19-NOV-21

CS4049D - ADVANCED COMPUTER NETWORKS

NAME: Bhukya Vasanth Kumar
ROLL: B180441CS (A- Batch)

Module 3 Notes

Transmission Control Protocol (TCP)

It is one of the main protocols of the Internet Protocol (IP) suite. The suite is also known as TCP/IP since it originated with initial implementation of IP.

Some key features:

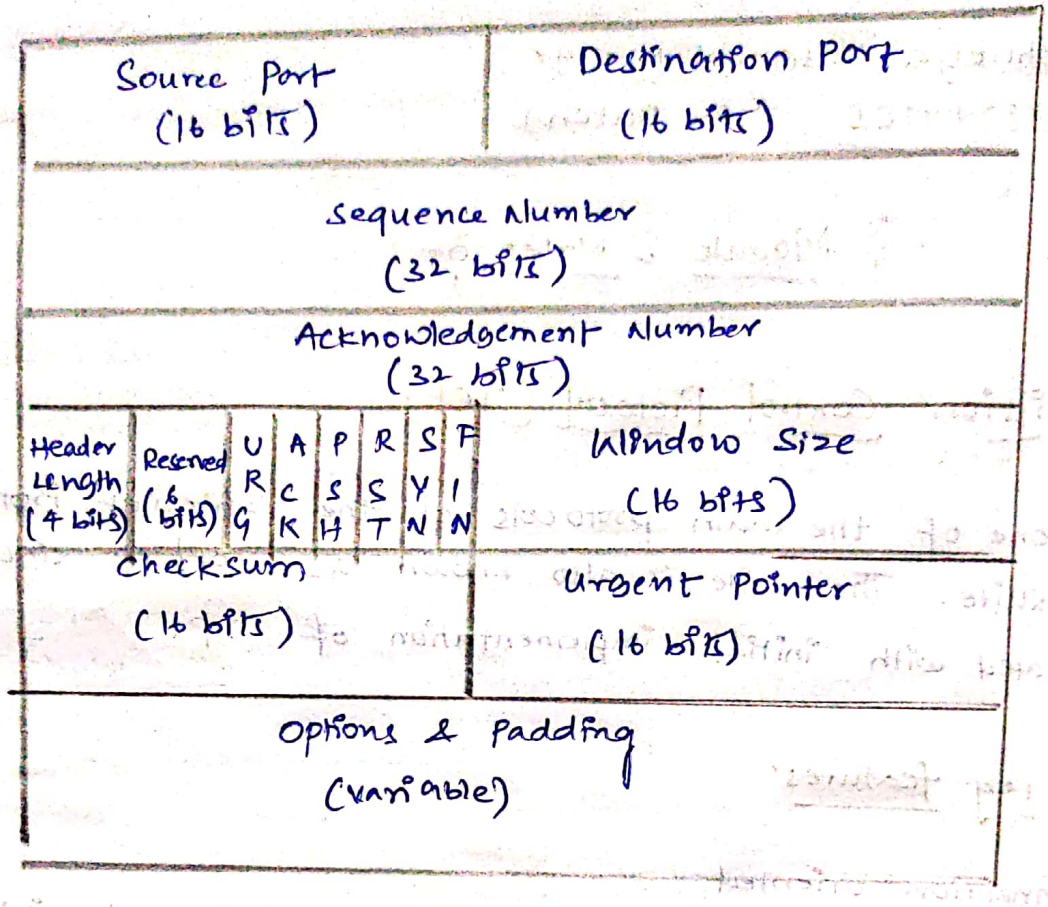
- Connection oriented
- Point-to-point communication between 2 end points.
- Reliable data transfer : packetization with sequence numbers, header, checksum, etc.
- Full duplex communication : bidirectional & symmetric.
- Reliable connection startup
- Graceful connection shutdown
- Reliable & adaptive retransmission

Other perks:

- End-to-End service: Application handle their own features at the end points and intermediary nodes only relay the payload.
- Reliable Transmission: If any packet loss occurs, the packets are retransmitted. The 3 way handshake ensures a connection before a payload is sent.

Scanned with CamScanner

Header Format



Source port: identifies source user process.

Destination port: identifies destination user process

Sequence number: byte number of the first byte in segment.

Acknowledgment number: byte number receiver expects next

Header length: indicates the length of TCP Header.

Control Flags: 1 bit control bit that control connection establishment, termination, flow control, etc.

URG: Urgent pointer is valid
ACK: Acknowledgment number is valid
PSH: Request for push
RST: Rest Connection
SYN: Synchronize sequence number
FIN: Terminate connection

Window size: Indicates window size for flow control
Checksum: holds checksum for error control

Scanned with CamScanner

Urgent pointer : If URG is set, then this 16-bit field is an offset from sequence number indicating last urgent byte.

options: optional field can be 0-320 bits

TCP Checksum

- It is a 16 bit ones complement of the ones complement sum of a pseudo header of information from the IP header, the TCP header and the data padded with zero octets at end to make a multiple of 2 octets.
- Filled with zeroes initially
- Tcp length is not transmitted but used in calculations.

Tcp Service Requests

- Unspecified passive open : Listen for connection requests from any user.
- Full passive open : Listen for connection requests from specified user
- Active open : Request connection
- Active open with data : Request connection & transmit data
- Send : send data
- Allocate : Issue incremental allocation for recieve data
- Close : close connection gracefully
- Abort : close connection abruptly
- Status : Report connection status

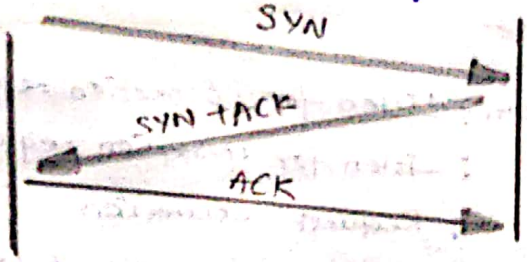
Scanned with CamScanner

Tcp Service Responses

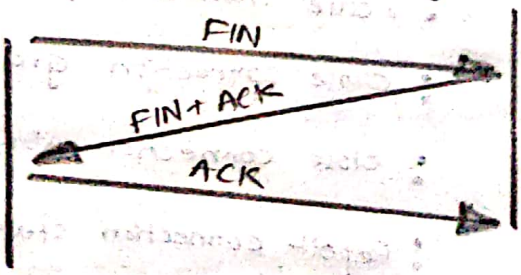
- Open ID : : Informs name assigned to pending request
- Open failure : Open request failed
- Open success : Open request succeeded
- Deliver : Reports arrival of data
- Closing : Remote TCP issued a close request
- Terminate : Connection terminated
- Status response : Reports connection status
- Error : Reports service request / internal error

Tcp Mechanisms

- connection Establishment
  3 way handshake with SYN flag set → request for connection



- connection Termination
  close with FIN flag set



Data Transfer

- Stream : Every byte numbered modulo 2^{32}
- Header contains sequence number of 1st byte
- Flow control : credit = no of bytes
- Data transmitted in Intervals determined by Tcp.
- Push : send now

Scanned with CamScanner

- urgent: send data in ordinary data stream with upg
- If TDPU not intended for this connection received → "reset" flag set in outgoing segment.

Implementation policies

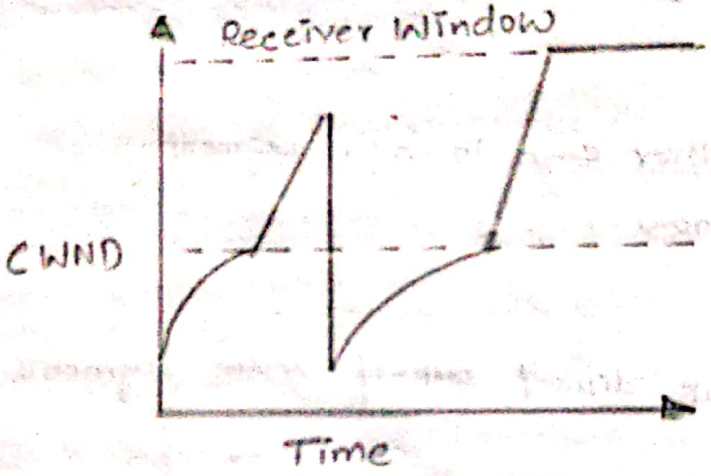
- Send policy
  - Too little: more overhead
  - Too large: delay
  - Push: send now
- Delivery policy
  - May store/ deliver each in-order segment
  - Push: send now
- Accept policy
  - May/ may not discard out-of-order segments.
- Retransmit policy
  - First only
  - Retransmit all
  - Retransmit individual (Maintain separate timer for each segment)
- Ack policy
  - Immediate (No piggybacking)
  - Cumulative (wait for outgoing data/ time out)

Slow start Flow Control

- Window → flow control avoids receiver overrun
- Needs congestion control → avoid network overrun
- Sender maintains 2 windows:
  - Credits from receiver
  - Congestion window from network
- Congestion window < receiver window

Scanned with CamScanner

- Start with congestion windows of 1 segment & do not disturb existing connections too much.
- Increase cwnd by 1 for every ack received.
- If packets lost → slow start threshold → cwnd/2
- set cwnd = 1
- Increment 1 per ack until ss threshold
- Increment 1/cwnd per ack afterwards



Fast Retransmit & Recovery

- If same packet acknowledgment 3 times →
  - → assume next packet lost
  - → retransmit of right away.
  - → retransmit only 1 packet
- useful when 1 packet lost, not multiple.

Selective ACK. (SACK)

- Initial Negotiation : sender to receiver "sack permitted"
- Selective ACK : variable length _ receiver to sender
- Left edge: 1st seq no in this block
- Right edge : Req no immediately after last seq number in block.

Scanned with CamScanner

- ACK field same : next byte receiver expects
- Missing segments received : ACK field advanced
- Receiver can send SACK only if sender has "SACK permitted" option in SYN segment of connection
- Data receiver can discard sacked data, sender must not until acknowledged.

Window Scaling Option

- Long fat pipe Networks (LFN) : satellite links
- Nud very large window sizes
- Normally, max window = 2^16 = 64K bytes
- Window scale option,

$$W = W * 2^{N(scale)}$$

i.e. $W = W * 2^{scale}$

- Max window = $2^{16} \times 2^{255}$
- option sent only in SYN & SYN+ACK segment.

TCP/IP Tools

- nslookup
- ping
- finger
- traceroute
- whois
- netstat
- ip/ipconfig/ifconfig
- dig/host
- tcpdump
- wget, etc.

Scanned with CamScanner

TCP Extensions for high speed networks

H-TCP (TCP for High speed & Long distance networks)

Why H-TCP ?

Future communication & computer networks
- → high speed & long distance connectivity
- → Require transport layer unhancement
- → Develop a systematic framework for modifying the basic TCP algorithm to render it suitable in a way of variety of network types.

The network new variant, H-TCP is:
- → suitable for deployment in high speed & long distance networks as well as connectional networks.
- → shown to be fair when deployed in homogenous network.
- → Friendly when competing with conventional TCP sources.
- → shown to rapidy respond to changes in available bandwidth & utilize link bandwidth efficiently.
- → shown to behave as a conventional TCP varient when deployed on conventional network types.

How does it work?

α_i, additive increase parameter for the scene is small for conventional networks (for backward compatibility) & large in high speed & long distance networks (for short duration congestion epoche even with large pipe sizes)

- → Modify the basic TCP paradigm by adjusting the rate α_i at which a source needs, inserts packets into network to reflect prevailing network conditions.

Scanned with CamScanner

→ Make α_i increase as function of time elapsed since last packet drop by i^{th} source.

High speed mode source increase function → $\alpha_i^H(\Delta_i)$

Low speed mode source increase function → α_i^L

$$\alpha_i = \begin{cases} \alpha_i^L & \Delta_i \leq \Delta^2 \\ \alpha_i^H(\Delta_i) & \Delta_i \geq \Delta^2 \end{cases}$$

Where,

Δ_i = Time elapsed since last congestion event by i^{th} src.

α_i^L = increase parameter for low speed regime.

$\alpha_i^H(\Delta_i)$ = increase function for high-speed regime.

Δ_2 = Threshold for switching from low to high speed regimes.

MulTCP (Schizophrenic TCP)

Originally proposed for differentiated services, to be used with a pricing scheme. TCP flow behaves like a collection of several virtual flows.

Algorithms Modified:

① Slow Start

For each ACK, triple the 'ewnd' for first 'k' RTTs, double after that.

- Congestion window at the end of 'k' RTTs = 3^k.
- Sum of congestion window of N flows = $N2^k$.
- Choose k such that, $3^k = N2^k$.

Scanned with CamScanner

(*) Linear Increase

Increase the cwnd by N/ cwnd for each incoming ACK.

(*) Multiplicative Decrease

For each loss, reduce cwnd for only one virtual flow,
i.e. reduce cwnd by $\left(\frac{1}{N}\right) * 0.5 *$ cwnd.

Downsides:

- Cannot modify/emulate timeouts
- N set by user or not controlled by end-to-end congestion situations and conditions.
- Relies on policing and pricing to prevent congestion collapse.
- More bursty compared to traditional TCP flavours.
- Suffers performance loss during timeouts compared to N individual flows.
- More aggressive than should be for SACK, when N < 5.

High Speed TCP (HSTCP)

How?

It modifies the response function of TCP, so that ridiculously low drop rates are not required to sustain high throughput.

Increase function : $w = w + a(w)/w$

Decrease function : $w = w(1 - b(w))$

For cwnd < 38, $a(w) = 1$ & $b(w) = 1/2$ (same as standard TCP)

For larger cwnd, $a(w)$ & $b(w)$ are long functions based on several measures $a(w)$ & $b(w)$ are precomputed and stored in lookup tables.

Observations:

HTCP emulates roughly N parallel TCP connections.

Scanned with CamScanner

Scalable TCP (STCP)

Modifications to TCP Algorithms:

- For cwnd ≤ lwnd, we traditional window update algorithm.
- For cwnd > lwnd,
  - Increase function = for each ack, cwnd = cwnd + 0.01
  - Decrease function = for each loss, cwnd = cwnd − [0.125 * cwnd]
- lwnd chosen to be 1b corresponding to drop probability $5.95 * 10^{-3}$
- This response function efficiently is MIMD.
- Fixed time for doubling sending rate (TD RTTs).

Comparison with Traditional TCP:

Traditional TCP scaling properties		Scalable TCP scaling properties
For each ack →	cwnd = cwnd+1 / cwnd	cwnd = cwnd + a
For each loss →	cwnd = cwnd − 0.5*cwnd	cwnd = cwnd − [b * cwnd]

Why a=0.01 & b=0.125 ?

- Impact on legacy traffic
- bandwidth allocation properties
- flow rate variance
- Convergence properties
- control theoretic stability

Pros:

- A special case of HSTCP response function
  → must be easy to implement, model mathematically & less ad-hoc.
- Fixed doubling time for sending rate (scalable)
  → Independent of link capacity

Scanned with CamScanner

Cons:

- MIMD inherently more aggressive than AIMD
- Not enough experimental results to show STCP will not starve standard TCP.

Fast AQM scalable TCP (FAST)

- Built on the TCP-Vegas core.
- Uses equation based control with queuing.
- Uses multiplicative decrease with packet loss.
- Based on flow-level dynamics rather than packet level dynamics.

Main Algorithm

- Estimate target rate
- Estimate difference between current rate & Target rate
- If every high, increase sending rate aggresively
- If not, increase sending rate very smoothly.
- Avoids oscillations, keep router queues stable.

Window Control Algorithm

For each RTT,

$$W = w(\text{base RTT}/\text{RTT}) + \alpha$$

where,

RTT = exponential weighted average of RTT

base RTT = minimum instataneous RTT

α = constant determining fairness / aggresiveness & convergence rate

α is computed based on link bandwidth, link RTT etc.
It is stored in a look up table.

Pros & Cons

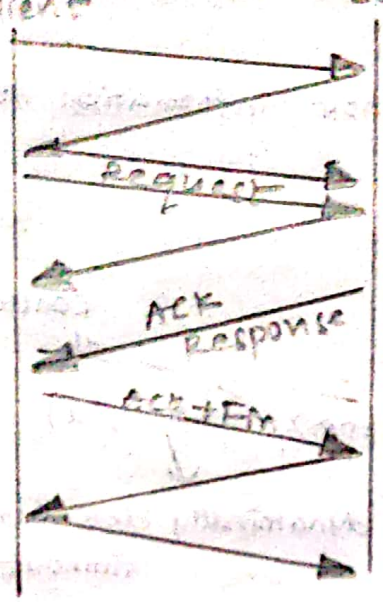
(+) Theoretically fair, not enough experimental results.

(*) Delay for congestion indication not fully understood or widely deployed yet.

Scanned with CamScanner

T/TCP : Transaction Oriented TCP

Avoids the 3 way handshake to prevent long delays for transaction-oriented client server applications.



Why T/TCP?

- Standard TCP uses atleast 10 segments for a single segment request - response sequence.
  - high relative overhead
  - Extra transaction latency for client on CPU & network speed increase.
- The 3 way handshake precludes high performance transaction processing
- A high rate of short connections like transactions leads to excessive consumption of kernel memory by connection control blocks in TIME WAIT state.

How does it work?

Transaction oriented TCP

- supresses the 3 way handshake
- shortens TIME WAIT state.

Scanned with CamScanner

Bypassing the 3 way handshake.

TCP Accelerated Open (TAO) Mechanism:

- Validates initial SYN segments
- Enforces at most one semantics without a 3-way handshake.

Basis: TCP uses cached per host information to immediately validate new SYNs.

Shortening TIME_WAIT state.

- Set TIME-WAIT delay to
  - TIME_WAIT_Delay = max(K*RTD, u)
    - K: constant
    - RTD: dynamically determined retransmission timeout
- If RTO is very small, short connections are possible.
  - TIME-WAIT delay reduces to u.
- To accelerate close sequence, u has to reduce below MSL set by IP layer.
  - Done using a monotonic number sequence, x.
- To further reduce TIME_WAIT_Delay
  - explicitly truncate TIME-WATE state
  - remove TIME-WAIT_Delay entirely.

TCP Performance Issues Over Wireless Links

TCP Flow Control

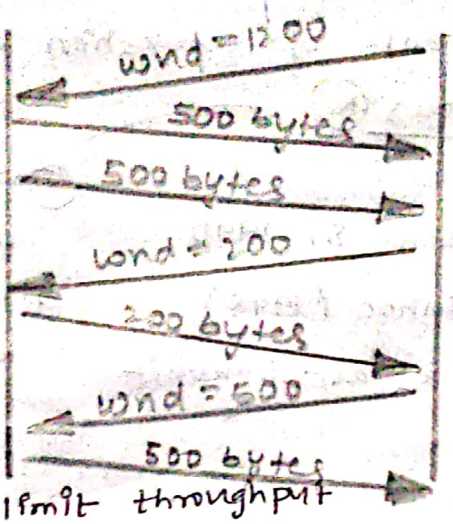
- prevents buffer overflow at receiver
- useful for fast sender transmitting to slow receiver.

How does it work?

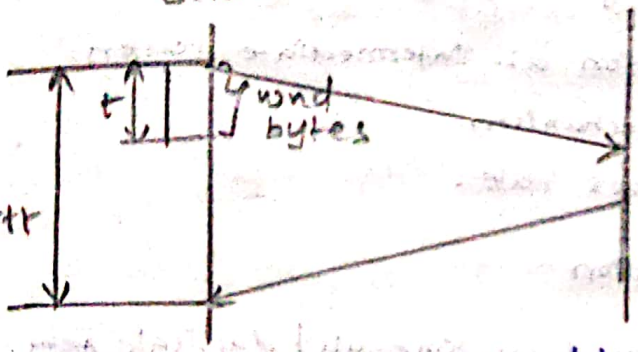
- Receiver advertises a window (wnd) in ack returned to sender.

Scanned with CamScanner

- Sender cannot send more than wnd unacknowledged bytes to receiver.



Flow control can limit throughput



let rtt be round trip time.

$t = wnd/b$ = time to transmit a full window of data,

where
b = link bandwidth.

For a link with high delay - bandwidth product (rtt x b), flow control window can limit connection throughput

$t \leq rtt$, throughput = wnd/rtt

TCP Congestion Avoidance

- Reduce congestion inside network: Requires cooperation of multiple senders, must rely on indirect measures of congestion
- Implemented at sender
- Introduce a congestion window (cwnd) in addition to flow control window (cd). size of this window needs to be managed.

Slow Start

- Aggressively grow congestion window until congestion is detected.

Scanned with CamScanner

Loss Detection & Retransmission

- Fast retransmission & recovery
- less severe adjustment of congestion window size

Options

- At end of options if, max. segment
- Later RFCs, are defined.

TCP Reno

- Developed by Van Jacobsen in 1990
- Improvement to TCP Tahoe (1998)
- Added Fast recovery & fast transmit

Congestion Avoidance

- Uses packet loss as an indicator of network congestion.
  - → Most packet loss due to congestion at intermediate routers.
    - → Has no congestion control mechanism
    - → packet loss due to bit errors rate.
- Reactive with respect to congestion
  - → Responds to packet losses indicated by timeout / duplicate ACKs.
- Packet loss
  - → Reduce congestion window size
  - → Due to timeout : cwnd = 1 and ENTER SLOW START
  - → Due to duplicate ACKs: $cwnd = \frac{cwnd}{2} + 3 * segment\_size$.
- Congestion window size increased when data succesfully acknowledged.

- Slow start:
  - Active if $cwnd \leq ssthresh$ (threshold)
  - During slow start, congestion window increased by segment size for every ACK recieved, it opens window exponentially

- Congestion Avoidance:
  - $cwnd = cwnd + (1/cwnd) * segment\_size/8$ for every ACK received
  - Additive growth in window size, about 1 segment every rtt.

Scanned with CamScanner

Slow start Mechanism

- Detects, avoids congestion as connection begins / after a timeout.
- RENO doubles congestion window every round trip if no loss occurs
- Aggressively reduce rate when invoked

Loss Detection

– course grained timeout indicates packet loss
  - sender starts timer when TCP segment sent
  - Timeout if Ack not received before timeout
  - Retransmission
  - slow start invoked

– 3 duplicate Acks indicate packet loss
  - Receiver to send Ack out of older segment
  - Sender to assume loss when 3 duplicate Acks received
  - Fast transmission & recovery

Pros:

- Simple bandwidth estimation scheme
- Aggresive congestion avoidance mechanism
  ↓
  Ensures bandwidth when connected to TCP Vegas.
- More widely deployed

Cons:

- periodic oscillation in window size
- oscillation in RTTs
- Inefficient bandwidth utilization
- Many retransmissions of same packets after a packet - dropped
- Biased against connections with longer delays
- Unfair share of network capacity

Scanned with CamScanner

TCP Vegas

Developed by Brakmo & Peterson – 1995

New congestion avoidance algorithm

Congestion Avoidance

Set congestion window size → difference b/w expected & actual data rates

→ To control no. of outstanding bytes in queues in network

$$diff = (cwnd/rtt * - cwnd/rtt) * rtt *$$

where,

cwnd = current congestion window size

rtt = minimum rtt

rtt * = actual rtt

diff = estimate backlog in queue

α = low threshold for diff ($diff > \alpha$)

β = high threshold for diff ($diff < \beta$)

* Vegas keeps atleast α bytes but $< \beta$ bytes in queue.
  - If $diff < \alpha$, cwnd +1
  - If $diff > \beta$, cwnd −1
  - Else ($\alpha \leq diff \leq \beta$), cwnd not changed.

* Vegas provides proactive response to congestion. Congestion window changed gradually as observed backlog changes.

Slow start Mechanism

Doubles congestion window every other round trip if no loss occurs.

Scanned with CamScanner

Loss Detection

- Course grain timeout mechanism
- Fine grain timeout mechanism
  - Duplicate Ack received & RTT of 1st unacknowledged segment > fine grain timeout value
  - Non duplicate Ack received after retransmission & RTT of segment > fine grain time out value.

Pros:

- Fair bandwidth estimation scheme
- smooth sending rate, efficient link utilization
- Detect losses faster than TCP Reno & recovers more efficiently from multiple drop.

Cons:

- Cannot compete with more aggresive Reno connections
- May not be stable if buffers are small

TCP Reno Vs TCP Vegas

In a homogenous environment, Vegas outperforms Reno.

40% - 70% better throughput
20% - 50% of losses

Factors:

- slow start & congestion avoidance have greatest influence on throughput
- Congestion detection mechanism has only minor effect
- May exhibit problems with fairness among competing connections and conditions.

Scanned with CamScanner

TCP problems in Wireless Networks

- Packet loss due to
  - Bit errors due to wireless channel impairements
  - Handoffs due to mobility
  - Possibly congestion
- TCP assumes packet loss due to
  - Congestion in network
  - Pack reordering
- TCP congestion avoidance can be triggered by packet loss
  - Do not respond well to packet loss due to bit errors / handoffs
  - performance is based on applications, can suffer.
- Bursts of error due to low signal strength
  - More than 1 packet loss in TCP
  - More likely to be detected as timeout → enter SLOW-START
- Delay often very high
  - RTT can be very long & variable
  - Tcp's timeout mechanism may not work well
  - Made worse by link level retransmission
- Links may be asymmetric.

Solutions to these problems

Link-layer approaches

- Hide losses due to congestion from sender
- make link appear more reliable
- ARQ, forward error correction (FEC) code
- Hybrid ARQ & FEC

Scanned with CamScanner

Pros

- No change to existing sender behaviour
- Matches layered protocol model

Cons

- Interaction with TCP,
  Ex: fast transmission with TCP can be triggered by delays due to link layer timeout & retransmission.
- Negative interactions with TCP reduced by making link-level protocol TCP aware.
  Ex: Snoop TCP

Advantages

- Attempts to retransmit & surpress duplicate ack.
- State is soft
- Hand off simplified

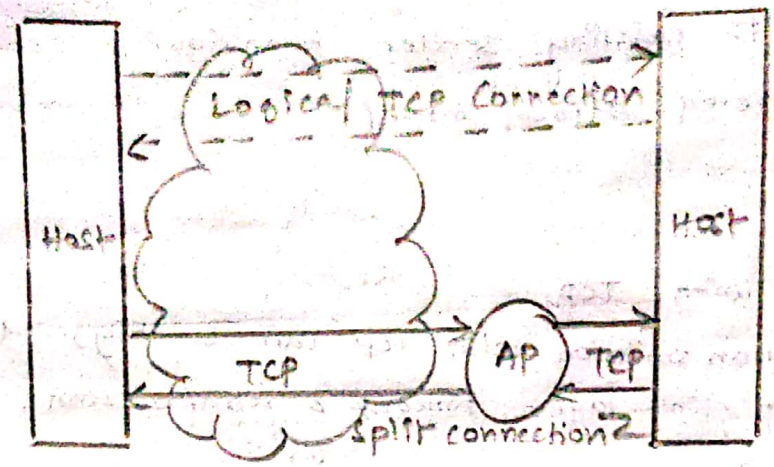
Disadvantages

- May not completely shield TCP from effects by mobility & wireless links.

Split Connection Approaches

- Hide wireless link entirely by terminating TCP connection prios to wireless link.
- Hides at base station / access point
- Uses special protocol / regular TCP over wireless link
  Ex: Indirect TCP
- Problems
  → Extra protocol overhead
  → violates end-to-end semantics of TCP
  → complicates head off
  → state info. at AP / base where protocol is split

Scanned with CamScanner



End-to-End approaches

- Make TCP sender aware of some issues not due to congestion
- Avoid congestion control when not needed.
- Use selective acknowledgement (SACKs) for "fine-grained" error recovering

  Ex: SACK RFC, SMART

- Use explicit loss notification (ECN) to distinguish b/w congestion & other losses.

Advantages

- Maintains end-to-end semantics of TCP
- No extra overhead at base stations for protocol processing / hand off.

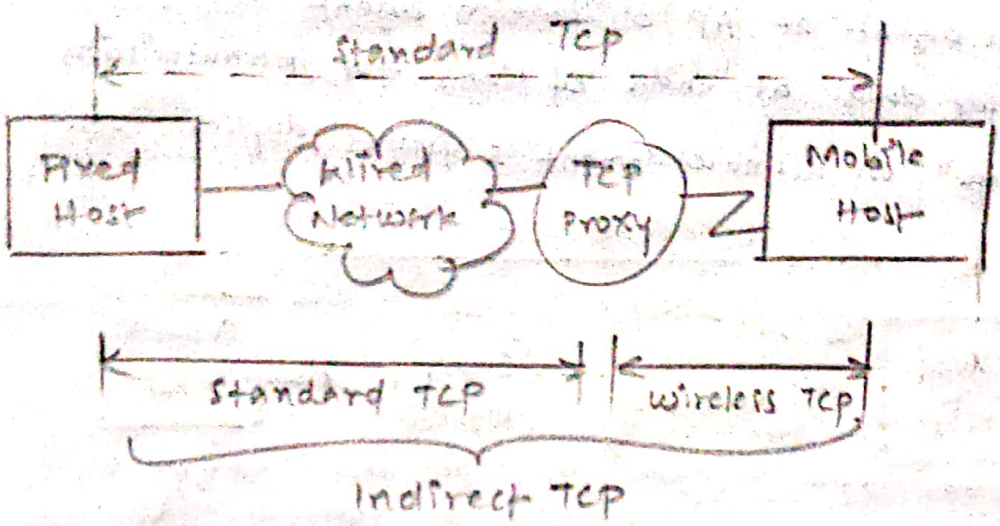
Disadvantages

- Requires modified TCP
- May not operate efficiently

  Ex: packet reordering Vs packet loss.

Scanned with CamScanner

Indirect TCP



Hand off

- Access point/router can act as mobile IP foreign agent & TCP proxy for indirected TCP (I-TCP)
- Mobile host moves to different foreign agent (handoff needed for mobile IP)
- Mobile host moves to different proxy. (hand full/hand off of full TCP headed for I-TCP)

Advantages

- No changes to TCP at hosts in fixed networks
- Errors from wireless links corrected at TCP proxy
- optimisation possible over wireless link

Disadvantages

- loss of TCP end-to-end semantics
- handoff overhead can be significant
- overhead at proxy for per-packet processing
- TCP proxy must be trusted

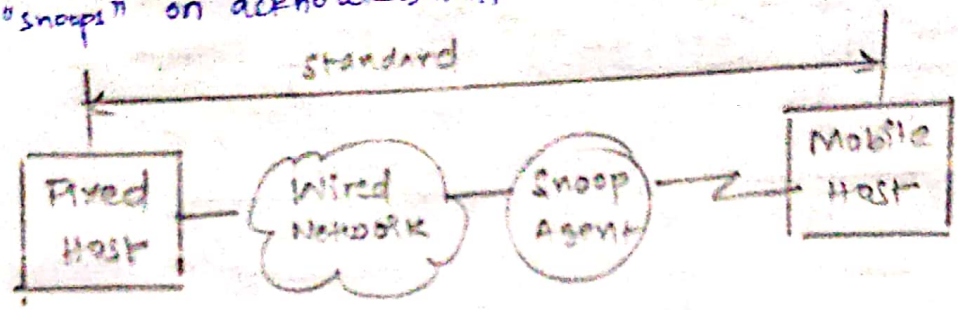
Wireless Transport

- Uses TCP as wireless transport protocol
- Timeouts at wireless sender, may stall original sender
- Selective acknowledgement protocols provide better performances

Scanned with CamScanner

Snoop TCP

- provide reliable Link layer, ie. TCP-aware
- Snoop agent at AP of foreign agent
- Buffer data at ends of links for retransmission
- "snoops" on acknowledgment & filters duplicate acknowledgment



Operation

- Snoop agent monitors & buffers data from fixed network to mobile host.
- Snoop agent monitors Acks from mobile host.
  - → can discard buffer data when acknowledged
  - → can retransmit data (delayed/ duplicate) ACK
- short timeout → fast retransmission
- Snoop Agent discards duplicate Acks from mobile host
- Snoop Agent discards duplicate data already sent by agent & acknowledged
- Snoop Agent cannot generate ACKs sent back to fixed host
  - ↓
  - end-to-end TCP semantics preserved.

Reverse Direction

- Snoop monitors traffic from mobile host back to fixed host and detect missing segments.
- A negative ACK sent immediately to mobile host.
- Mobile host can retransmit missing segment, in time to avoid TCP time out at fixed host.

Scanned with CamScanner

Advantages:

- Preserves end-to-end semantics
- No changes in TCP for fixed hosts
- No changes in TCP possible for mobile hosts, reserve direction traffic can benefit from days.
- No need for hand off
- Automatic fall back to standard TCP

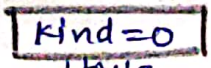
Disadvantages

- Does not fully isolate wireless link errors from fixed network
- Mobile host modified to handle NACKs for reverse traffic
- Cannot snoop encrypted datagrams
- Retransmission of data from agent not authenticated.

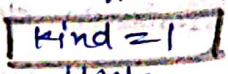
TCP Options

The only options defined in original TCP specification are the end of options list, no operation & maximum segment size option. Later RFCs 793 & 1323 defined new options.

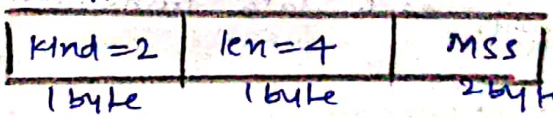
→ End of option list



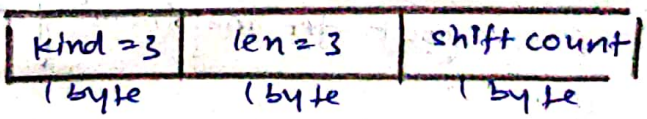
→ No operation



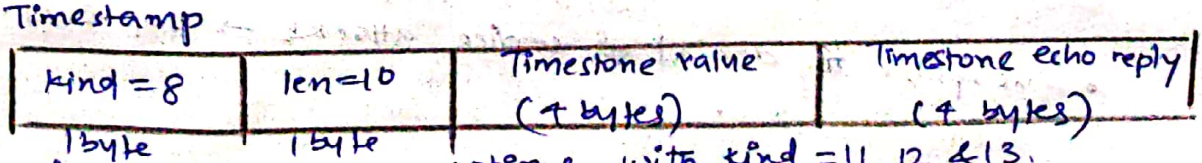
→ Max. segment size



→ Window scale Factor



→ Timestamp



* TCP proposal specifies 3 new options with kind = 11, 12 & 13.
* They are connection count (cc), NEW, ECHO & cc.
* Four other options of kind = 4, 5, 6 & 7. They are called 'echo-options'. They are replaced by Timestamp.

Scanned with CamScanner

Stream Control Transmission Protocol (SCTP)

- IETF approved as a proposed standard in Oct 2020.
- A general purpose transport layer protocol
- Offers point-to-point, connection-oriented, reliable delivery transport service.
- Powerful congestion control & packet loss recovery
- Any app. running over TCP can be ported to run over SCTP
- Support for multihoming and partial ordering

enables SCTP host to establish "session" with another SCTP host over multiple interfaces, identified by separate IP addresses.

lets SCTP provide in order delivery of one or more related sequences of messages following between 2 hosts

- Reliable delivery of fast processing of multiple, unrelated data streams.

TCP Issues

- Strict order of transmission delivery since, service.
- Treats each data transmission as an unstructured sequence of bytes
- No support for multihoming → sockets-based application programming interface
- Susceptible to denial of service attacks → TCP SYN storms.

SCTP features

Association → protocol state installed on 2 peer SCTP hosts exchange messages

- Message boundary preservation
- No head-of-line blocking

Scanned with CamScanner

- Multiple delivery nodes
- Multihoming support
- TCP congestion control
- selective acknowledgments
- User data fragmentation
- Heartbeat keep alive mechanism
- DOS protection

Multihoming

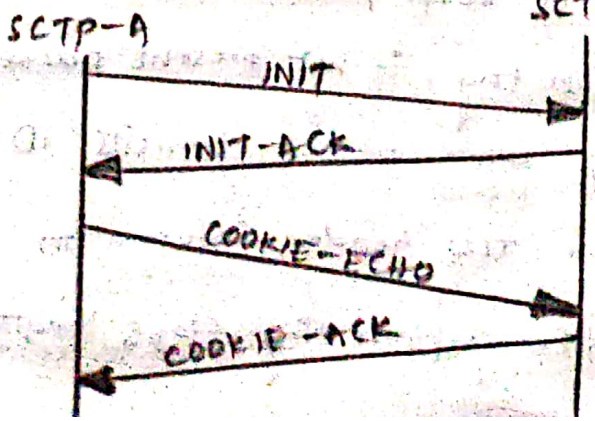
- Offers network resilience to find failed interfaces on the host and fast recovery during network failures
- Less affected by network reconvergences → Lost packets retransmitted to alternate address
- Stress Reduction: Reduce stress on internet routing system
- Topology diversity: diversity in routing paths dictates level of fault tolerance
- Delivery options: separate reliable & ordered delivery into 2 Independent functions.

SCTP Initiation

SCTP Association defined as

[a set of IP addresses at A] + [port A] + [a set of IP addresses at Z] + [port Z]

Before data exchange, exchange communications state using a 4-way handshake. Eliminates exposure to TCP SYN flooding attacks



Scanned with CamScanner

1. INIT (Initial Contact message)

2. INIT-ACK → with state cookie

↓

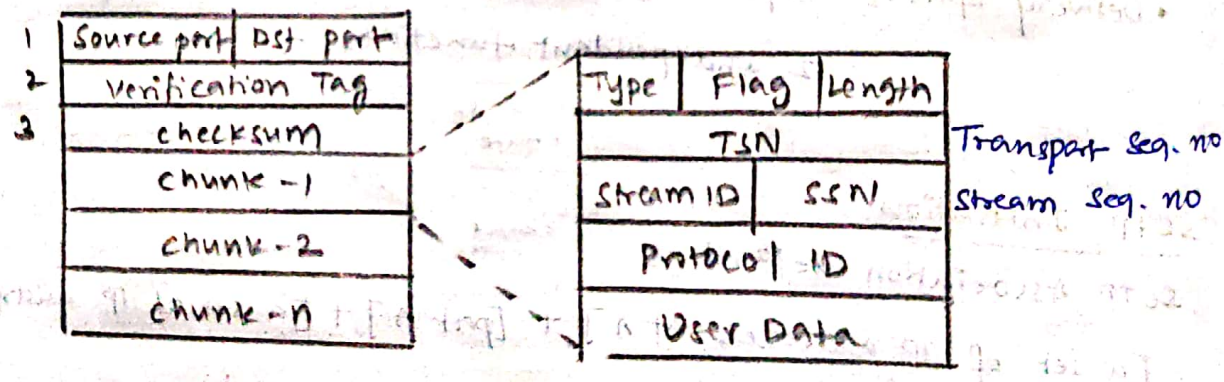
→ Parameters in both:

- list of all IP addresses part of association
- Initial transport seq. no for reliable data transfer
- Initiation tag for every in bound SCTP packet
- No. of outbound stream each side is requesting
- No of inbound streams each side can support

3. COOKIE-ECHO → state cookie, might have user DATA messages

4. COOKIE-ACK →
- sent after receiver fully reconstructs its state & acknowledges.
- setup is complete
- can also bundle user DATA messages.

SCTP Data Transfer



SCTP packet Format (common header + one/more variable length data chunks)

1→ Identify recipient of SCTP packet

2→ value of initiation tag, if not present packet dropped

3→ Assure integrity while packet transits ID networks.

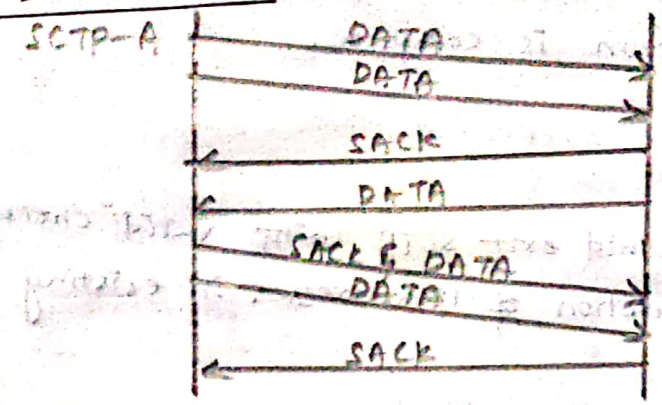
Every chunk includes TLV header information

- chunk type / delivery processing flags, length field.

Scanned with CamScanner

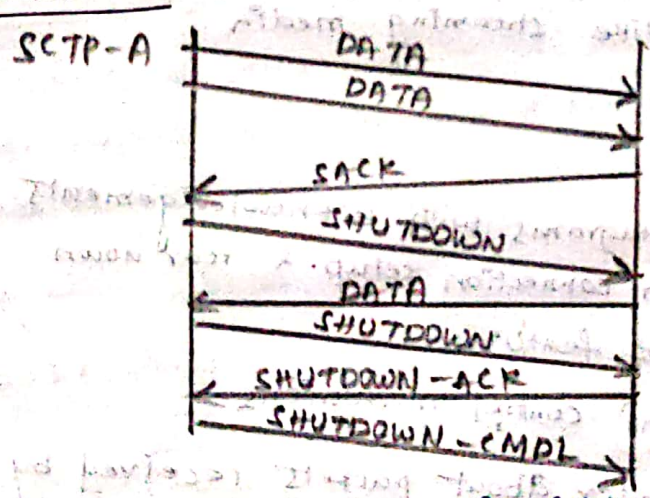
TSN → for per-association reliability
SSN → for per-stream ordering

SCTP Data Exchange



SCTP host sends SACKs in response to every SCTP packet carrying data chunk. The SACK fully describes receiverls data.

SCTP shutdown



No "half-closed" state in SCTP

A wishes to shutdown & terminate association with 2.

1. SCTP enters SHUTDOWN_PENDING STATE
   ↳ accept no data from app, send now queued data to 2.

2. After acknowledging all queued data, A sends SHUTDOWN chunk & enters SHUTDOWN-SENT state.

3. upon receiving SHUTDOWN chunk, 2 notifies upper layer
   → stops accepting new data
   → enters SHUTDOWN-RECEIVED state.

Scanned with CamScanner

4. Z transmits any remaining data to A
5. SHUTDOWN chunks follow to inform Z → data arrived + assoc. shutting down
6. All queued data on Z acknowledged → A sends SHUTDOWN-ACK chunk, then, SHUTDOWN-COMPLETE chunk
7. Association shutdown is complete.

Future Issues

- Corrupted packet could exit SCTP with valid checksum
- Dynamic addition / deletion of IP addresses in existing association
- PPRT support

Datagram Congestion Control Protocol (DCCP)

- Congestion controlled, unreliable flow of datagrams
- Suitable for apps like streaming media

DCCP Features

- Unreliable flow of datagrams with acknowledgements
- Reliable handshake for connection setup & tear down
- Reliable negotiation of features
- TCP friendly congestion control mechanisms
- options to inform sender about packets received by receiver & their status.
- Congestion control incorporating explicit congestion Notification (ECN)
- Allow server to avoid holding state for unacknowledged / already finished connection.
- Path MTU discovery.

Important Differences from TCP

- Packet stream protocol, not byte stream
- Unreliable → no transmission of datagram

Scanned with CamScanner

- Packet seq. nos, not for bytes
- space for options (upto 1020 bytes)
- Feature negotiation
- choice of congestion control (2 end-points)
- Different acknowledgment formats
- No receive window → congestion control, not flow control protocol
- Distinguishing different kinds of loss
- Definition of acknowledgement (which options have been processed)

- Integrated support for mobility
- No simulations open

What is a Half-connection?

- A subset of connection
- Data packets sent in one direction + corresponding ack. sent in other direction
- HC sender : sends data
- HC Receiver : sends ACKs.
- Every connection (half) managed by congestion control of mechanism.
- specified by CCID ~~congestion~~ congestion control identify.

Typical DCCP Connection

1. DCCP Request : client → server, service, features, CCID etc.
2. DCCP Response : server → client, indicates willingness to communicate
3. DCCP-ACK : client → server, acknowledges the response packet

Scanned with CamScanner

4. 0/ more DCCP-Ack : To finalise feature negotiation

5. Exchange of DCCP-Data, DCCP-Ack and/or DCCP-Data Ack.

6. DCCP-Close Req : server sends request to close.

7. DCCP-close : client acknowledges the close

8. DCCP-Reset : server sends reason set to "closed" &
(server) cleans connection state.

9. DCCP-Reset : Client receives packet & holds state for any
(client) remaining packets to clear.

Scanned with CamScanner