$$1.5d + 0.9d = 1$$

$$\therefore d = \cdots$$

CSE - 321

Transport layer

→ Transport layer -এর address port.
different process different port-এ চলবে।

UDP

→ একটা Packet send করবে, ওটা পৌঁছুল কিনা ওটা নিয়ে চিন্তা করবে না।

→ UDP checksum optional. হিসাব না করলে সব bit 0 দিবে।

→ Command এর সাথে parameter গুলো pack করাকে marshaling বলে। Opposite - unmarshaling.

Remote Procedure call:

Problems → দুইটা different machine pointer ফিজিক্যালি পাস করানো যায় না, int -এর length, string এর null char এগুলো machine to machine vary করে।

UDP use হয় → দুই case দুইটা → Parameter, UDP packet এর চেয়ে বড় হলে (কারণ fragment হলে, এক part গেলে, বাকিটা গেলো না হতে পারে।) এমন

Idempotent operation হলে। (যেগুলো same method different time এ different output হয়)

এজন্য এখানে TCP use হয়।

RTP

RTP অনেকগুলো stream কে merge করে multimedia এর।

TCP

TCP header → 20–60 bytes

Acknowledge number এর দুই নিয়মে

Convention → Upto which number I have seen
→ which seq. number is next to be sent.

Unexpected scenario এ হলে → reset

Willingly connection close → finish

Window → Ack ছাড়া at once যত অনেকগুলো packet sent করবে,

checksum calculate করার সময়, শুধু header + একটা extra part (checksum রাখার) 16 bit করে নিয়ে XOR করে, Receiver checksum কে XOR করে 0 পাওয়া যায়, না হলে error.

TCP তে segment size header-এ থাকে না বলে connection establish-এর সময় ঠিক করে নেয়া হয়।

Nagel's solution. 1 byte send করে অপেক্ষা করে, ACK আসলে আবার 1 byte পাঠায়। → slow sender

slow receiver ← min (Minimum Segment, Buffer/2) সর্বনিম্ন advertise করে। → clarke's solution

7/8/23

CSE - 321

TCP:

- Congesⁿ control → written as "Congestion control"
- Flow control
- Error control
    - Corruption → checksum (may be supported by UDP)
    - Loss → Retransmission
    - Duplication → Seq #
    - Reordering → Seq #

Solⁿ to SYN Flood attack

- SYN cookie: Don't create state, rather enter the connⁿ-request related info in the server.

↑ First এ resource allocate না করে, info গুলোতে cookie এ রাখে, handshake এর 3rd ধাপে resource allocate করে।

(*) Advertised window = Max Rcvr Buffer − ((NBE − 1) − LBR)

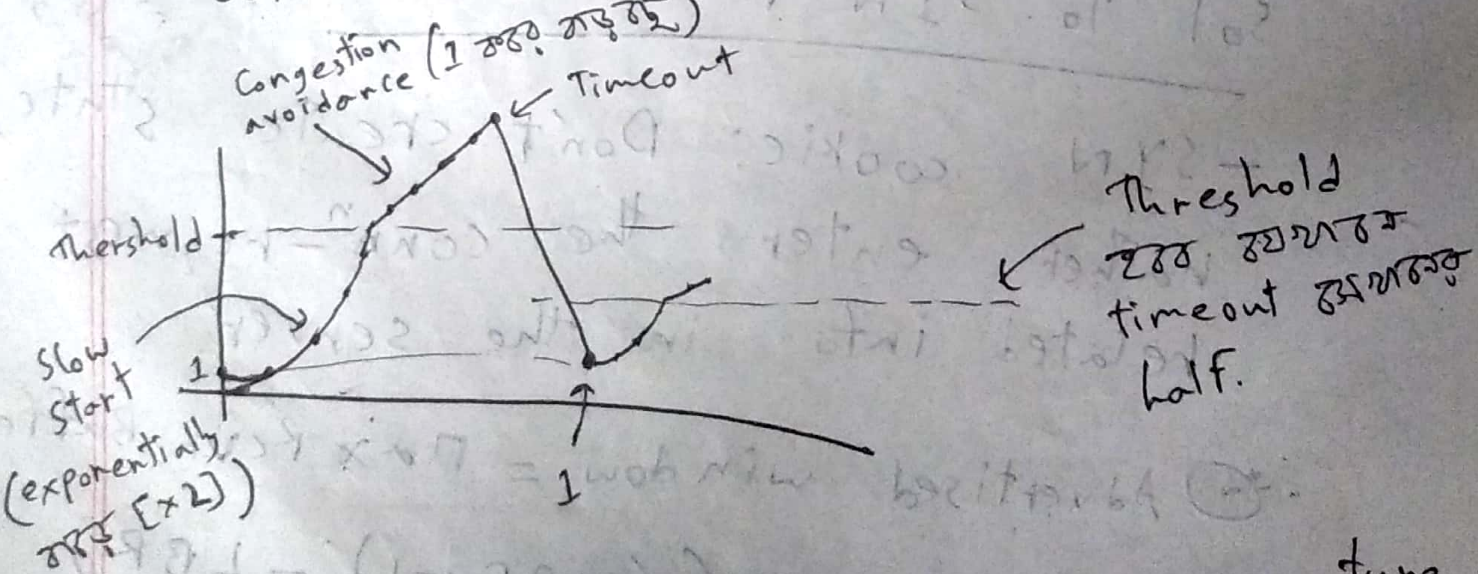
(*) Effective window = Advertised window − (LBS − LBA)

* Size of TCP socket buffer (expected) = BW * delay

[To ensure the pipe can be full]

→ Flow control এর জন্য receiver এর window. → Advertise window

→ Congestⁿ control এর জন্য sender window maintain করে, → Congestion window



→ Timeout এর সময় কম হলে premature expiration ঘটবে, বেশি হলে loss বুঝতে অনেক বেশি delay wait করা হবে।

From Patterson:

RFC → Request for comment [Network standards এর Documentation]

- UDP তে checksum করা যায় optionally, মানে কিছু corruption ধরা যায়, এই- এর type-এর reliability UDP তে আছে।
- Sender ও connection establish এর সময় random seq # দিয়ে start করবে।

(Cause, same application break করলে আবার আগের instance এর seq # এর সাথে overlap করতে পারে, মানে confusion সৃষ্টি করতে পারে, তাই randomly start করে।)

- 3-way handshake 100% guarantee করে না, but এটা যথাসম্ভব ভাল করে।
- Duplicate frame ধরার জন্য seq #. Duplicate ACK ধরার জন্য ACK #.

ACK-এর number টা হল upto what packet receive করা হয়েছে তার পরের টা.

- Sliding window → min (Congestion window, advertise window)

- Out of order receive করলে তা buffer → এ রাখে দিবে, e.g.

1.2.3.5 → 4 না আসা পর্যন্ত 5 buffer-এ রাখে দিবে, 3 পর্যন্ত ACK করবে,

New RTT = α(old RTT) + (1-α)(new sample)

→ Time out হলে RTO কে আরো double করা হয়.

→ Retransmission হলে ওইটা RTT এর sample হিসেবে নিব না, কারণ ACK first নাকি retry-এর ACK এটা বুঝার উপায় নেই.

Negative Ack ⟵

CSE - 321

Congestion Control

— Expected feature
- Efficient
- Fair
- Avoiding Congesⁿ collapse

Congesⁿ detecⁿ

- RTT detecⁿ
- RTO detecⁿ (bad for wireless)
- Router hint
(- Packet drop)

cause wireless medium inherently lossy. Congesⁿ drop

MSS → Maximum segment size (Analogous to packet size)

Increase in cwnd

	After each ACK	After each RTT ($\frac{\#\text{ of ACKs}}{RTT} = \frac{cwnd}{MSS}$)
slow start	$1\ MSS = \frac{\Delta cwnd}{(cwnd/MSS)}$	1 cwnd
Congesⁿ avoidance	$MSS * \frac{MSS}{cwnd} = \frac{1\ MSS}{(CWND/MSS)}$	1 MSS