L10:- CHAPTER 3:- Transport Layer:-

In ch.2 we make an application like (e-mail, DNS, p2p, ...) then go from Socket → يروح Transport layer two protocol TCP, UDP.
• And have the port number.

Sec 3.1:- Transport layer services:-

* Communication become between transport layer and transport layer. physical فاردوير
* transport layer is logical communication:- we talke about softwear running in different hosts. between process
* transport run in end systems:- 1. Send side:- message into (Segment).
2. rcv side:- reassembles Segment into message.

•Δ:- Internet transport-layer protocol:-
TCP:- reliable, in order delivery:- بضمن يوصل
positive • Congestion control → full buffer. cause
• flow-control → control send data like send 100 packt/s or 50 p/s.
• Congestion setup → Hand shacking.

UDP:- Unreliable, unordered delivery:
• no-frills extension of IP → best-effort
↑ control ما في ضمان يوصل الداتا بس بحاول واذا ضاعت داتا ما الو علاقة وما في

Services not available:-
• delay guarantees.
• bandwidth guarantees.

Sec 3.2 :- multiplexing and demultiplixing :-

multiplexing :- *بعطي أوت بوت من خلال السيليكشن لاين بغض نظر عن الانبوت

Demult :- *بعمل عكس الـ multiplexing

- بيعت أكثر من application مثلاً بدي جوجل واليوتيوب بيعتهم على الـ socket الي بجيبهم الي بتعامل اياهم هو multiplexing و demultiplexer بيرجعلي أكثر من replay هو الي بتعامل فيه كل replay بيعته.

*How demultiplexing works :-

يعني من وين ولوين → يعني لوين

TCP / UDP : segment

1- Source port number 16-bit
2- dest port number 16-bit
3- header fields 32 bit
4- application data 32 bit

* الـ ip بيتعامل في layer ثاني.

الهوست زي ما حكينا بيكون عند تبيل فيها الـ IP Address والـ host بعمل request فيه (IP Add & port num) بطلبه من server وبيرجعلي جواب عملية الـ request والـ replay الي بتعامل هو multiplexer و demultiplexer المهم نعرف على شو بعتمد.

* Connectionless demultiplexing :

يعني ما بعمل handshaking (UDP) يعني أني أكون محدد

- destination IP Address
- destination port #

* Connection - oriented demux (TCP) :-

TCP socket identified by 4-tuple :-

- Source IP Address
- dest IP Address
- source port #
- dest port #

*لو عندي أكثر من جهاز و طلبنا جوجل كلنا بنطلب port 80 بس كل واحد بعمله لحاله socket عشان يرجع لكل حدا لحال ، حتى لو أنا فاتح جوجل مرتين لكل وحدة سوكيت لحاله لما يوصل سيرفر بوزع كل request على process بيكون multi threads عشان هيك كل واحد اله socket لحال.

Sec 3.3:- Connectionless transport : UDP

UDP:- User Datagram protocol.

Review:-

A- make best effort so it may be :- 1- lost ← لما بجيب مثلا disconnection
no frills ↗ segment
2- delivered out of order to app. ← توصل مخربطة

B- Connectionless:- 1- no handshaking.
2- each UDP segment handled independently of others. ← ما بيجيب كل الinformation ما بجيش كل ال Address

C. UDP use:- 1- DNS 2- SNMP :- Simple Network Mangment protocol.
3- streaming multimedia apps

D- Reliable transfer over UDP:- ← مش كثير مهمة
1- Add reliability at application layer
2. Application-specific error recovery

* Why using UDP:-

1. no connection establishment (no or less delay).
2. Simple.
3. Small header size. ← 8 Byte — 32 bit → (4 Byte)

32 bit

source port	dest port
length	checksum

← كله 8 Byte

4. no congestion control
5. Faster than TCP.

* CheckSum:- 16 bit

عشان نعرف اذا داتا وصلت بدون error آو لا

* عندي داتا بقلبها لـ 1's complement وببعت على reciver بفحص بجمع داتا مع checkSum ↓ 1's complement → اذا كلهم 1's → اذا برجعلي Yes فش ايررور اذا NO في ايررور.

(حاصل جمع رقميه مع نفسها 1, 0 بهمنيش)

Ex:- add two 16-bit:

num1 1110011001100110

يرجع أجمعه carry num2 + 1101010101010101

sum:- (1) 1011101110111011

→ 1

final Sum:- 1011101110111100

1's complement بعمل checksum:- 0100010001000011

* بيبعت الداتا (num1, 2) ومعاهم كمان ال checksum على ال reciver بجمع checksum مع حاصل جمع رقميه (sum) بجيني كلهم 1's اذا كلهم 1's فش ايرور.
اختلاف bit واحد يعني في ايرور.

Sec:- 3.4:- principle of reliable data transfer:-

عشان أتأكد أنه الداتا وصلت ال reciver بيبعت Acknowledge.
عشان أتأكد أنها فش ايرور بعمل checksum.
عشان أعرف أنه ال packet وصلني قبل هالمرة أو لا عن طريق رقم sequence num زي ال ip

* هسا channel الي ببعت من خلالها داتا يا بتكون ← reliable or unreliable
اذا هي reliable فش مشكلة عندي اما اذا هي Unreliable بدي أحاول أخليها reliable

* Reliable data transfer:- rdt

Send side { rdt_Send():- ← ال Application بيعمل call لهاد الفنكشن
udt_Send():- بيبعت ال packet على channel
→ unreliable channel

receive side { rdt_rcv():- على الرسيفر بستقبل البكيت وبتأكد أنه فش ايرور.
deliver_data():- بيبعت data لل application بحكيله أنه وصل كل شي.

→ rdt:- reliable data transfer

FSM :- Finite State Machiens

انا عند 1 وبدي أروح على 2
Actions بجهيزي

أنا حاليا

* لازم يكون في سبب عشان أروح على state2 فمثلاً بكون أستنى في packet وجاء وصلني
ببعث مثلاً Acknowldgment هاي عبارة عن Actions
على state1 يبعث البكيت وعند state2 بكون أستنى في Acknowledge

⇒ Type of channel :-

A) rdt 1.0 :- reliable transfer over a reliable channel.
معنى أن ال channel تكون reliable :
• no bit errors 0 ⇄ 1 (تغير) • no loss of packet
signal loss , تشويش , full Buffer ... etc

* Sparate FSM's for :-
1- Sender :- rdt_send(data) ; call function to send — ببعث
wait for call from above — packet = make_pkt(data) ; make data as packet
udt_Send(packet) , يبعتها على udt
↑ Application layer
2- receiver :- rdt_rcv(packet) , يستقبل البكيت
extract(packet, data) , بعملها أكسترات
deliver_data(data) , يعطيها لل Application layer
call بعمل
rdt1.0 اذا فش إيرور
اذا في إيرور

B) rdt 2.0 : channel with bit errors :-
→ checksum to detect bit errors. الحل نرد نبعت (data)

Q :- How to recover from errors :-
• Acknowledgement (ACKs) :- يرجعلنا مسج يبعث اي اشي المهم يحكيلنا
أن البكيت وصل تمام OK
→

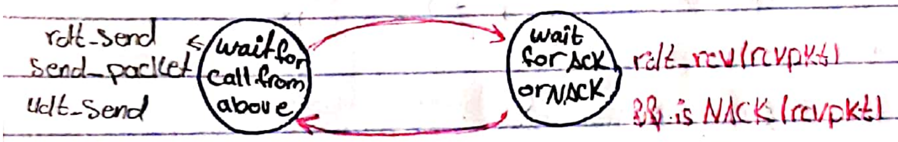
• negative acknowledgements (NACKs):
بتحكيلنا أنه البكيت فيه ايرور.

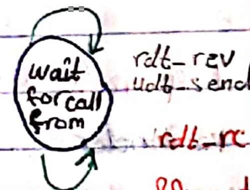
→ rdt2.0 : byond rdt1.0 :-
• error detection
• feedback :- control msgs (ACK, NACK).

* كيف راح نستعملهم (NACK , ACK) : slide-29
وملتني data بجلها send قبل ما أرسلها بحولها لبكيت بس مش بكيت
مع داتا بالحاله بكون معاها Sndpkt = make_pkt(data, checksum) : checksum

أنا في state2 : اذا وصلني البكيت فش فيه مشاكل ببعت Acknowledgment
اذا فيه مشاكل ببعت NACK

اذا وصلني (ACK) بقدر أبعت بكيت جديد ، ممنوع أبعت بكيت جديد الا لو وصلني ACK.





أنا في state 1 wait call from above Sender

reciever

Sender

* اذا وصلني (NACK) ممنوع أبعت بكيت جديد لحد ما يبعت (ACK)
* وأنا على receiver اذا ما في corrupt بطلعه لفوق اذا في ببعت (NACK)

⇐ في مشاكل فيهم : لما يكون في ايرور
● اذا (ACK or NACK) فيهم مشاكل : corrupt خربة
Sender بعرفش بالضبط شو الاشي فبضل بدو يبعت الـ packet بس ممكن يصير duplicate
فبنحط مع البكيت (Sequence number) عشان أعرف اذا البكيت هاد
انبعت من قبل أو لا.

L11:- C) rdt3.0:- channel with errors and loss.

ACK give me if the packet loss or not.

check sum give me if the packet corruption or have error. (خربت)

* اذا ما صار ACK بعرف أنه البكيت ضايع، طيب واذا ال ACK ضايع؟

slide 40 →

Sol:- Stop-and-Wait

Sender: ↓

بتوصلني داتا من Application layer بعطيها بكيت مع sequence number SQN

وكمان CRC ← لقدام بنحكي عنها هي للايرور و check sum للايرورز كمان

ببعت البكيت وكمان ببلش timer ← udt_send(pkt), Start_timer

* لأنه ممكن ال ACK ما يوصل، مثلاً أنا بستنى 10ms ← اذا ما أجاني ACK بعرف

أنه في مشاكل فبيد يبعت ال packet (القديم) timer ← timeout خلصت ال 10ms

و SQN عشان نتأكد أنه البكيت وصل بالترتيب وما يصير duplicate للبكيت.

← بيبعت البكيت الجديد لما يوصلني ACK & SQN(ACK) = SQN → مفش غلط فيها.

بيعمل Stop timer.

* بس ما أبعت بقعد أستنى اذا أجاني ACK سلبي ↑ أو اذا وصلني NACK مع ايرور

بظل استنى لحد ما يوصلني ACK. أما اذا ما وصلني ACK بالمرة timeout

بيعمل بحاجة resend.

on receiver → ACK = ACK(SQN, CRC)

⇐ الأشياء الي ما بتخلي ال ACK توصلني:-

1- البكيت ما وصل أصلاً.

2- البكيت الأصلي وصل بس ال ACK صارلها loss.

3- ديلاي على ال ACK.

Receiver:

أجاه بكيت ما فش فيه ايرور & SQN صحيحة ⇐ بعمل اكسترا كت لداتا وبيعتها

ال Application layer وبيبعت ACK فيها SQN. وبزيد SQN++ حابده الجديد بصير

أجاني بكيت فيه ايرور أو SQN مش صحيح ⇐ بيبعت ACK → البكيت القديم

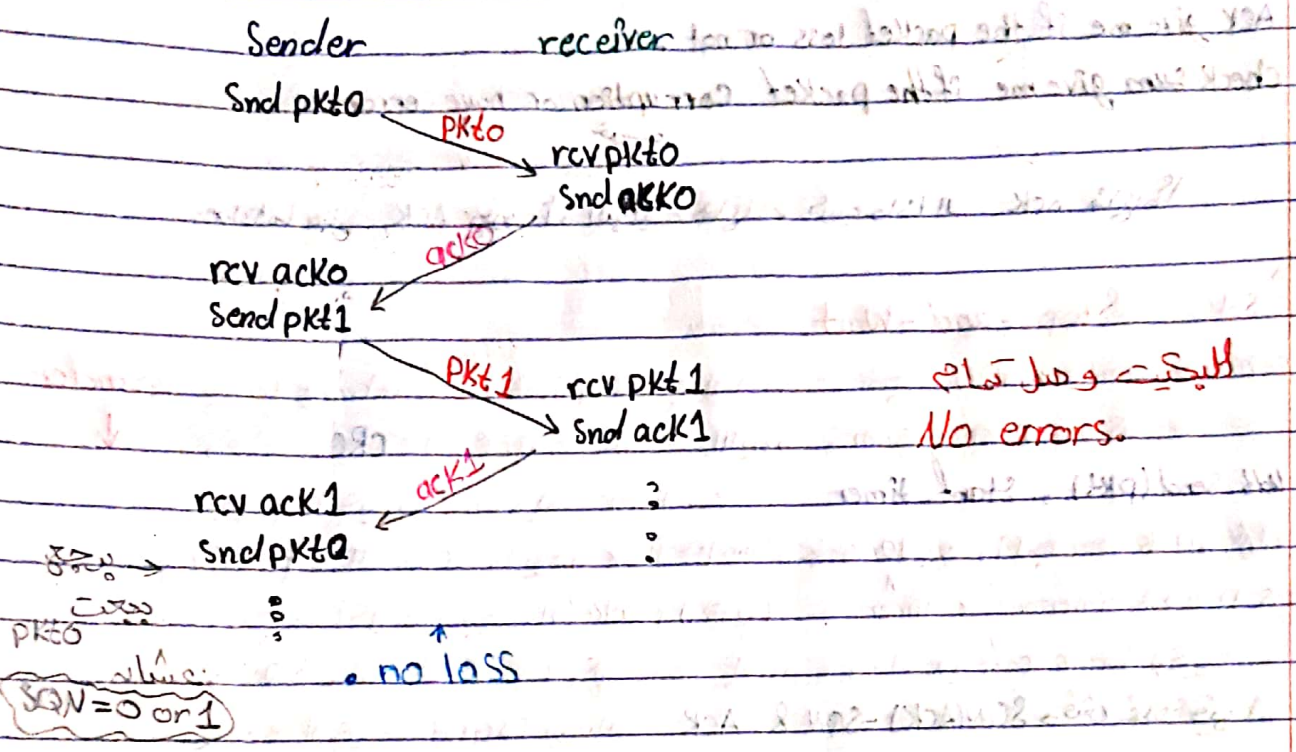
لما يجرا بيرجع يبعته لأنه أنا عملت SQN++ ووصلني SQN القديم فصار

SQN ≠ SQN++ فبيبعت ACK ويرجع يعيد الارسال البكيت من أول.

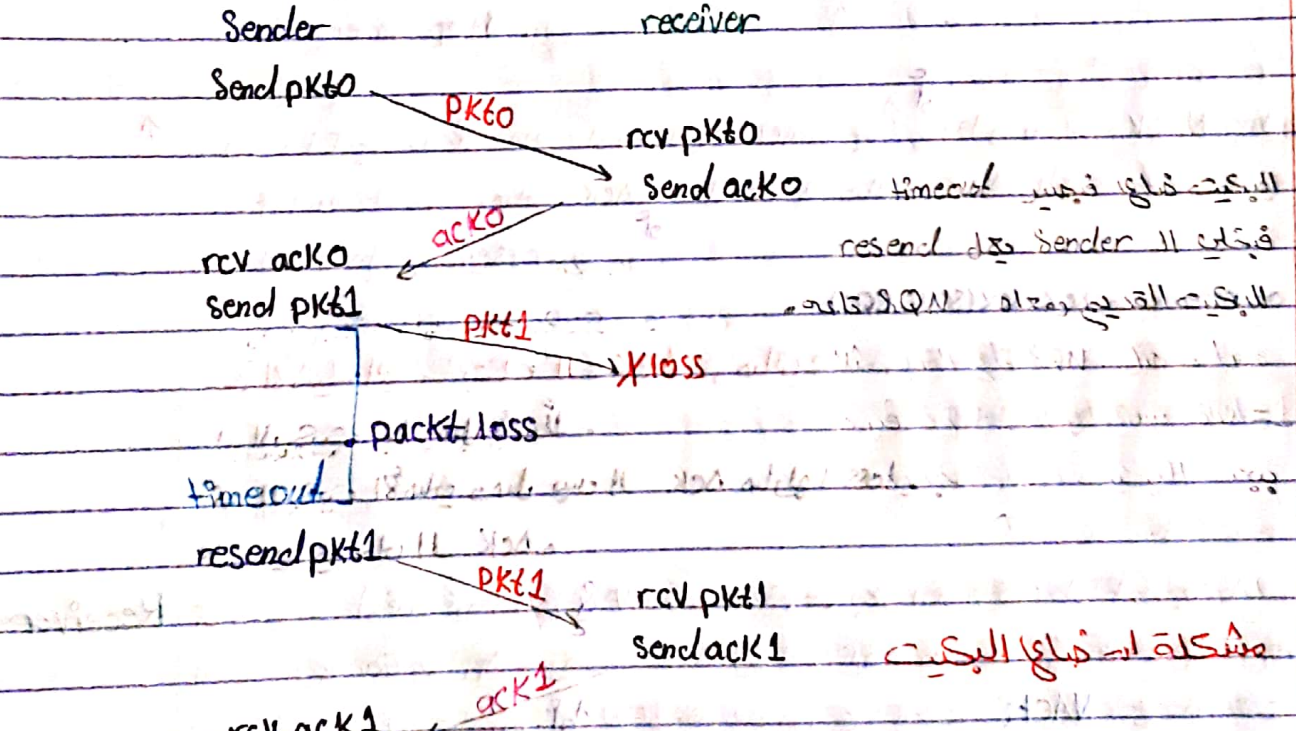
* In Stop and wait for SQN just 1-bit minimum بيكفي بس 1

بتقلل عدد البت عشان أخفف من ال overhead. SQN ما الها دخل في داتا.

rdt 3.0 Actions.



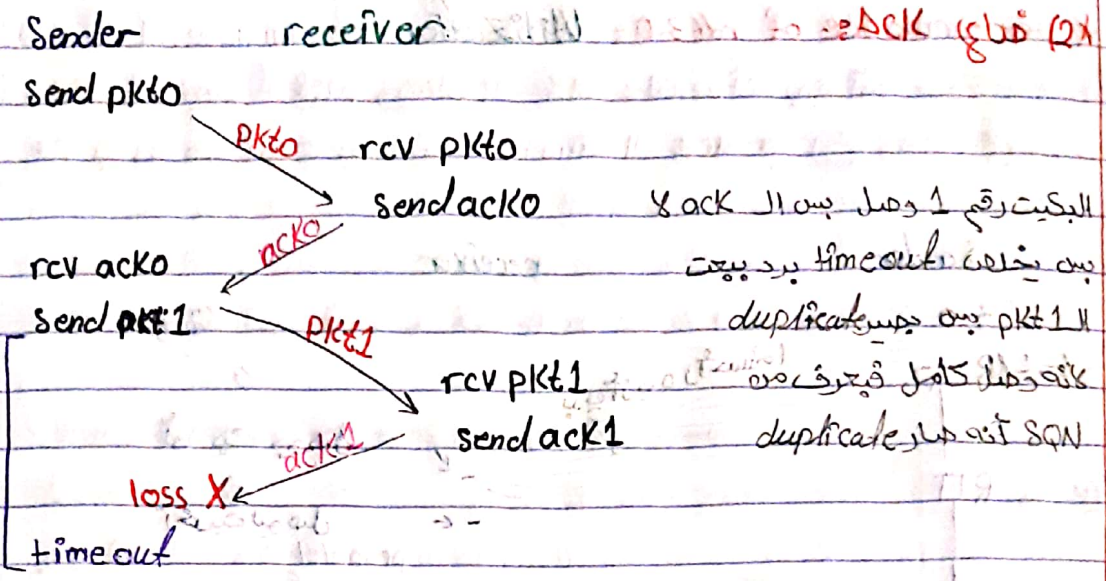
البكيت وصل تمام
No errors.



* البكيت ضاع فجبر timeout
فبتلي الـ Sender يعمل resend للبكيت القديم بنفس الـ SQN تبعه.

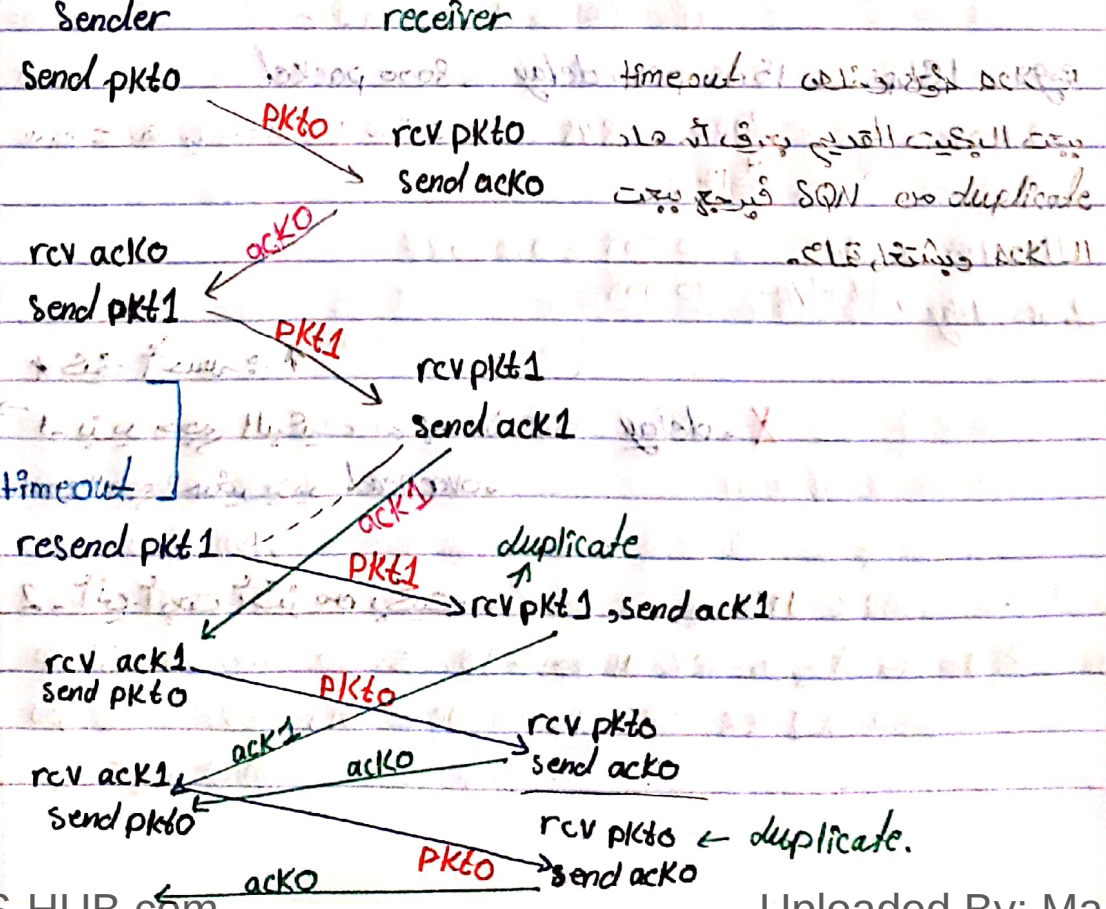
① مشكلة اذا ضاع البكيت

(2 فقدان ACK:



البكيت رقم 1 وصل بس الـ ack
بس ضاعت، timeout برد بيعت
الـ pkt1 بس جبس duplicate
لانه وصل كامل فيحرف من
SQN انه صار duplicate

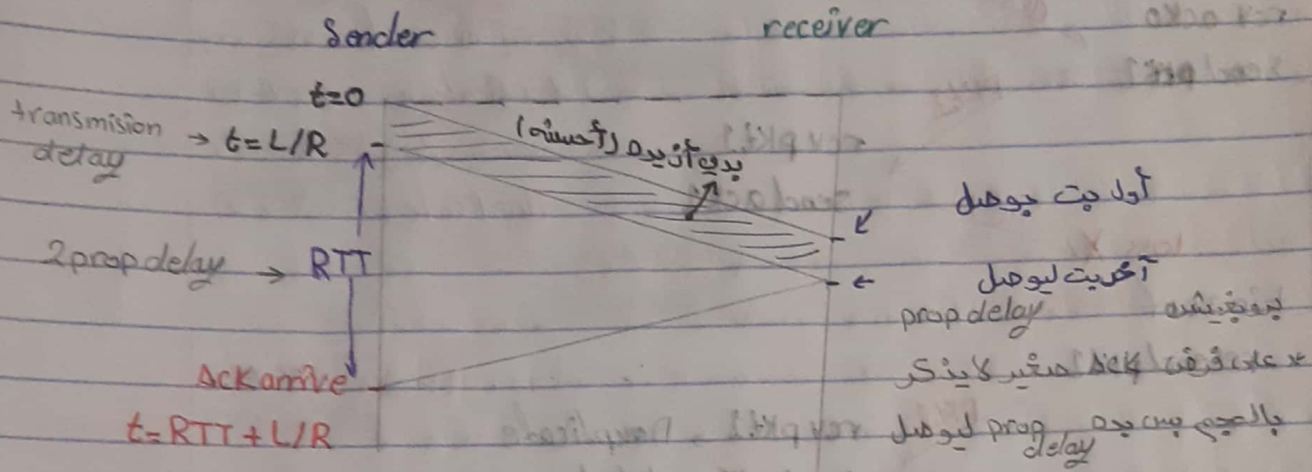
(3 Delay ACK: الـ ACK طول ليوصل



timeout
بيعت البكيت القديم بس انه صار
SQN من duplicate فيرجع يبعت
الـ ACK1 ويشتغل عادي.

* performance of rdt3.0:- Utilization

المشكلة أني ببعت packet وبستنى الـ ack ، لنفرض أني بدي أبعت بكيت بمسافة كثير بعيدة هل أبعت بكيت واحدة وأستنى الـ ack ؟ طبعًا وقت راح يكون ...



RTT كبيرة بتكون الـ Utilization كثير صغيرة

$$U_{sender} = \frac{L/R}{RTT + L/R}$$

في stop-and-wait

مستحيل أبعت بكيت جديد قبل ما يوصل الـ ack قبل $t = RTT + \frac{L}{R}$

المشكلة بالمسافة لما تكون بعيدة

e.g.:- 1Gbps , 15ms prop delay , 8000 packet

$$D_{trans} = \frac{L}{R} = \frac{8000}{10^9 = 1G} = 8\mu s$$

$$U_{sender} = \frac{L/R}{RTT + L/R} = \frac{0.008}{30.008} = 0.00027$$

روحة رجعة 2.prop delay ↗ (RTT)

* كيف أحسن :- ↑

1- بزيد حجم البكيت ، بس مشكلته delay . ✗ → بس مش خيار

note: الحجم الكبير بزيد overhead.

2- أني أبعت أكثر من بكيت ✓ قبل ما يجيني الـ ACK

على فرض أني بعتت 50 بكيت قبل ما يجي الـ ACK طيب إذا ما وصل شي ؟ عمل بدي أبعتهم كلهم

لكل بكيت SQN والـ ACK إذا ما وصلني ACK لهاد البكيت بس بعيد أبعتها لحاله

2- way to performance:-

* **pipelined protocol:-**

اذا بدأوا أبعت بكيت بكيت فيها مشكلة delay وما بقدر أبعت بكيت تاني إلا
لما يوصل الـ ack للبكيت اللي قبل.

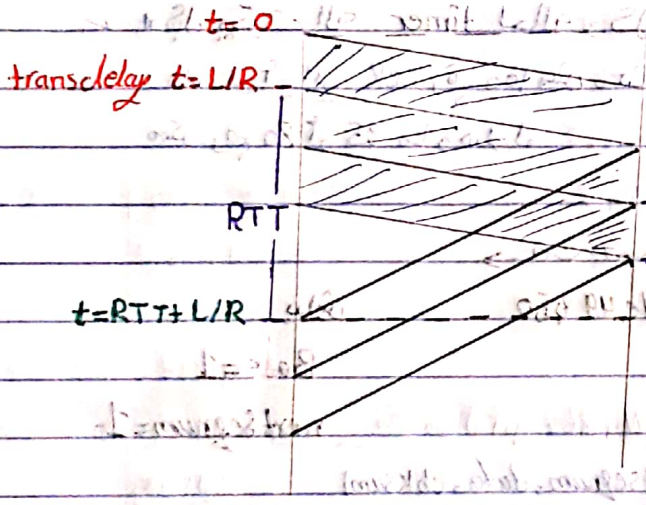
pipelined:-

بصير أبعت أكثر من بكيت ورا بعض أو مرة وحدة وبصير استقبل أكثر من ACK

Two Generic forms of pipelined protocol:-

1- Go-Back-N 2- Selective repeat

Sender receiver → (pipelining: increas Utilization)



Utilization = الوقت اللي بدته / الوقت الكامل

3packet بعت

$$U_{sender} = \frac{3L/R}{RTT+L/R}$$ → U زدت ٣ أضعاف

data transfer أخذت

e.g اللي قبل ← $U_{sender} = \frac{3L/R}{RTT+L/R} = \frac{0.0024}{30.008} = 0.00081$

• Go Back N

* cumulative ack — يبعت receiver ، ACK تيجي قبل ما يجي N packets يبعت *
يعني أجاني البكيت الاول والثاني والثالث مسالا وصلت عند اللي :
واحد لحين واصلين يبعت ACK عنهم الثلاثة

* بس لو وصل الاول والثالث والثاني ما وصلوا بيبعت ACK بس للثاني الي أجي بترتيب.
أجاني الاول والثالث. والثاني ما أجنيش ما يبعت ACK لأنه في Gap و SQN خطأ.

في غلط ←
شرح بوضح →

✓ *أذا timeout خلص كل الي ما أجاني ACK الهم بيرجع يبعتهم.
اذا أجاني ACK رقم 100 معناها كل الي قبله كلهم وصلوا ، 6 مثلاً وصلني ACK رقم 7 ضمنياً
بكون عارف أنه ACK رقم 6 وصل ... الخ.

• Selective Repeat:-

✓ * كل بكيت اله timer لحاله وكل بكيت بجيني بيبعتله ACK لحاله
هان لو أجاني ACK رقم 100 مش معناها انه 99 وصل فالي قبله وصل وصيلة.
ممكن رقم 80 أو 35 ما وصلوا.

* individual ack in rcvr

←o~o~o~o~o~o~o→

L12 :- * GBN :- Sender extends FSM :- slide 49 & 50

بيلش
Base = 1
next seqnum = 1

```
rdt_send(data)
if (nextseqnum < base + N) {      معناها الي بيبعت N-pkt قبل ACK
    sndpkt[nextseqnum] = make_pkt(nextseqnum, data, chksum)      يعمل البكيت
    udt_send(sndpkt[nextseqnum])      يبعت البكيت
    if (base == nextseqnum)
        start_timer
    nextseqnum++
}
else
    refuse_data(data)
```

```
timeout
start_timer
udt_send(sndpkt[base])
udt_send(sndpkt[base+1])
   ⋮
udt_send(sndpkt[nextseqnum-1])
```

يبعت كل البكيت الي ما أتكالها ACK →
بتفقدهم وحدة وحدة زي for loop

GBN Actions

Slide 51

Sender window (N=4) — sender — reciver

[0 1 2 3] 4 5 6 7 8 — sen pkt0 → recv pkt0, send ACK0

// // تقسيهم ما — send pkt1 → rcv pkt1, send ACK1

// // تغير اشي — send pkt2 → X loss

// // — send pkt3 (wait) → rcv pkt3, snd ACK1 discard

ACK بيكيت 4 معنوي ايجتو إلا اما بجيني أول — ACK بحمل وصلني

Appli. layer ← على 0 [1 2 3 4] 5 6 7 8 Shift بقدر اعمل — rcv ACK0, snd pkt4 → rcv pkt4, snd ACK1 discard

بتروح ← 0 1 [2 3 4 5] 6 7 8 — rcv ACK1, snd pkt5 → rcv pkt5, snd ACK1 discard

ignore duplicate ACK — timeout

*أول 4 بكيت رحين الويندو ، ببعتهم بدون ACK بحدها مع كل ACK بتوصلني بعمل Shift للـ window وببعت البكيت الجديد الي صارجوا الويندو

* الـ pkt2 ضاع ووصلني pkt3 ببعث آخر ACK وصلني صح دالي هو ACK1 resend(ACK1)

معنى resend ACK1 ← 1 و pkt0 واصلات 100% معناها بعرف أنه pkt2 ضاع ، بوصلني عند sender ACK1 فش أله معنى لا

فسا لما بجيني timeout ببعث كل البكيت اللي ما تعللهم ACK لـ بكيت 3 و4 و5

علي أنا resend ACK1 بتعللهم discard ، وبرجع ابعتهم.

لما أرجع أبعث بيلش من أول فيبعتهم كلهم قبل ما يجي ACK — send pkt2 → rcv pkt2, Send ACK2

// // — Send pkt3 → // // 3 — // // 3

0 1 [2 3 4 5] 6 7 8 — Send pkt4 → // // 4 — // // 4

0 1 [2 3 4 5] 6 7 8 — Send pkt5 → // // 5 — // // 5

مثال ←

Sender — reciver

[0 1 2 3] 4 5 6 7 8 — pkt0

*البكيت وصل الي ما وصل ACK1 ، لا ACK الي ضاع — pkt 1

بحمل تحت نفس ترتيب ACK — pkt 2

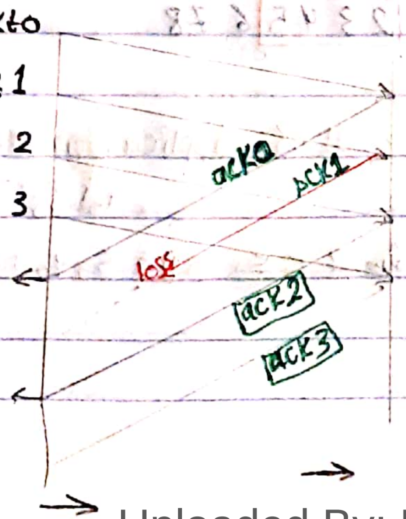
pkt 3

0 [1 2 3 4] 5 6 7 8

0 1 2 [3 4 5 6] 7 8 — وصل ACK2 معناها

وصلوا pkt0,1

وأبعت ACK0,1 بس ACK1 ضاع



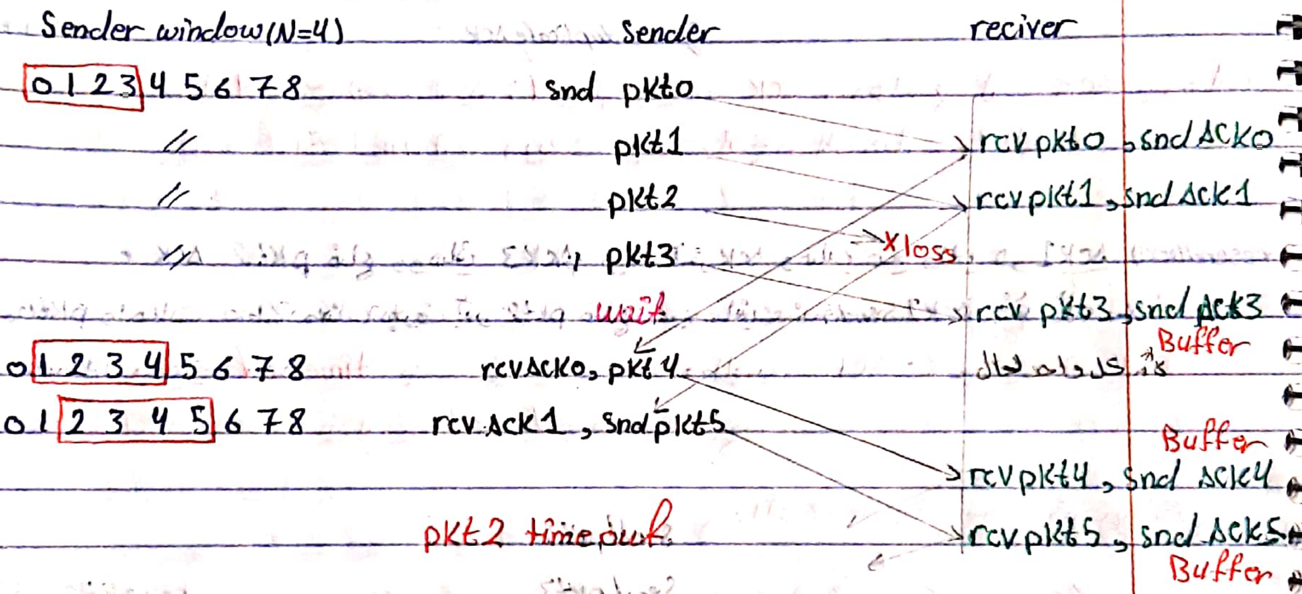
* GBN حكينا هو cumulative ACK، فلو ضاع الـ ACK مش مشكلة لأنه الـ ACK الي بعده بضمنلي 100% انه البكيت الي قبل واصليه فيعمل Shift عادي ويكمل.

→ مهم

Selective repeat:-

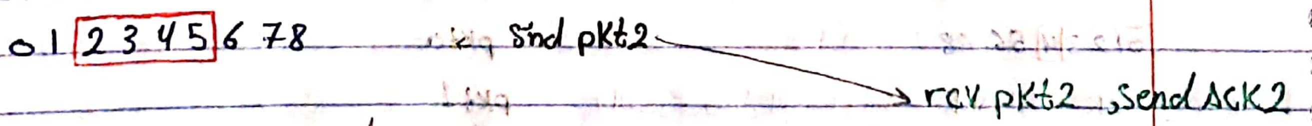
قلنا لكل الـ pkt الها timer و ack لحالها وما بيعتمد على اشي ثاني.

ومعني الـ ACK من بكيت معين والي قبله لا بقدر أبعث البكيت الجاهز على الـ Application layer لأنه ما بقدر إلا لما الي قبله مش واصل.



بكيت 2 راح ولا وصل بكيت 3 بيبعث ACK3 لأنه كل بكيت لحال، وحطها بين بكيت 3 وصل.

وهنا لما يصير timeout: بس يبعث الـ pkt 2، pkt 3,4,5 بضلهن محفوظات على الـ Buffer



بيبعثهم على الـ Application layer، لأنه الـ window = 4 وهما الـ 4 كلهم وصلوا. → deliver:- pkt 2,3,4,5

زي ما حكينا فوق بكيت 3 و4 و5 قبلهم بكيت 2 مش واصل فما بقدر أبعثهم إلا ليوصله.

Selective repeat dilemma:

* بدنا نشوف شو علاقة الـ(SQN) مع الـ window size.

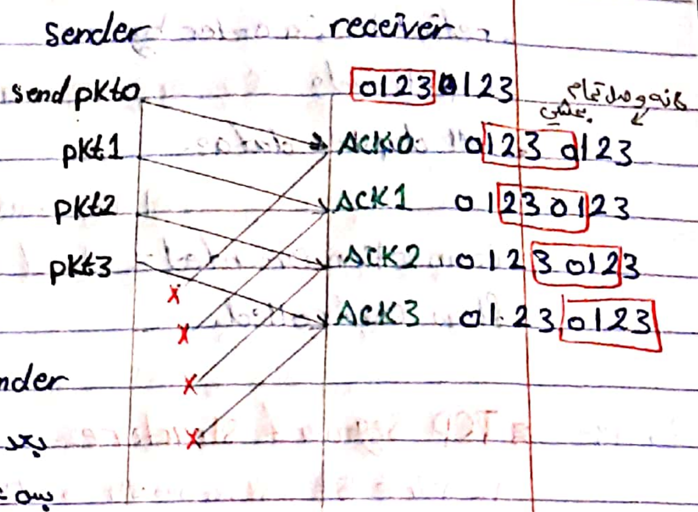
ex:-

* 2 bit sequence number , W = 4 ,

SQN = 2 bit ← لا

بقدر أبعث 4 بكيت قبل أول ACK

0123 0123



* نفترض أنه كل الـ ACK ضاعت

ما بيعمل shift للـ W لأنه ما وصله ولا أي ACK

مهم

sender بضل يستنى في ACK ما وصله بعمل resend بعد timeout. بعمل الـ resend لـ 0123 الأولى

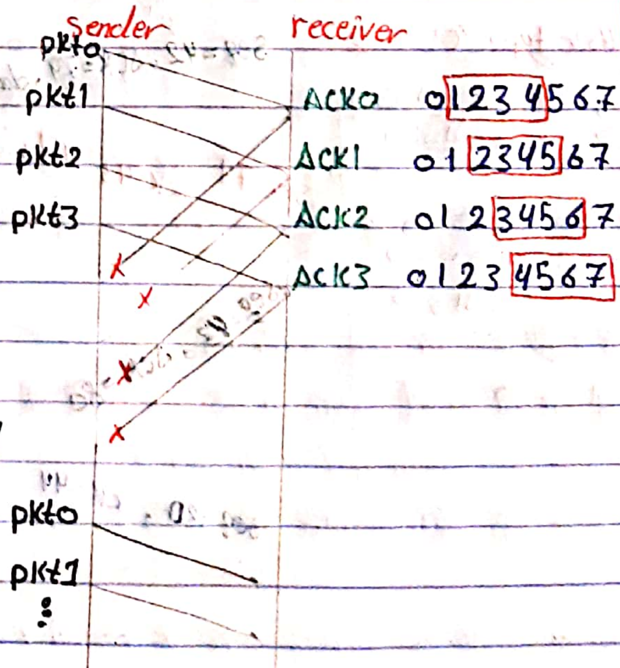
بس على الـ reciver لازم يفهم أنهم مش 0123 التانيات

لو أنا عندي 01234567 بحتت 0123 والـ ACK ضاع على reciver بكون يستنى يوصله 4567 فلما أعمل resend بكون سهل عليه يميز شو وصلني

بس أنا عندي 0123 0123 بصير خربطة فلازم أحط 1 bit إضافي ← 3 bit seq. num

0-7 : 000-111 0123 4567

0123 4567



resend 01234567

* SQN = 2 W

Sequence number = 2 window size

L13:- Sec 3.5:- connection-oriented transport TCP:-

TCP:- Transmission Control protocol

- point to point:- One sender, one receiver.
- reliable, in order byte:- بيبعث بايت بايت بترتيب
- pipelined:- TCP congestion and flow control set window size.
- full duplex data:- بيبعث ويستقبل بنفس اللحظة بالاتجاهين
  - half // الاتجاهين بس مش بنفس اللحظة — MMS:- maximum segment size — simeplex // بس باتجاه واحد
- Connection Oriented:- Hand shaking
- flow controlled.

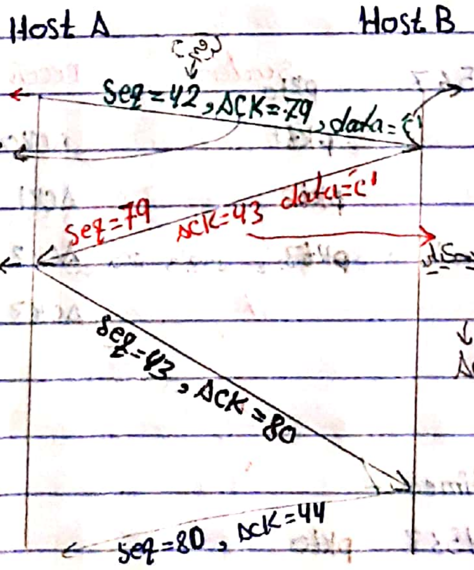
3.5:- TCP Segment Structure:-

Slide 58 (الي مش محددة) → Slide 62 (بالتفصيل) see it.

TCP sequence number:- number of first byte in segment data 32 bit

Acknowledgements:- بيحكي انه ال data وصلت أو لأ

Ex:-



كل الي قبل 79 وصلوا فالهوست B راح يبلش ب seq = 79 ←

الهوست B بيجي وصلني 1B هلا صحيح بيلش من Seq=43

الهوست B بده يرجع ACK عشان يحكيلي أنه داتا وصلت فيرجع بال ACK

ACK = Seq + # Byte for data
= 42 + 1 = 43

وصلت داتا تمام من ال ACK مسابقي أرجع أبعت داتا جديدة لل 1 Byte فتبعث ال ACK

ACK = Seq + # Byte
= 79 + 1 = 80

Δ:- TCP round trip time, timeout (RTT)

لوفاي بكيت برجع أبعته بعد time out يخلص ، هم لازم نحطه مش أقل من RTT

هل RTT ثابت ؟؟ لا، كيف راح نحسبه ؟؟

لما يوصل ACK بعرف أنه البكيت وصل وبعمل stop وبحكي أنه هاد ال RTT .

• timeout لازم يكون أكبر من RTT بشوي مش كثير.

• too short RTT?

قسنا ال RTT وطلع رقم متغير لو جبنا timeout رقم متغير بعمل duplicate برجع يبعث البكيت.

• too long RTT?

بصير تأخير في بعث البكيت لأني بستنى وقت أطول لا الرويوت ينزل.

فبدنا رقم معقول نحسبه من RTT.

How to estimate RTT?

Sample RTT:- measured time from segment transmission until ACK receipt.

• ignore retransmissions.

⇒ $\text{EstimatedRTT} = (1-\alpha) * \text{EstimatedRTT} + \alpha * \text{SampleRTT}$ (القديم ↑ EstimatedRTT)

• $\alpha = 0.125$

<—o~o~o~o~o—>

L14:- Safety margin — ال Estimated RTT ما بنجيبها لحالها بنحط معها اشي اسمه

time interval:- Estimated RTT + Safety margin.

• large variation in Estimated RTT → large safety margin.

* estimate SampleRTT deviation from EstimatedRTT:-

⇒ $\text{DevRTT} = (1-\beta) * \text{DevRTT} + \beta * |\text{SampleRTT} - \text{EstimatedRTT}|$

$\beta = 0.25$

⇒ $\text{Time Interval} = \text{EstimatedRTT} + 4 * \text{DevRTT}$

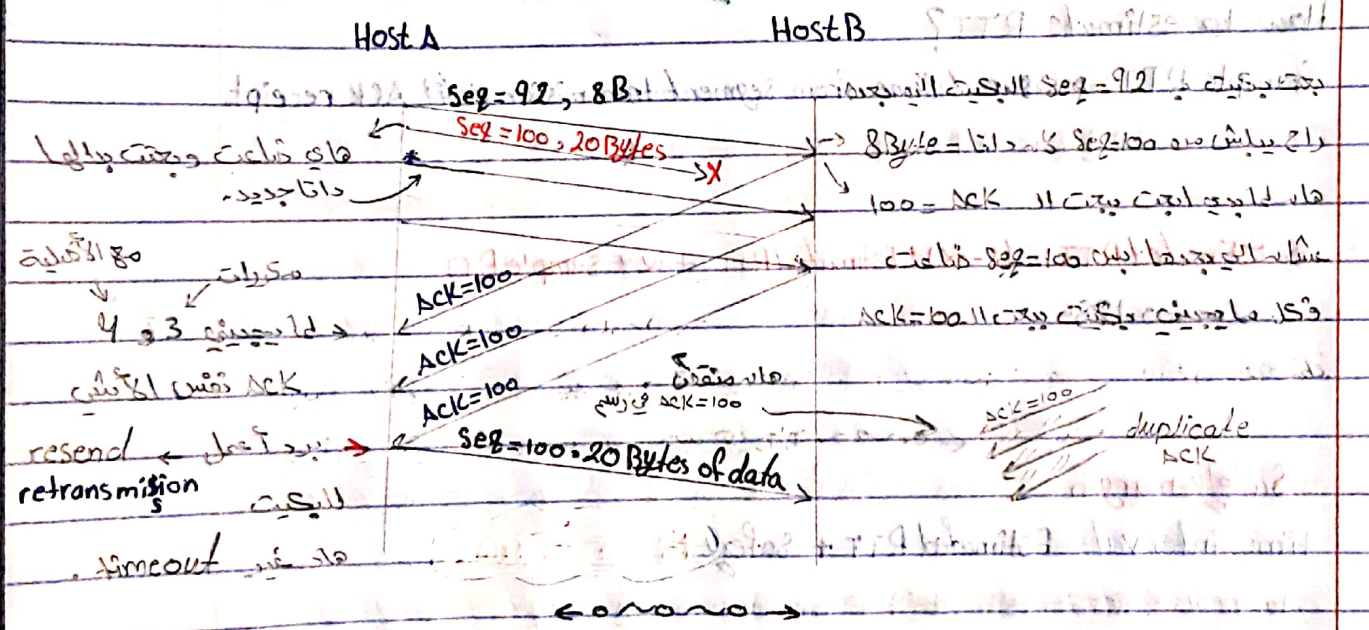
↑ estimatedRTT — ↑ safety margin

3.5 Reliable data Transfer :-

* TCP creates rdt service on top of IP's unreliable service:-
  - pipelined Segments :- يعني بنبعت أكثر من سيجمنت
  - Comulative acks :- الـ ACK بتكفي عن كل اللي قبل.
  - Signal transmission timer :-

* Retransmission triggered by:-
  - timeout
  - duplicated acks

Duplicated acks :-



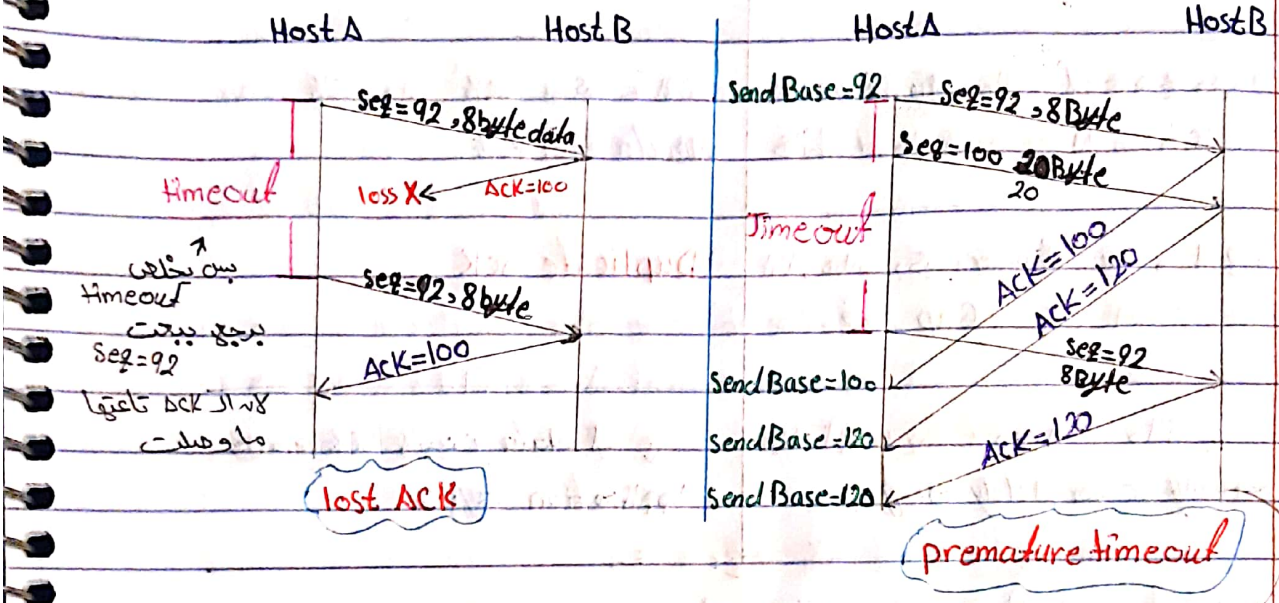
* TCP Sender events:-
  - Data rcvd from app: Application layer داتا جتلي من الـ
    1. creat Segment with Seq #
    2. Seq # not started from Zero → Seq أو بكتب بدل ٥ بيتات بيلش random وبكفي عن الـ Byte مش عن البيت.
    3. Start timer if not already running.

■ Time out:- لما يخلص

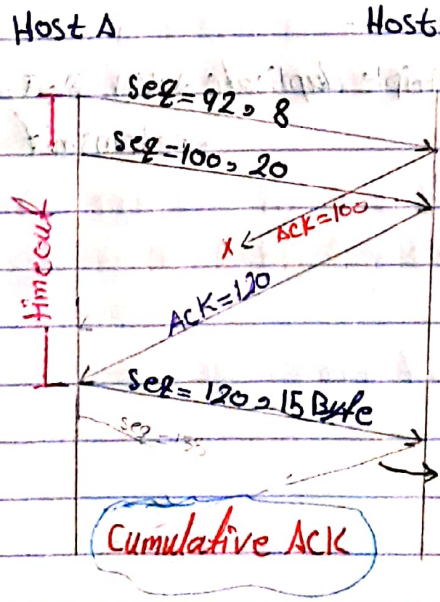
1- retransmission Segment / Data اليوما نخليها loss مار ACK
2- restart timer بالعربي

←o~o~o~o→

Δ:- TCP :- retransmission Scenarios:-



timeout سريع خلص قبل ما يوصل الـ ACK يرد يبعت البيت فبعت Seq=92, 8Byte data مسبقا رجع ACK=120 وصل فهو بعرف انه كل البيتات اللي قبل وصلوا غير بيتت ACK=120 لانه عرف انه البيت Seq=92 هو duplicate



فلما الـ ACK=100 ووصل الـ ACK=120 بكمل عادي لانه حكيتا لما يوصل ACK بجي عن حاله وعن اللي قبل فلما وصل ACK=120 كان متأكد انه كل البيتات اللي قبل وصلوا.

See slide 70 , 74 ← المعدلات

event at reciver	TCP receiever action
اذا وصلت داتا مع ال seq وال ACK وكله تمام .	delay ACK upto 500ms (0.5s) ما يبعت على طول ACK عشان ما يكون ACKS كثيرة ورا بعض
اذا وصلني بكيت بال order صح وقبله كان في بكيت مش عامله ACK	يبعت ACK وال ACK بحكي عن الاثنين cumulative ACK.
اذا وصلني بكيت out of order و High seq # هاد في Gap	Duplicate ACK
اذا اجاني بكيت و بعمل fills ↓ partially or completely for gap	يبعت ACK وبيعت data ل ↓ Application layer.

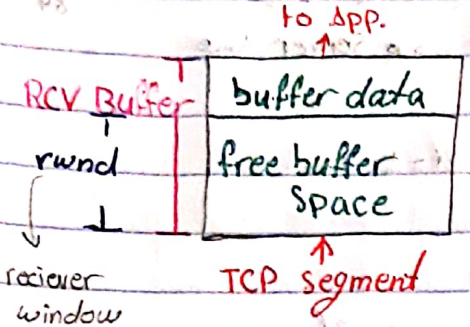
Δ:- TCP fast retransmit:-

معناها ما أستنى ال time out لانه ممكن يكون طويل واذا كان طويل بنزل عندي ← throughput . بعمل detect ال سيجنال فيها مشكلة عن طريق duplicate ACKs . ← حكينا قبل عنها ⇐

⇐ اذا اجاني (triple duplicate ACKs) ← total 4 ACK لنفس الرقم بعمل retransmit .

3.5:- flow Control:-

عشان ما يصير عندي overflow الحل أني أراقب الـ buffer تاع الرسيفر.



→ Rcv Buffer size 4096 Byte

rwnd:- free space in Buffer that can I recieve data on it. like selective repeat remember: GBN make cumulative Ack but not make buffering.

→ * TCP make buffering and cumulative Ack.

⇒ How can I Know How much Space I have?:-

At Receiver:

- last Byte Read (by the application).
- last Byte Rcved (arrived from the network).

* • last Byte Rcvd − Last Byte Read <= (RcvBuffer).

⇒ rwnd = RcvBuffer − (last Byte Rcvd − last Byte Read).

size

ex:-

last Byte Read = 100 (send to Application), last Byte Rcved = 200

size Buffer = 1000.

rwnd = 1000 − (200 − 100) = 900 Byte space free.

At Sender:-

- last Byte Send
- last Byte Acked

* • last Byte Send − last Byte Acked <= rwnd

في المثال اللي قبل طلع عندي 900 Byte free فأنا مسموحلي أبعت 900 قبل ما يجي الـ Ack.

3.5:- Connection management:-

كيف أني أبلش الـ connection عن طريق الـ handshaking. random

أول بكيت بحله exchange بيبعت معه bit اسمه (SYN bit = 1) و Seq = x

لما server يرد SYN bit = 1 و ACK bit = 1 و AckNum = x+1. slide 80 84

3-way handshaking ↗

بصير أبعت SYN bit = 1 اذا في اتصال وتمام بيرجعلي ACK bit = 1 و SYN bit = 1

* TCP Close Connection:-

بيبعت bit اسمه ← FIN (finish) ← FIN bit = 1 وعلى الرسيفر يرجعلي FIN bit = 1 و ACK FIN = 1 bit
بيسكر من طرف sender
بيسكر من عند rcv.

RST هاي بتسكر جهتين ومن جهة ثانية كمان
لا بتخش

* في كمان bit اسمه RST: ← reset اذا بدي أسكرها أو اللي في الـ connection

* اذا عملت FIN: سكرت الـ socket تبعي أبعت داتا بس ممكن توصلني

<-o~o~o~o~o->

L15:- Sec 3.7:- TCP Congestion Control.

⇒ Sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs.

* If ACKs not recieved (ما وصلني) then I Know there is an issue in network so TCP decrease cwnd (TCP Sender Congestion window size).

. Additive increase:- increase cwnd by 1 MSS every RTT until loss detected.

. multiplicative decrease:- cut cwnd in half after loss.

* $rate \approx \frac{cwnd}{RTT}$ bytes/sec

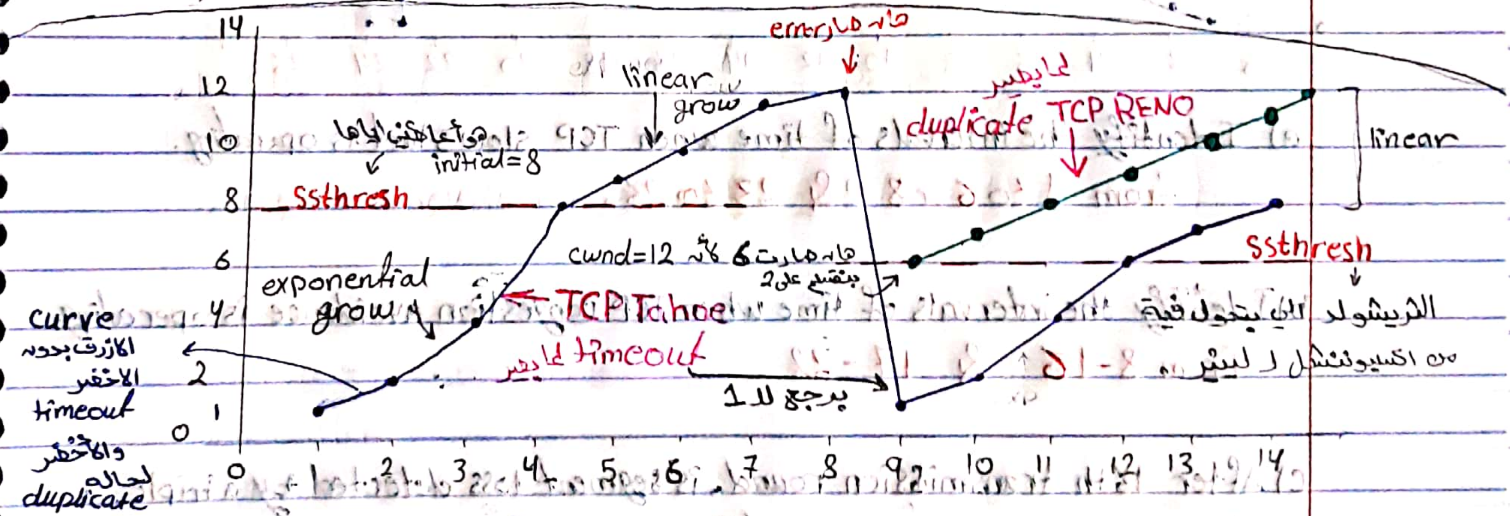
* TCP Slow Start:-

initially send 1 segment (cwnd = 1 MSS), then every time I have an Ack make increment by double for cwnd so that mean I send 2 segment then 4 segment ... etc. note:- grow exponential until threshold then grow linear.

* TCP: detecting, reacting to loss.

1) If happened timeout:- Tahoe
cwnd back to be 1 MSS, the TCP slow start ↑.

2) If loss by 3 duplicate ACKs: TCP RENO.
happened when packet loss. Sec 3.5 L14
cwnd divide by 2 (cwnd/2) then grows linearly.



exponential increase swich to linear when we reach ssthresh

* ssthresh:- Slow Start threshold.

ssthresh after loss (error) become → ssthresh = cwnd/2 befor loss

أعلى قيمة وصلتلها cwnd 1st segment قبل الايرور ‎/2. شبيه RENO في حسابه

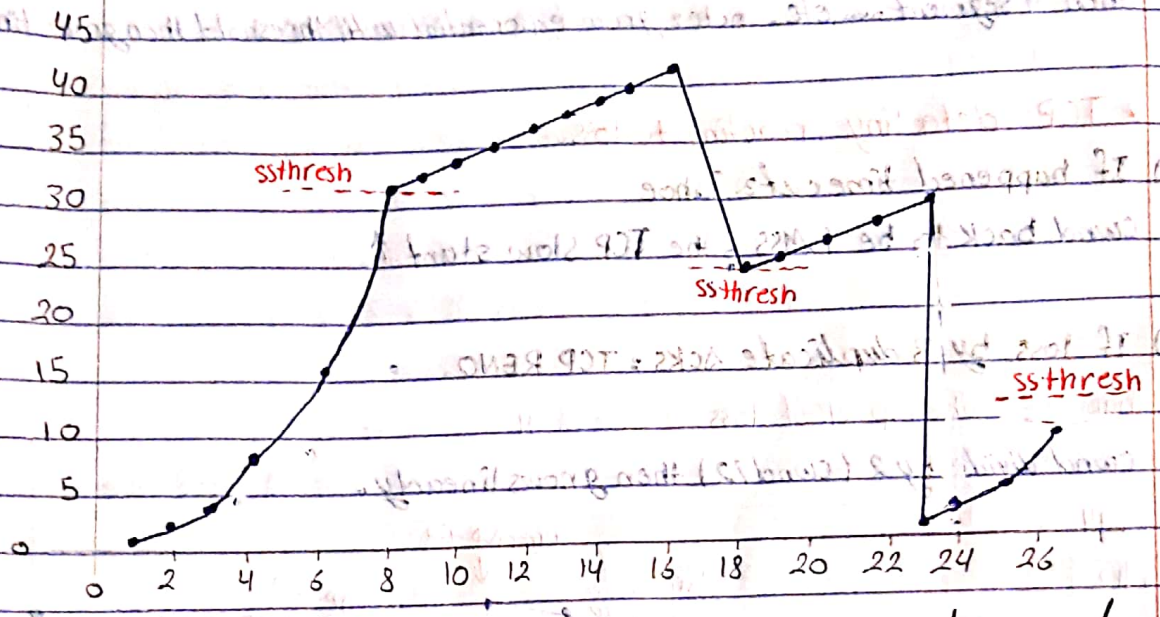
دعاك ال RENO بلش error جديدة بجير ليفير بس هو بيلش من cwnd/2 لما بجير

الايرور ما يرجع للواحد زي Tahoe.

⇒

from Book

P40 :- TCP



a) Identify the intervals of time when TCP slow start is operating.

from 1 to 6 or 8 & 23 to 26.

b) Identify the intervals of time when TCP congestion avoidance is operating.

From 8-16 & 17-22

c) After 16th transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?

RENo (duplicate ACK) since segment not down to 1

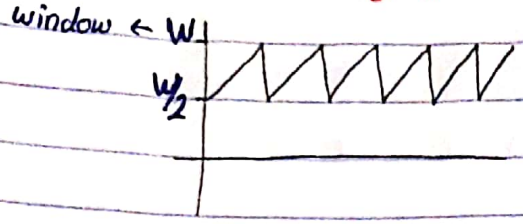
D) After 22nd (like part c).

Taheo (timeout) Back to 1.

ssthresh = 32 & 24 & 12

initially

* TCP throughput :-



avg TCP thruput = $\frac{3}{4}\frac{W}{RTT}$ bytes/sec

ex:- 1500 byte segment, 100ms RTT, want 10 Gbps throughput
requires W = 83,333 segments كبير كبير

* TCP thruput = $\frac{1.22\ MSS}{RTT\sqrt{L}}$

to achieve 10 Gbps throughput, need a loss rate of $L = 2 \times 10^{-10}$ Very Small loss

* TCP Fairness :-

fairness goal:- if K TCP sessions share same bottleneck link of bandwith R, each should have average rate of R/K.

* multimedia apps often use UDP:- send at constant rate, tolerate packet loss. But they don't use TCP: do not want rate throttled by congestion control.

ex:- link of rate R with 9 existing conections:-

A • new app asks for 1 TCP, gets rate :- R/10

B • new app asks for 10 TCP, gets rate:. every connection take $\frac{R}{20}$.

(B) 10 وبعد يه فتحت (A) 1 باقول بعد يه فتحت 9 ← 20 connection على ڤش عندي

$\frac{1}{20}$ يوخذ 10 من كل كونيكشن من 20 بس كل كونيكشن 10 النص لأنهم $\frac{R}{2}$ ← فري (B) كلهي يوخذوا

End chapter 3.