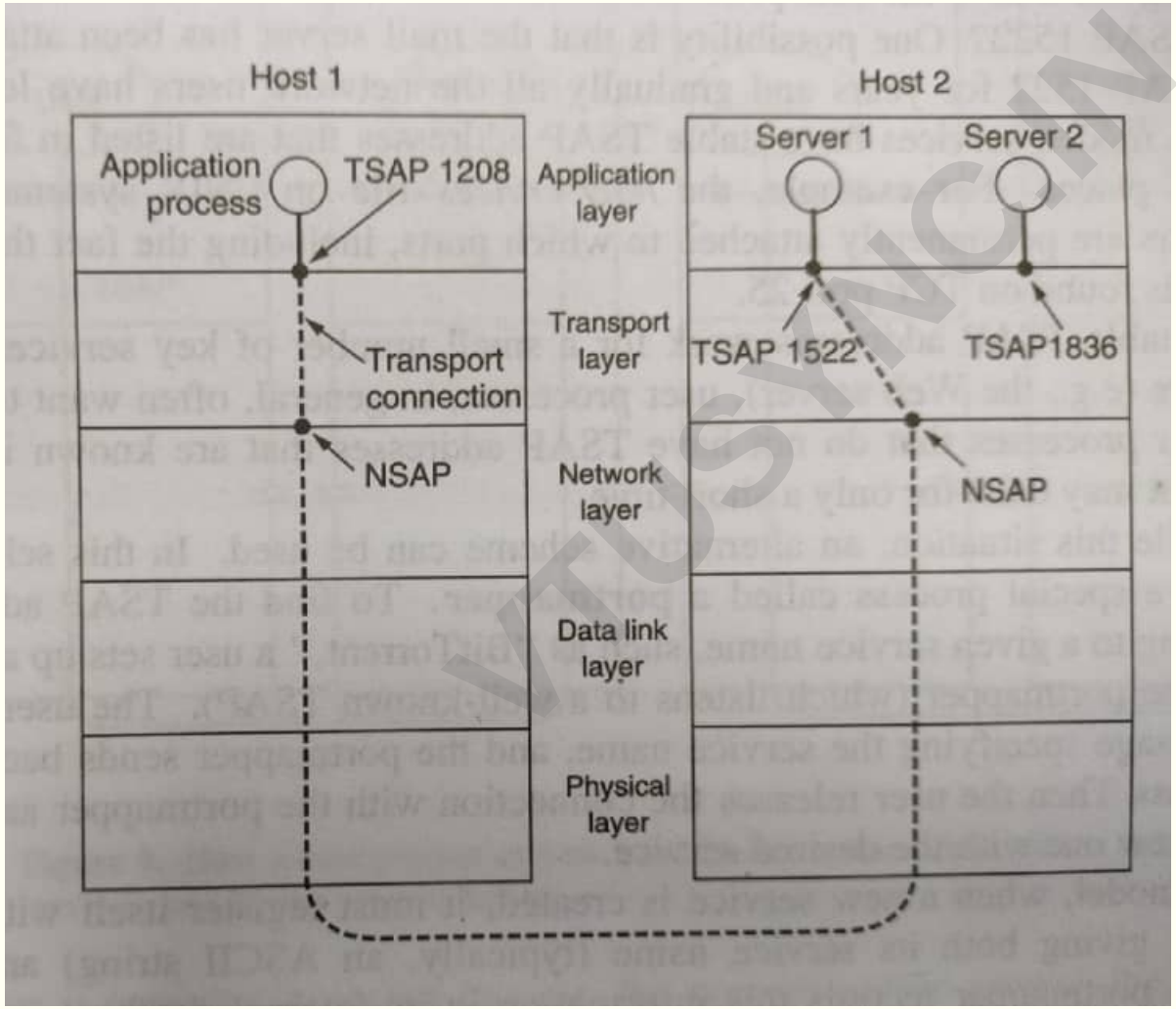
VTUSYNC.IN

Transport layer addressing

- When an application process wishes to set up a connection to a remote application process (on another host), it must specify it unambiguously.
- In the Internet, such addresses are called Ports. (in generic terms, an address is called TSAP – Transport Service Access Point).
- This is similar to an IP address in the Network layer (NSAP – Network Service Access Point).
- However, note that a host could be running many applications simultaneously, in which case a single IP address will lead to many Port addresses.

VTUSYNC.IN

Transport layer addressing



VTUSYNCIN

The Transport layer at the destination host should know the port to which the data is to be delivered.

Eg. shown is from TSAP 1208 on Host 1 to TSAP 1522 on Host 2.

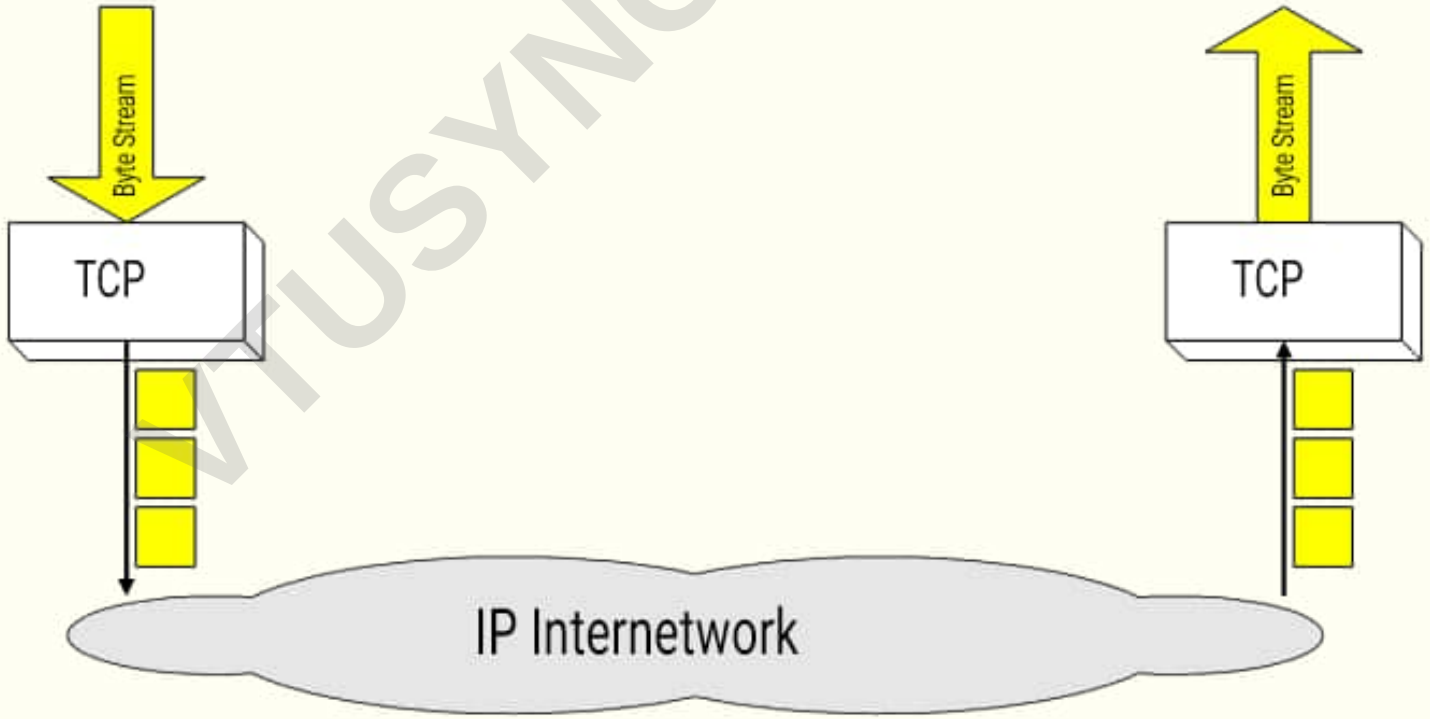
Connection-oriented vs. Connectionless sessions

- The Network layer too had Connectionless (Datagram Packet Switching) and Connection-oriented (Virtual Circuit Packet Switching) transmissions.
- However, they were restricted to transmission from the transmitting router (representing the transmitting host) to the terminal router (representing the destination host).
- Any deficiencies in the service has to be handled by the Transport layer.
- Transport layer has two main protocols:
  - TCP (Transport Control Protocol) – Connection-oriented
  - UDP (User Datagram Protocol) – Connectionless.

VTUSYNC.IN

Transmission Control Protocol (TCP)

- Provides a reliable, end-to-end, byte-stream transmission over an unreliable internetwork.



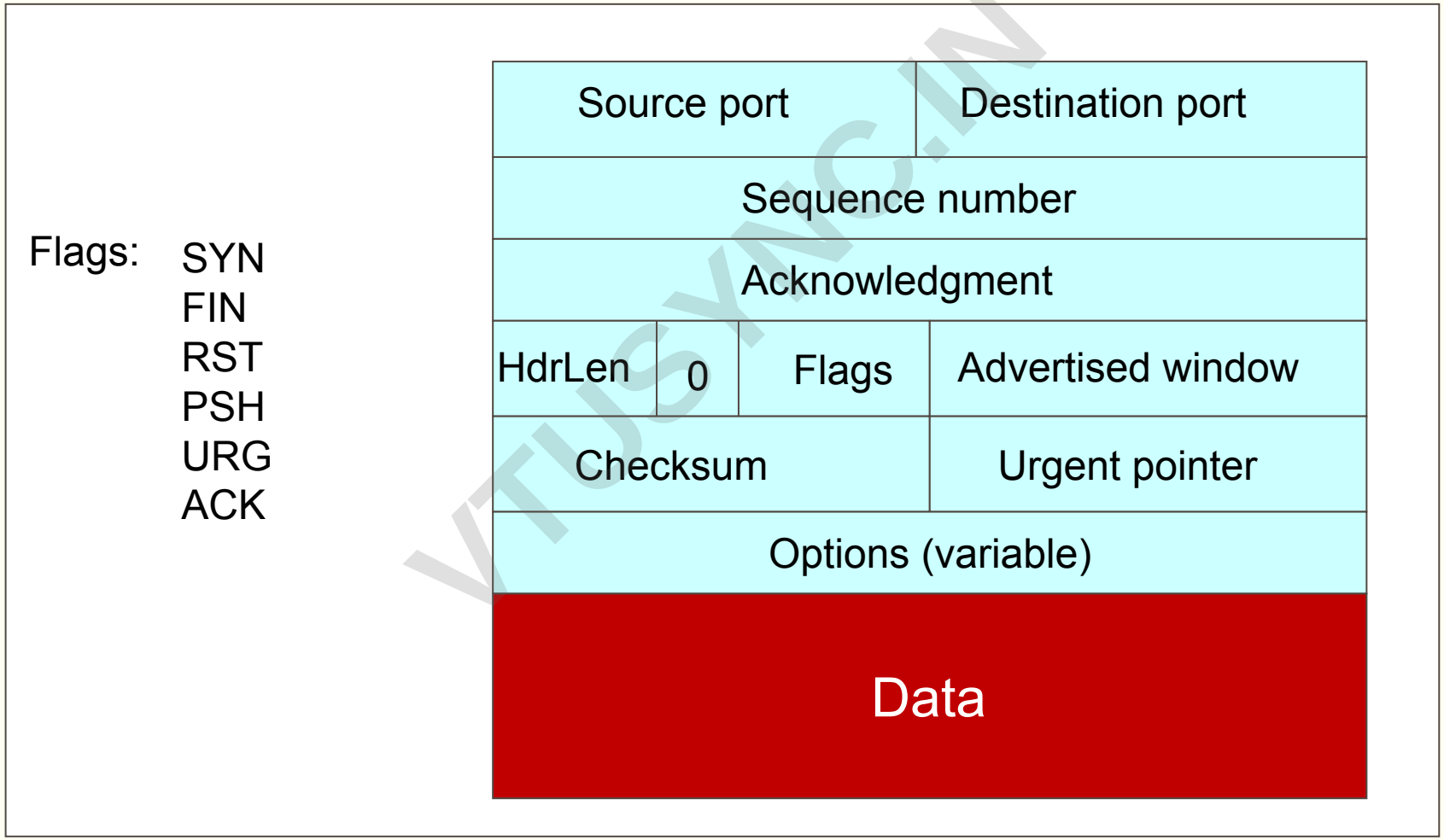
VTUSYNC.IN

Transmission Control Protocol (TCP)

- Is Connection-oriented
  - Explicit set-up and tear-down of TCP session
- Offers Stream-of-bytes service
  - Receives a stream of bytes from higher layers.
  - Passes them as Segments to the lower layers.
- Provides Reliable, in-order delivery
  - Checksums to detect corrupted data
  - Acknowledgments & retransmissions for reliable delivery
  - Sequence numbers to detect losses and reorder data

VTUSYNC.IN

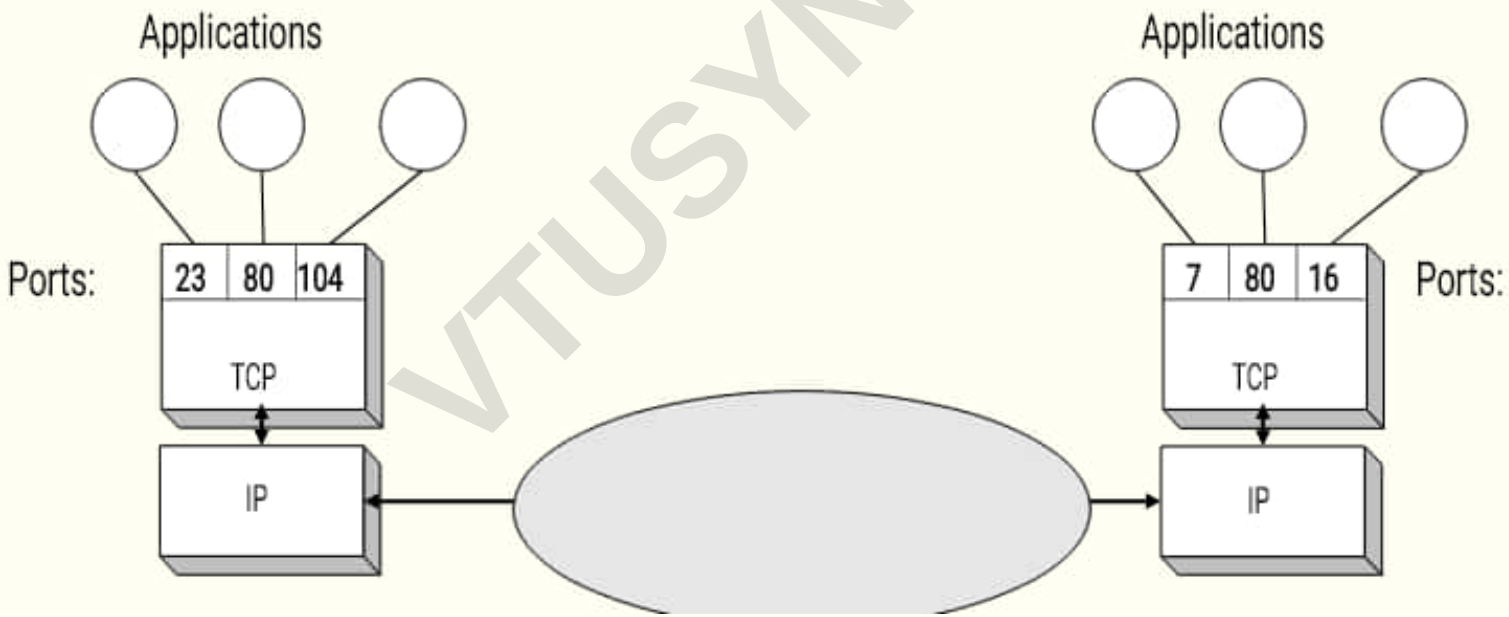
TCP header



VTUSYNC.IN

TCP Header fields

- **Port Number:**
  - A port number identifies the endpoint of a connection.
  - A pair `<IP address, port number>` identifies one endpoint of a connection.



VTUSYNC.IN

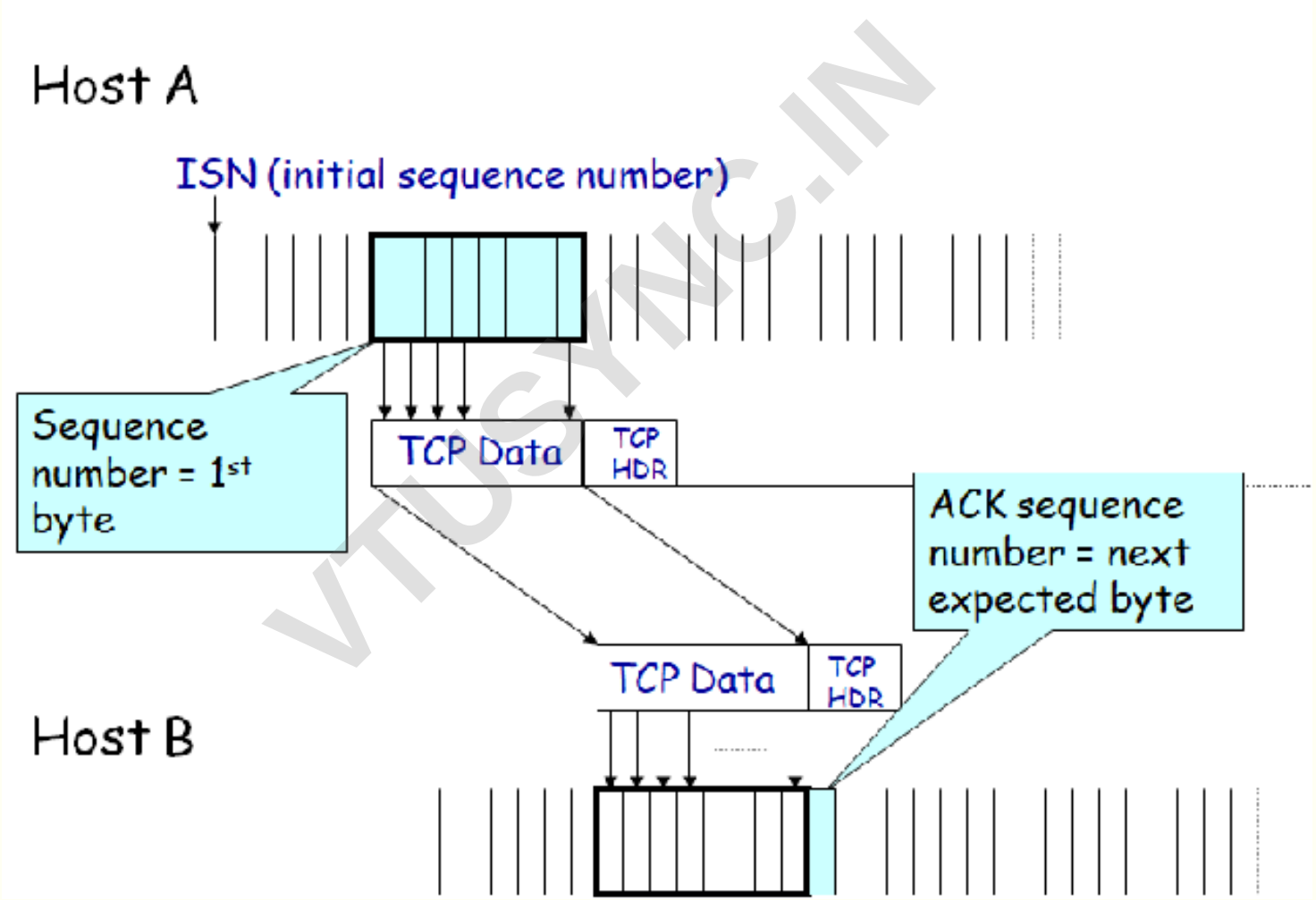
TCP Header fields

- **Sequence Number:**
  - The range of the 32-bit SeqNo is

    $0 <= SeqNo <= 2^{32} - 1 \approx 4.3$ Giga
  - Each sequence number identifies a byte in the byte stream.
  - Initial Sequence Number (ISN) of a connection is set during connection establishment.

VTUSYNC.IN

TCP Header fields



VTUSYNC.IN

TCP Header fields

- **Acknowledgement Number:**
  - Acknowledgements are piggybacked.
  - A segment from A -> B can contain an acknowledgement for a data sent in the B -> A direction.
  - If a host sends an AckNo in a segment it sets the "ACK flag"
  - The AckNo contains the SeqNo of the next byte that a host wants to receive.
  - Example: The acknowledgement for a segment with sequence numbers 0-1500 is AckNo=1501.

VTUSYNC.IN

TCP Header fields

- **Header Length:**
  - Length of header in terms of 32-bit words (4 bytes)
  - The TCP header has a variable length (with minimum 20 bytes)
  - This 4-bit field can hold numbers from 5 (20 bytes) to 15 (60 bytes)
- **Flags:**
  - SYN: Synchronize sequence numbers
    - Sent in the first packet when initiating a connection
  - FIN: Sender is finished with sending
    - Used for closing a connection
    - Both sides of a connection must send a FIN

VTUSYNC.IN

TCP Header fields

- **Flags: (contd)**
  - RST: Causes the receiver to Reset the connection
    - Receiver terminates the connection and indicates higher layer application about the reset
  - ACK: Acknowledgement Number is valid
  - URG: Urgent pointer is valid
    - If the bit is set, the following bytes, in the range given below, contain an urgent message:

      SeqNo <= urgent message <= SeqNo+urgent pointer
  - PSH: PUSH flag is used to tell the receiver to pass all data (that it currently has) to the application.
    - Normally set by sender when the sender's buffer is empty.

VTUSYNC.IN

TCP Header fields

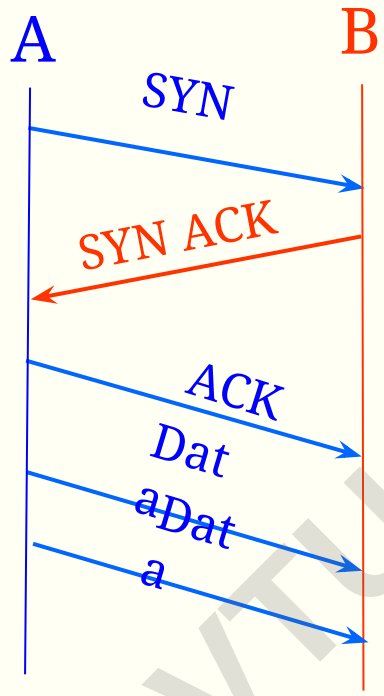
- **Window Size:**
  - Each side advertises the size of its sliding window.
  - Window size is the maximum number of bytes that a receiver can accept.
  - Maximum window size is 2^{16}-1= 65535 bytes
- **TCP Checksum:**
  - TCP checksum covers over both TCP header **and** TCP data
- **Urgent Pointer:**
  - Only valid if **URG** flag is set

VTUSYNC.IN

Establishing a connection (Hand-shaking)

VTUSYNC.IN

Establishing a TCP Connection: 3-way handshake.



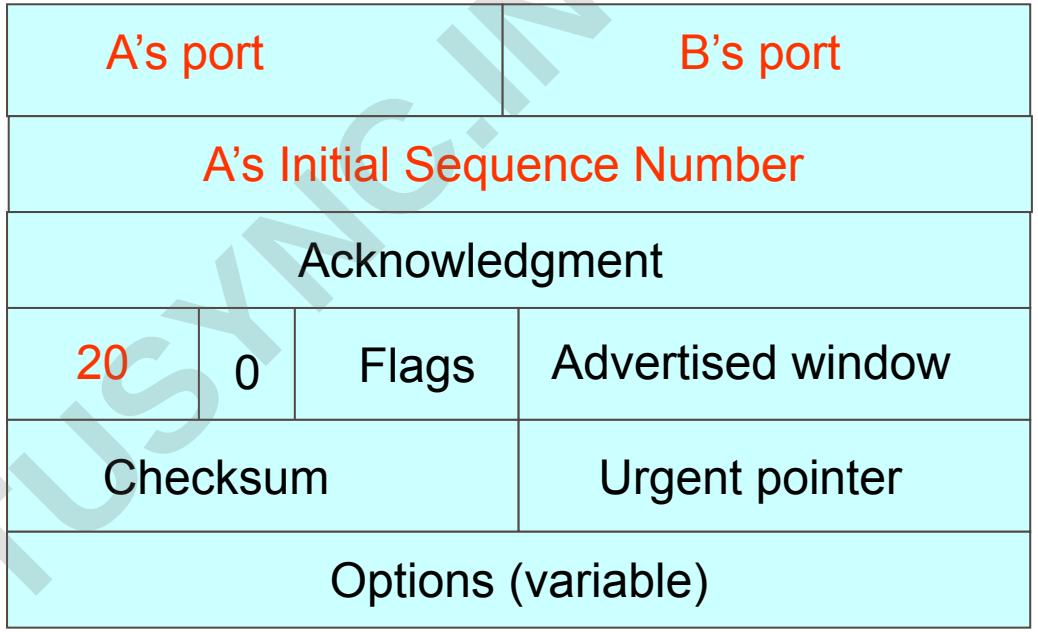
Each host tells its ISN (Initial sequence number) to the other host.

VTUSYNC.IN

- Three-way handshake to establish connection
  - Host A sends a **SYN** (open) to the host B
  - Host B returns a SYN acknowledgment (**SYN ACK**)
  - Host A sends an **ACK** to acknowledge the SYN ACK

Step 1: A's Initial SYN Packet

Flags: SYN
FIN
RST
PSH
URG
ACK



VTUSYNC.IN

A tells B it wants to open a connection…

Step 2: B's response (SYN-ACK) packet to A

Flags:
SYN
FIN
RST
PSH
URG
ACK

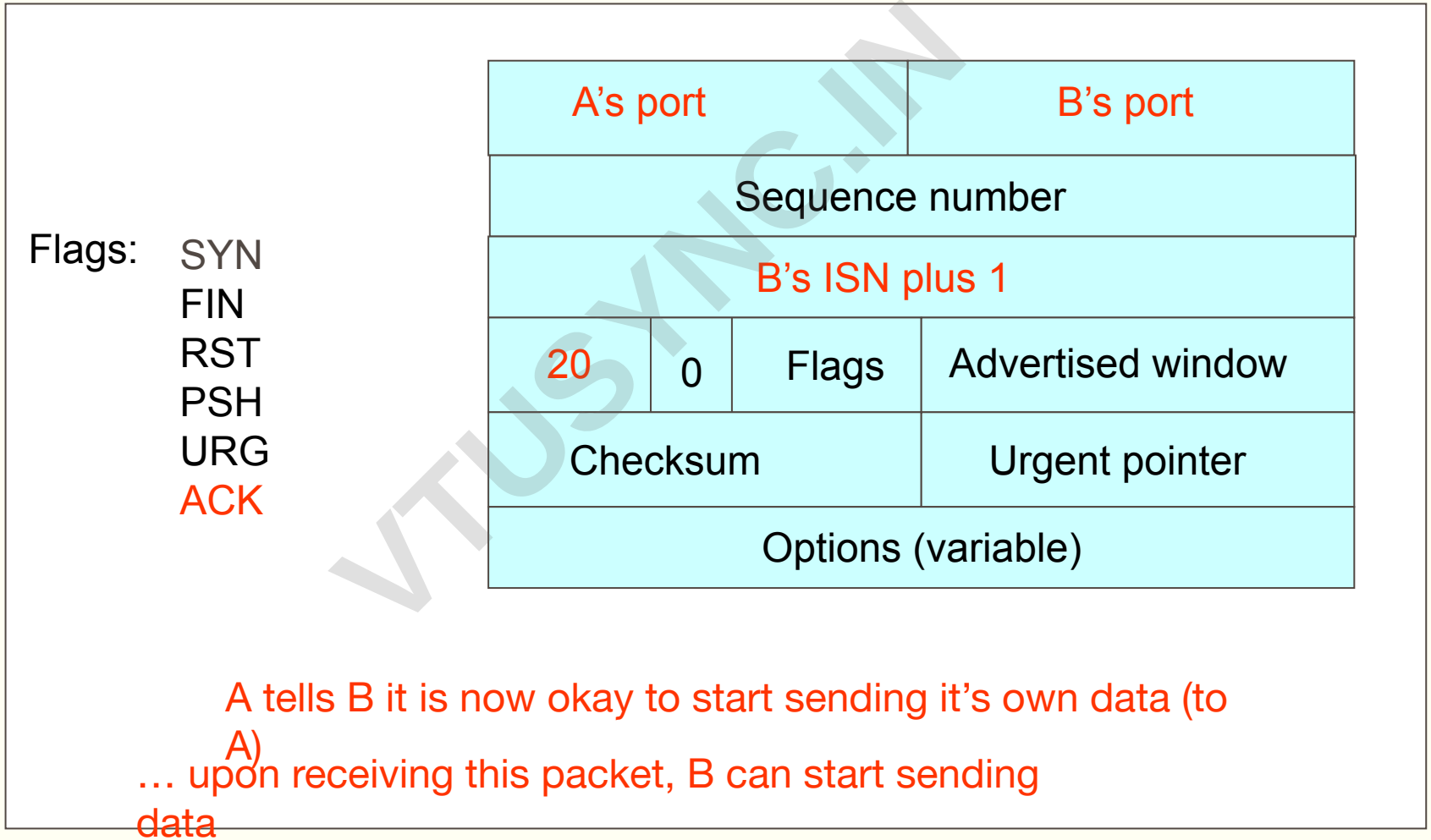
B's port	A's port
B's Initial Sequence Number	
A's ISN plus 1	
20 | 0 | Flags	Advertised window
Checksum	Urgent pointer
Options (variable)	

B tells A it accepts, and is ready to hear the next byte…

… upon receiving this packet, A can start sending data

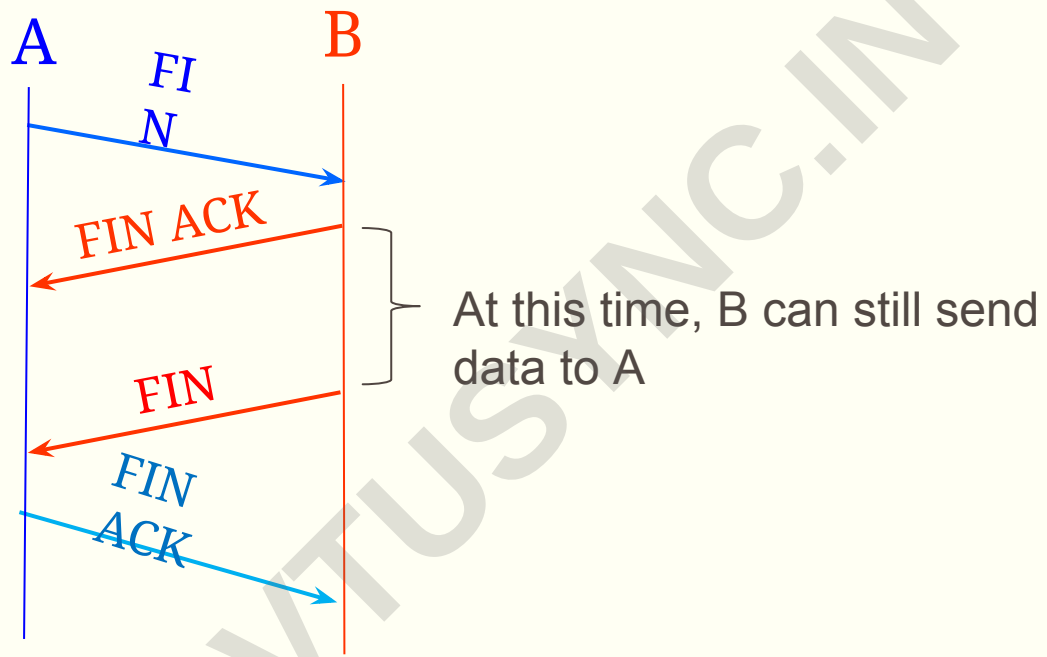
VTUSYNC.IN

Step 3: A's ACK of the SYN-ACK



VTUSYNC.IN

Tearing down a TCP Connection



VTUSYNC.IN

- Each end of the data flow must be shut down independently.
- If one end is done, it sends a FIN segment. This means that no more data will be sent.

Error control in TCP

- The Data Link layer implements error control measures like using a CRC Checksum and using some ARQ protocols.
- If every frame is verified, on EVERY link, at the Data Link layer, can something be wrong, end-to-end, when the message reaches the destination host?
- While the Link layer checks take care of the errors that may occur on the various links, they cannot take care of the errors that may occur INSIDE the routers.
- At every router, various fields (TTL, Fragment Offset, etc) will / may change, leading to a change in the Packet Header checksum.

VTUSYNC.IN

Flow control in TCP

- The Data Link layer implements flow control measures like using Sliding window protocols.
- What would be the optimum size of a window?
- At the Data Link layer (for a link between adjacent nodes), the window size can be pretty small (1, 2, 4,…) since the Bandwidth-Delay product is low.
- Bandwidth-Delay product broadly indicates how many bits (segments) could have been transmitted by the time a sender receives the acknowledgment of a message segment sent earlier.

VTUSYNC.IN

Flow control in TCP

- However, at the Transport Link layer (host to host), the Bandwidth-Delay product will be high because of the high delay required to traverse the entire path to the destination.
- Therefore, the Window size here (at the Transport Layer) has to be much larger.
- The same logic applies to the Buffering at the transmitting host.
  - It has to be much larger than that used at an intermediate router.

VTUSYNC.IN

Wireless issues in TCP

- Transport layer protocols such as TCP that implement congestion control should be independent of the underlying Network and Data Link layer technologies.
- But, in practice, there are various issues with wireless links / networks.
- The main issue is that % packet loss is often used as a measure of congestion.
- This % packet loss is normally high on wireless links / networks, due to the higher rate of transmission error (due to noise).

VTUSYNC.IN

Wireless issues in TCP

- By the time, this % packet loss reaches around 10%, the connection is effectively stopped because of the throttling by the congestion control mechanisms.
- One solution is to differentiate the packet losses due to transmission error and due to insufficient bandwidth.
- Things will be more complicated when the path contains a few wired links and a few wireless links.

VTUSYNC.IN

User Datagram Protocol (UDP)

VTUSYNC.IN

User Datagram Protocol (UDP)

- The main characteristics of UDP are as follows
  - UDP implements a connectionless transfer.
    - It does not require connection establishment prior to data transfer
  - UDP service is unreliable
    - UDP does not guarantee the delivery of datagram to the destination
    - It does not guarantee the correct order of the datagrams.
  - UDP computes the checksum for the entire header plus data
  - UDP does not buffer the incoming datagrams.
  - UDP does not have the feature to segment long messages.

VTUSYNC.IN

User Datagram Protocol (UDP)

- UDP results in speedier communication because it does not spend time forming a firm connection with the destination before transferring the data.
- User Datagram Protocol (UDP) is used for:
- Sending relatively small amounts of data, eliminating concerns regarding controlling errors or the flow of the packets
- Routing update protocols such as Routing Information Protocol (RIP) and Network control protocols like SNMP.
- Multicasting because UDP works well with packet switching.

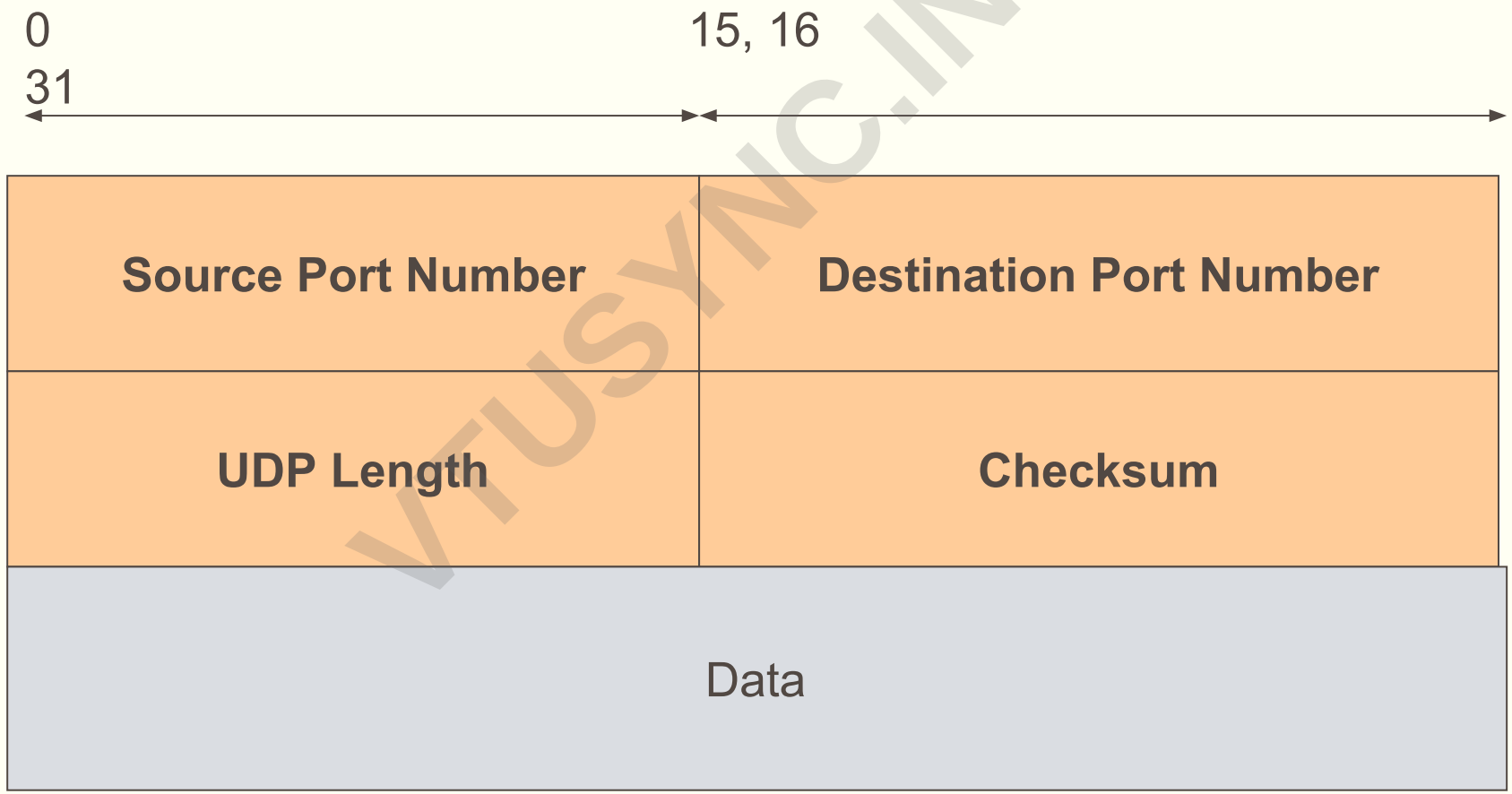
VTUSYNC.IN

User Datagram Protocol (UDP)

- Real-time applications in which the information needs to be delivered quickly and smoothly.
- It is specifically chosen for time-sensitive applications like gaming, playing videos, or Domain Name System (DNS) lookups (convert Domain names to IP addresses).
- For users, it is better to have the overall transmission arrive on time than wait for it to get there in a near-perfect state.
  - For this reason, UDP is commonly used in Voice over Internet Protocol (VoIP) applications as well.

VTUSYNC.IN

UDP header



VTUSYNC.IN

User Datagram Protocol (UDP)

- Source port number identifies the sender's port, when used, and should be assumed to be the port to reply to if needed.
- If not used, it should be zero.
- If the source host is the server, the port number is likely to be a well-known port number from 0 to 1023.
- If the source host is the client, the port number is likely to be an ephemeral (temporary, short-lived) port number.
- In networking, you might use well-known ports (0 to 1023), registered ports (1024 to 49151), and dynamic or ephemeral ports (49152 to 65535), as defined by the Internet Assigned Numbers Authority (IANA).

VTUSYNC.IN

User Datagram Protocol (UDP)

- Destination port number identifies the receiver's port and is required.
- UDP Length field specifies the length in bytes of the UDP header and data.
  - The minimum length is 8 bytes, the length of the header.
  - The field size sets a theoretical limit of 65,535 bytes (8-byte header + 65,527 bytes of data) for a UDP datagram.
- The Checksum field may be used for error-checking of the header and data (by the Application layer); The field carries all-zeros if unused.

VTUSYNC.IN

Congestion control

VTUSYNC.IN

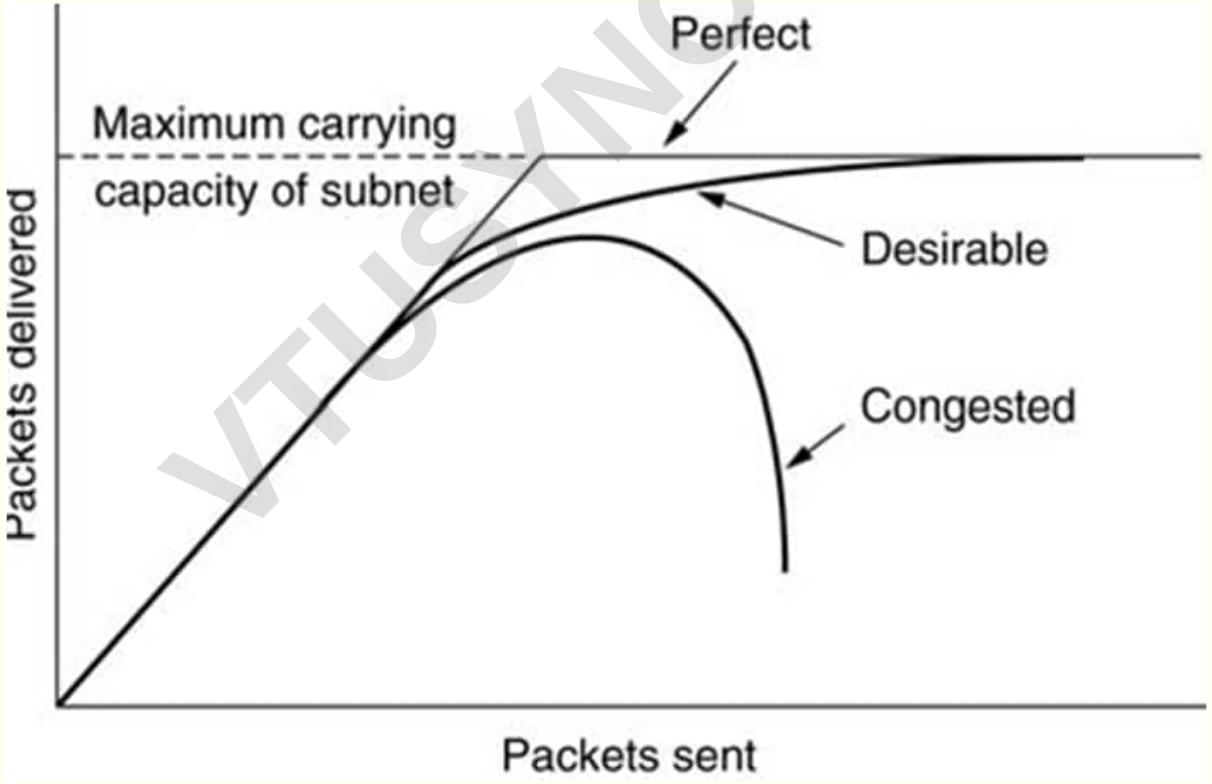
Congestion control

- **Congestion control** has to ensure that the subnet is able to carry the traffic offered by the hosts.
  - It is a global issue, involving all hosts, all routers and all links in the network.
- When too many packets are present in a small part of the subnet, it leads to congestion.
  - Then, the data carrying capacity of the subnet reduces.
- When the number of packets dumped onto the subnet by the hosts is within the carrying capacity, they are all delivered (except the ones with transmission errors)
  - The number delivered is proportional to the number sent by the hosts

VTUSYNC.IN

Congestion control

- As traffic increases beyond the capacity, the network cannot cope and packets are lost.



VTUSYNC.IN

Causes of congestion

- Load greater than the resources in a part of the network
  - More traffic and low bandwidth lines
- Insufficient buffer at the routers
  - Having a very large buffer is counter-productive (why?)
- Slow processors at the routers

VTUSYNC.IN

Measures of congestion

- Percentage of packets discarded for lack of buffer space
- Average queue length at the routers
- Average packet delay
- Standard deviation of the packet delay
- Number of packets that are timed-out and retransmitted

In all the above cases, a rising number indicates growing congestion.

VTUSYNC.IN

Congestion control algorithms

- In spite of the routers adapting to the traffic conditions, and changes in topology (adaptive routing), traffic can build up in a part of the network.
- The basic principle behind all the Congestion Control algorithms is: Reduce load and / or Increase capacity.
  - Often, it is difficult to increase capacity temporarily and at short notice.
  - Therefore, most of the algorithms concentrate on reducing (regulating) the load.

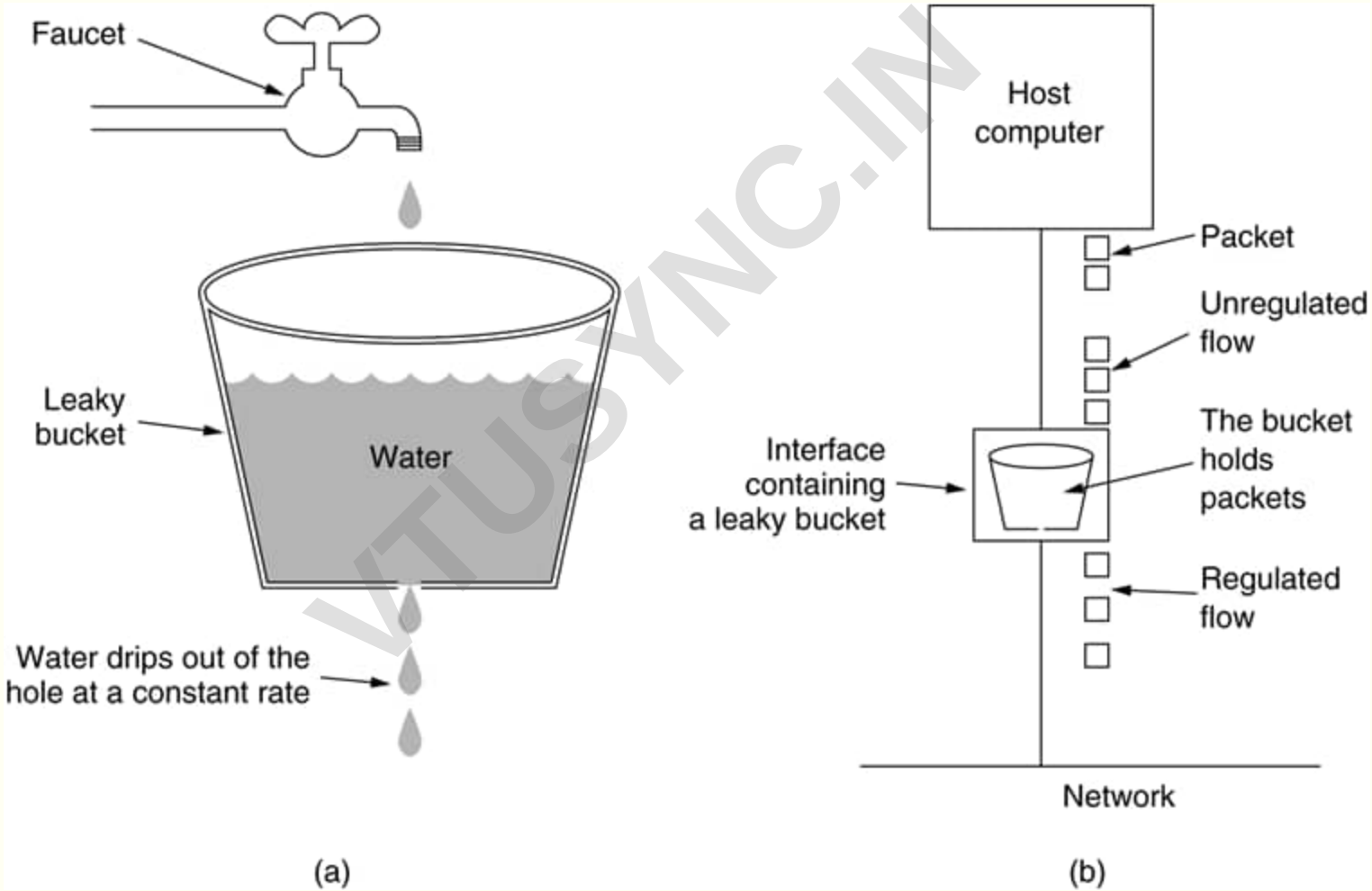
VTUSYNC.IN

Congestion control algorithms

- Common techniques:
- **Admission Control techniques:**
  - Leaky bucket algorithm
  - Token bucket algorithm
- **Traffic throttling techniques:**
  - Choke packet algorithm
  - Explicit Congestion Notification
  - Load shedding

VTUSYNC.IN

Leaky Bucket algorithm



VTUSYNC.IN

Leaky Bucket algorithm

- The leaky bucket tries to ensure that the number of packets put out on to the network (on an average) is constant, regardless of the burstiness of the input; It does nothing when input is idle or the rate is below the defined constant rate.
- 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.
- It is implemented as a single-server queue with constant service time.
  - If the bucket (buffer) overflows, then packets are discarded.

VTUSYNC.IN

Leaky Bucket algorithm

- When packets are of variable length, it is better to allow a fixed number of bytes per tick.
- The leaky bucket algorithm imposes the same restriction on all hosts (some sending almost continuously and some sending occasionally).
  - In short, it does not allow any burstiness beyond the constant output rate that has been defined.

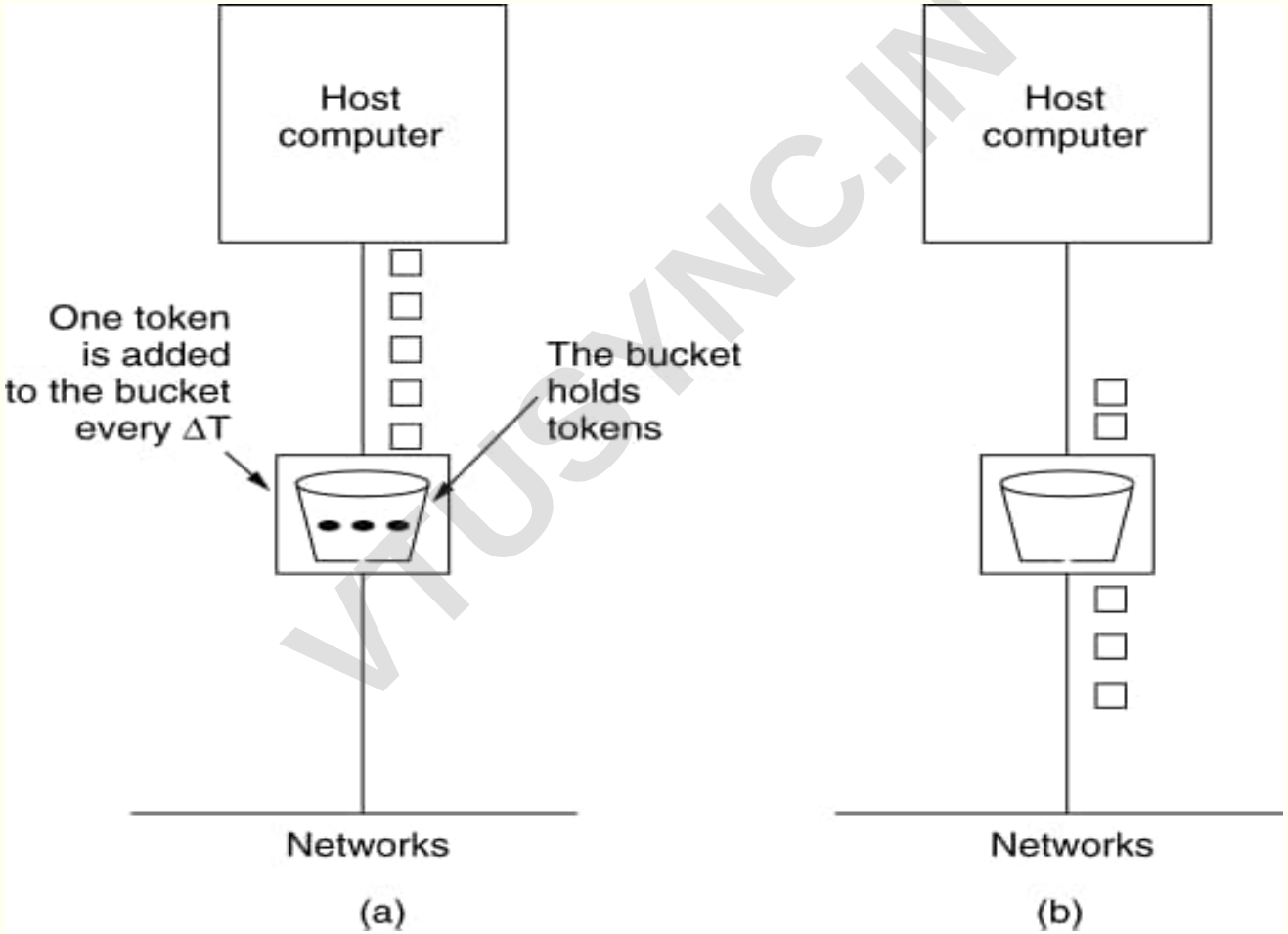
VTUSYNC.IN

Token Bucket algorithm

- The Token Bucket algorithm allows the output rate to vary, depending on the size of the burst.
- In the TB algorithm, the bucket holds tokens.
  - Each token entitles the host to transmit a fixed number of bytes.
  - To transmit a packet, the host must capture and destroy the required number of tokens (depending on the packet size).
- Tokens are generated by a clock at the rate of one token every Δt sec.
- Idle hosts can capture and save up tokens (up to the max. size of the bucket) in order to send larger bursts later.
  - When the bucket fills up, new tokens are lost.

VTUSYNC.IN

Token Bucket algorithm



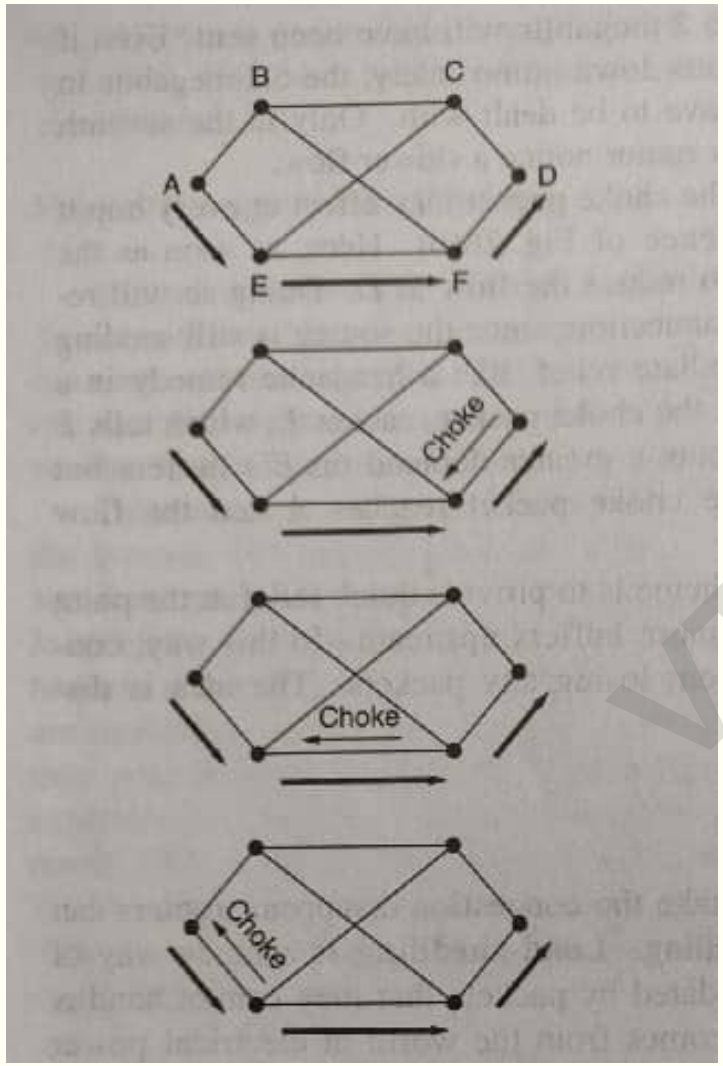
VTUSYNC.IN

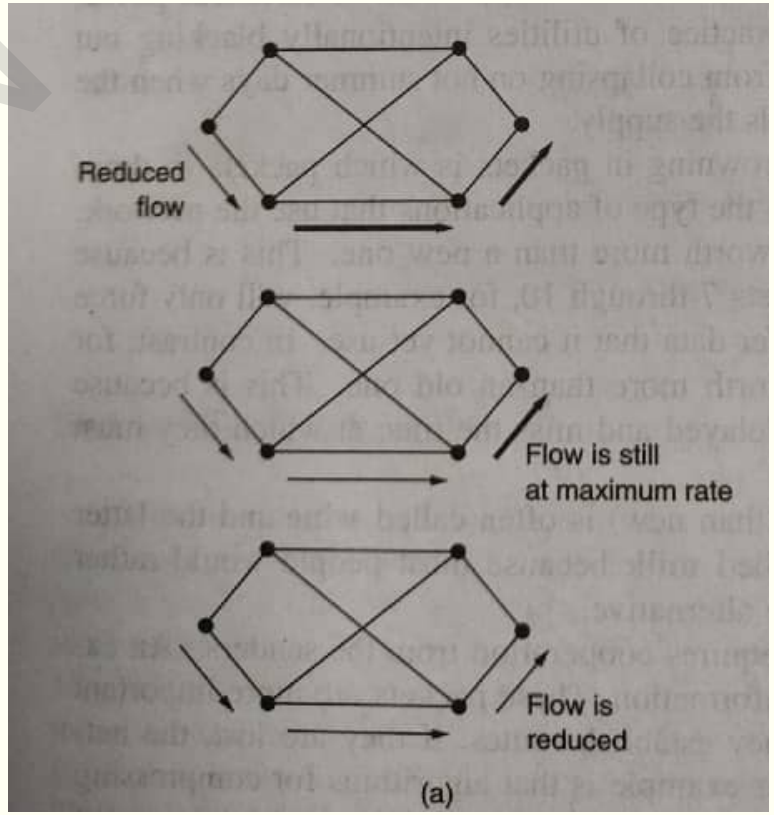
Choke Packet algorithm

- When a router is congested or about to get congested, it starts sending choke packets to the sources from where it received packets.
- A choke packet is a control packet generated at a congested node.
- The source, on receiving the choke packet must reduce its transmission rate by a certain percentage.
- This is a direct way of telling the sources to slow down temporarily.

VTUSYNC.IN

Choke Packet algorithm





D is the congested router, and A is the source of the congestion

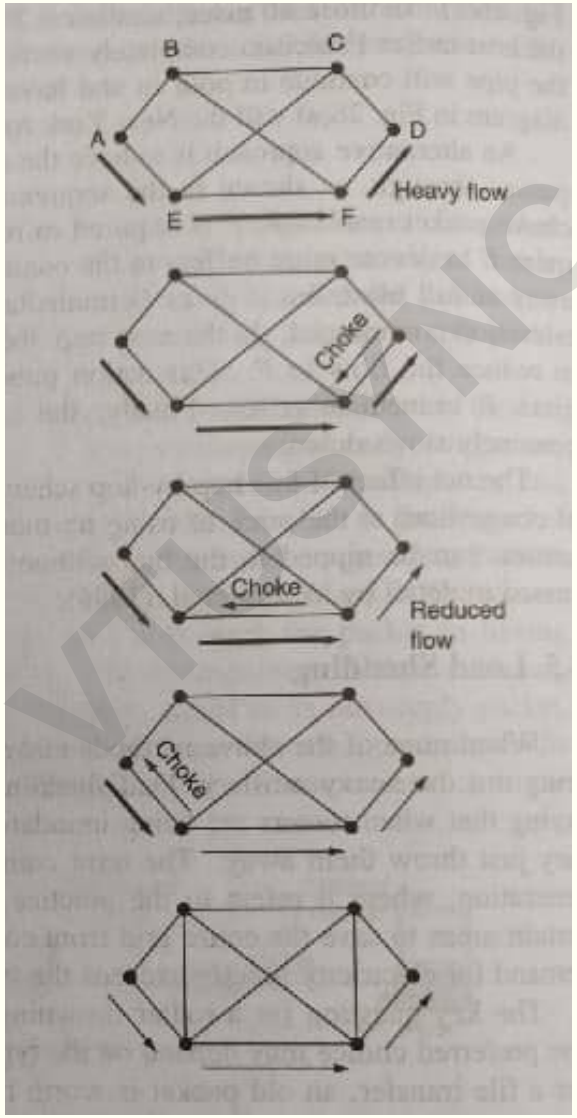
VTUSYNC.IN

Hop-by-hop Choke Packet algorithm

- Over long distances or at high speeds, choke packets are not very effective.
- A more efficient method is to send to choke packets hop-by-hop.
- This requires each hop to reduce its transmission even before the choke packet arrive at the source.

VTUSYNC.IN

Hop-by-hop Choke Packet algorithm



- This technique will spread the congestion backwards – To the routers along the way.
- It is better to have many routers ‘mildly’ congested than having one router congested ‘heavily’

VTUSYNC.IN

Explicit Congestion Notification (ECN)

- Instead of generating additional (choke) packets to warn of congestion, a router can tag any packet it forwards (by setting a bit in the packet's header) to signal that it is experiencing congestion.
- When the packet is delivered, the destination notes that there is congestion and informs the sender when it sends a Reply / Acknowledgement packet.
  - The sender can then throttle its transmissions, as before.
- This method is called Explicit Congestion Notification (ECN).

VTUSYNC.IN

Load shedding

- This method is the last resort, when none of the milder methods, described earlier, are able to control congestion.
- Load shedding just means that the congested router will throw away packets to reduce the load, when the buffer is filled up / or about to get filled up.
- The key question here is about which packets to discard; The answer depends on the type of application to which the packets belong.
  - For a File Transfer, an old packet is worth more than a new one. Dropping packet 6 and keeping packets 7 to 10 (for example) will only force the receiver to do more work to buffer the data that it cannot yet use.
  - In contrast, in real-time applications, a new packet is worth more than an old one, because packets become useless if they are delayed and miss the time when they must be played out.

VTUSYNC.IN

Load shedding – Random Early Detection (R.E.D.)

- Dealing with congestion when it first starts is more effective than dealing with it when it gets worse.
- Random Early Detection is a type of Load Shedding that discards packets before the buffer space is filled up.
- This idea stems from the fact that most hosts on the Internet do not get congestion signals in the form of ECN.
  - The only reliable indication of congestion that they get is Packet loss.
- To determine when to start discarding, routers maintain a running average of their queue length.

VTUSYNC.IN

Load shedding – Random Early Detection (R.E.D.)

- When the average queue length on some link exceeds a threshold, that particular link is said to be congested, and a small fraction of the packets (coming on that link) are dropped at random.
- Picking packets at random makes it more likely that the fastest senders will see more packet drops.
  - This is the best option since a router cannot tell which of the sources (sending packets on that link) is contributing more to the congestion.
- The affected sender will notice the loss, and its Transport layer protocol will slow down.
  - The (higher-than-normal) lost packets are thus delivering the same message as a choke packet, but implicitly.

VTUSYNC.IN

VTUSYNC.IN