GARRAIO GERUZA
UDP - USER DATAGRAM PROTOCOL
TCP - TRANSMISSION CONTROL PROTOCOL

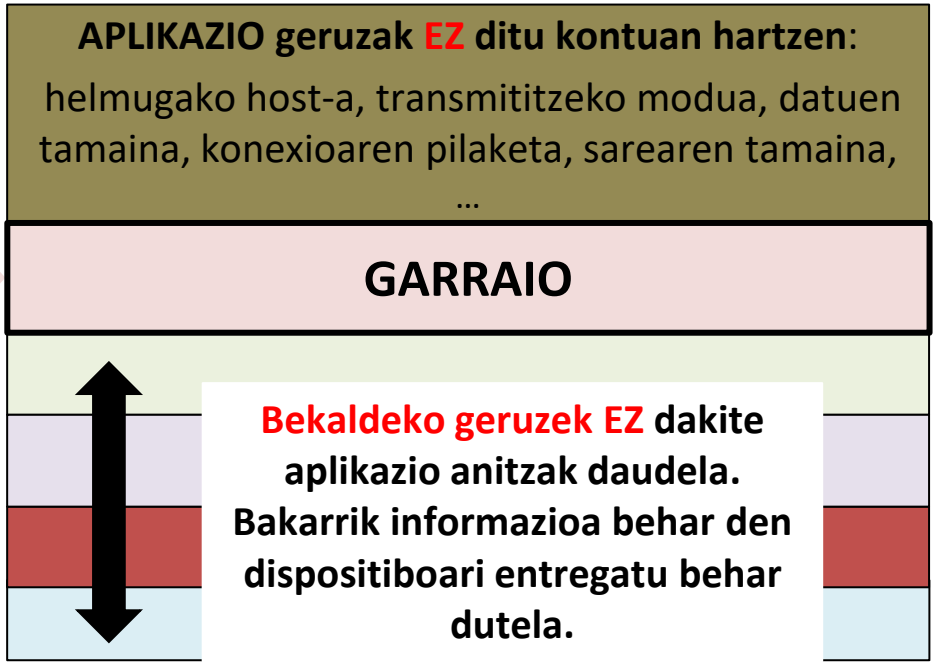
4. Gaia

Gaizka Saldaña
Dpto. Ingeniería de Sistemas y Automática

GARRAIO GERUZA SARRERA

- **GARRAIO** geruzan Aplikazio geruzako datuak onartzen dira eta sare geruzara bidaltzeko prestatzen dira.
- Aplikazioen datuak **ENPAKETATZEN**, **GARRAIOTZEN** eta helmugan dagoen zerbitzuari edo aplikazioari **ENTREGATZEN** dira.

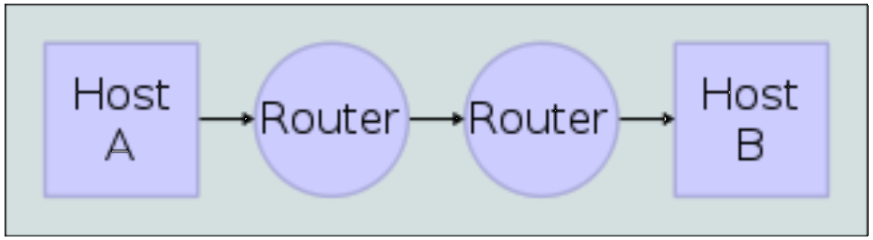
MUTURRETIK-MUTURRERA
Bidaltzeaz arduratzen da (jatorrizko host-etik helmugako host-era)

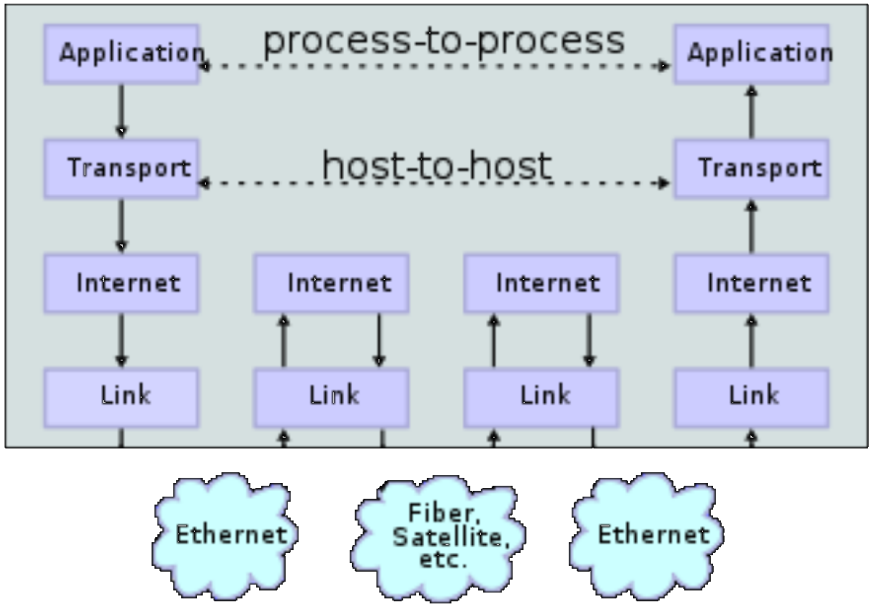


Eskaintzen dituen zerbitzuak izan ahal dira:

- Konexio orientatuak
- Konexiora ez orientatuak

GARRAIO GERUZA SARRERA

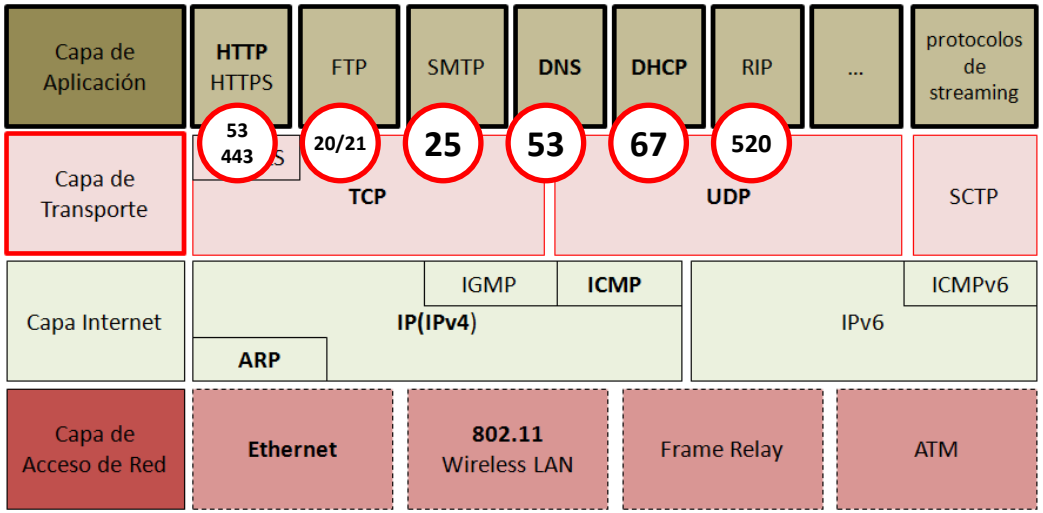




GARRAIO GERUZA FUNTZIOAK

Jatorrizko host batek sarean komunikatzen diren aplikazio anitzak izan ahal ditu, helmugan dauden aplikazio bakarra edo anitzekin.

- **Aplikazioen MULTIPLEXAZIOA**
  - Helmugako hostean paketeak aplikazio zuzenera bideratu behar dira.
  - Garraio geruzaren erresponsabilitatea da aplikazioetan "komunikazio multiple" hau manteintzea.
  - Hau lortzeko, garraio geruzak identifikazio zenbaki bat esleitzen dio aplikazioari, **ATAKA ZENBAKI bakarra host horretan dagoen aplikazioarentzat.**
  - Ataka zenbaki hau garraio geruzaren goiburuan (header) erabiltzen da.



GARRAIO GERUZA FUNTZIOAK

Sareek protokolo datu unitate bakarrean sartu ahal diren datu kantitate limitazioa dute (PDU - Protocol Data Unit)

SEGMENTAZIOA

- Garraio geruzak aplikazio datuak **tamaina onartu batean zatitzen ditu.**
- Garraio geruzako protokoloek datu hauek zatituko dituzten zerbitzuak deskribatuko ditu.
- Prozesu honetan datu segmentu bakoitzean beharrezkoa den **ENKAPSULAZIOA** ere egiten da (informazio osagarri duen goiburua datuak birmuntatzeko).

JATORRIAN

BIRMUNTAKETA

- Segmentuak, helmugan, **birmuntattu** behar dira, aplikazio geruzan datuak erabili ahal izateko.
- Garraio geruzak datuak birmuntatzen ditu aplikazio geruzara bidali baino lehen.
- Garraio geruzako protokoloek goiburuan dagoen informazioa segmentuak birmuntatzeko nola erabili deskribatzen dute.

HELMUGAN

GARRAIO GERUZA FUNTZIOAK

Saretik transmititzen ari den bitartean datu zati bat hondatu edo guztiz galdu daiteke.

APLIKAZIO bakoitzak behar ezberdinak ditu. Adibidez, Datu-baseak, web orriak eta posta elektronikoak bidalitako datu guztiak bere helmugara jatorrizko egoeran iristea behar dute datuak erabilgarriak izan daitezen.
Beste aplikazio batzuk datu kantitate txikien galerarekiko tolerantzia handiagoa dute. (ej. – Bideo Streaming-a)

- Garraio mailako protokolo bat datuen bidaltze fidagarria ziurtatzeko **modu** bat inplementatu **dezake**. Fidagarritasuna lortzeko ekintza basikoak 3 dira:
  - Transmititutako datuen **JARRAIPENA.**
  - Jasotako datuen **HARTZE-agiria** edo baieztapena**.**
  - Hartze-agiririk gabeko edozein datuaren **BIRBIDALTZEA.**
- **Horretarako, jatorrizko garraio geruza** elkarrizketako datu pakete guztien jarraipenaz eta hartze-agiria duen edozein paketearen birbidaltzeaz arduratzen da.
- **Helmugako garraio geruza** paketeak jasotzen diren ahala ere jarraitu behar ditu eta hauen jasotzea antzeman eta pakete errepikatuak detektatu.

➢ **Baliabide gehigarrien erabilera gauzatzen da:**
- Datuen antzemate, jarraipen eta birbidaltzea dela eta.
- **Host igorle eta hartzaileen artean kontrol datu gehiago trukatzen direlako.**

Fidagarritasunaren eta sarean dagoen kargaren konpromisua

GARRAIO GERUZA FUNTZIOAK

Datuak orden okerrean iritsi daitezke, sareak transmisio denbora desberdinak izan ditzaketen bide ugari eskaintzen baitituzte.

- **EMATE ORDENATUA**
  - Garraio mailako protokoloak segmentuak **zenbakitu eta sekuentziatzen** dituztenean **orden egokian birmuntatuko** direla ziurtatu dezakete.

Sarean dauden host-ak baliabide mugatuak dituzte (memoria edo banda zabalera adibidez).

- **FLUXU KONTROLA**
  - Garraio geruzak bere baliabideak sobrekargatuta daudela ohartzen denean, **protokolo batzuk datuak bidaltzen dituen aplikazioari hauen bidaltze abiadura edo fluxua murriztea eskatu diezaiokete.**
  - Fluxu kontrola segmentuen galtzea eragotzi dezake eta birbidaltzearen beharra saihestu.

Funtzio batzuk maila bat baino gehiago egin ditzakete, adibidez fluxu kontrola lotura mailan ere kudeatu daiteke.

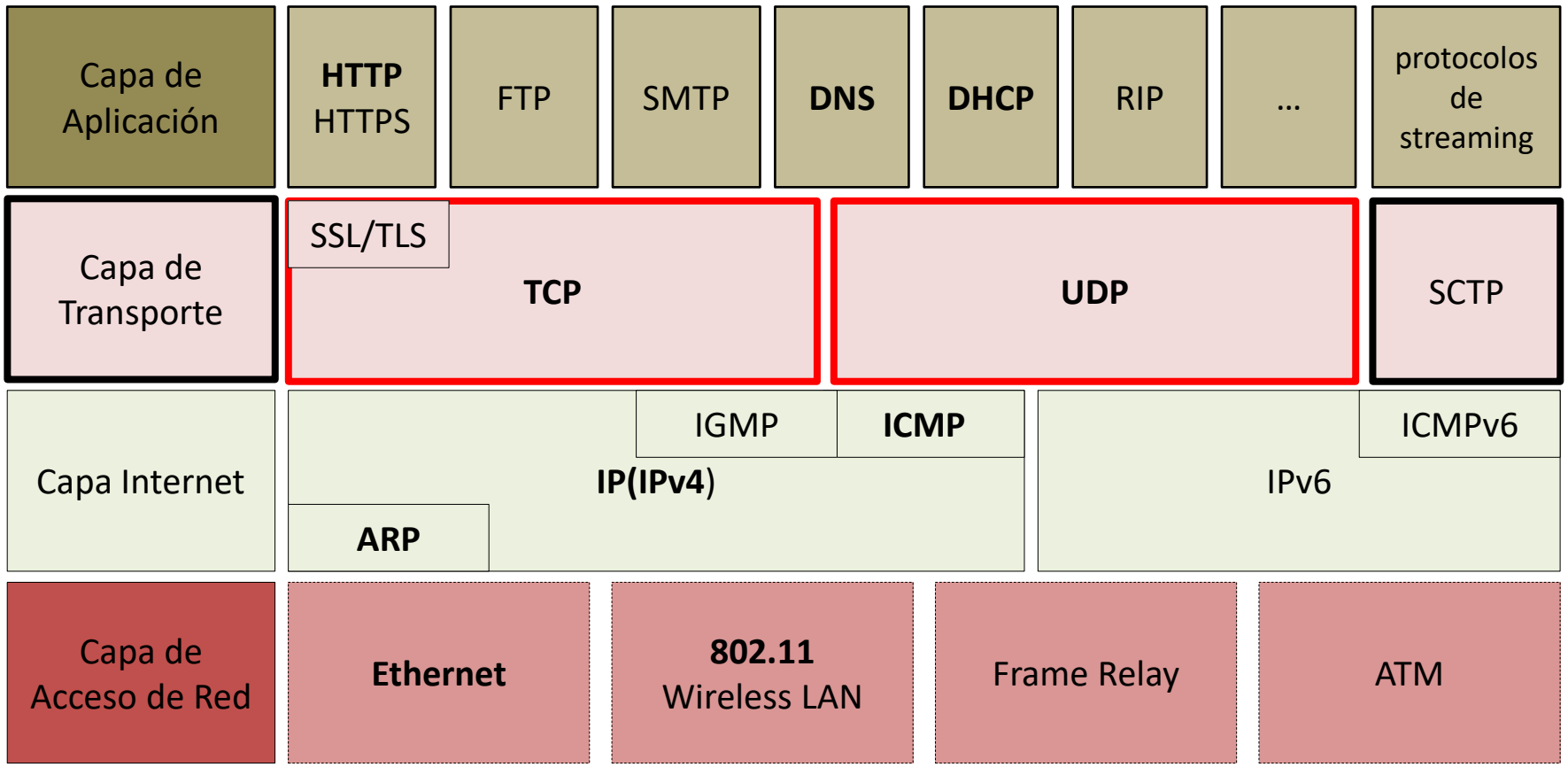
GARRAIO GERUZA
TCP/IP

- APLIKAZIO desberdinak behar desberdinak dituzte:
  - **Segmentuak ordenaturik** iritsi behar dira.
  - **Datu guztiak** iritsi behar dira mezua ulertzeko.
  - Beste aplikazioek datuen **galtzea onartzen** dute.

PROTOKOLO DESBERDINAK

Protokoloak eskainitako funtzionalitatea hainbat eta handiagoa, erabilitako baliabideak eta sarearekiko eskaera handiagoa.

GARRAIO GERUZA
TCP/IP



PROTOKOLOAK
TCP ETA UDP

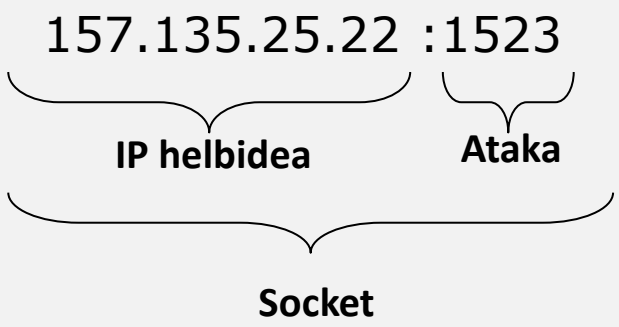
- TCP/IP ereduko garraio geruzako protokolorik garrantzitsuenak:
  - **TCP** (Transmission Control Protocol - Transmisiorako Kontrol Protokoloa)
  - **UDP** (User Datagram Protocol – Erabiltzailearen Datagrama Protokoloa)
- Bi protokoloak aplikazio anitzeko komunikazioa kudeatzen dute.
- Beraien arteko **desberdintasuna bakoitzak inplementatzen** dituen **funtzio espezifikoetan** datza.

TCP Funtzioak	UDP Funtzioak
• Aplikazioen **Multiplexazioa**. • **Segmentazioa (Datu zatiketa)** • **Akatsen** Kontrola. • Fluxu Kontrola. • Itomen edo Kongestio Kontrola. • Galdutako datuen Birbidaltzea. • Konexio/Deskonexioa.	• Aplikazioen **Multiplexazioa**. • **Segmentazioa (Datu zatiketa)** • Akatsen egiaztapena (Aukerazkoa)

TCP / UDP PROTOKOLOA

APLIKAZIOEN MULTIPLEXAZIOA

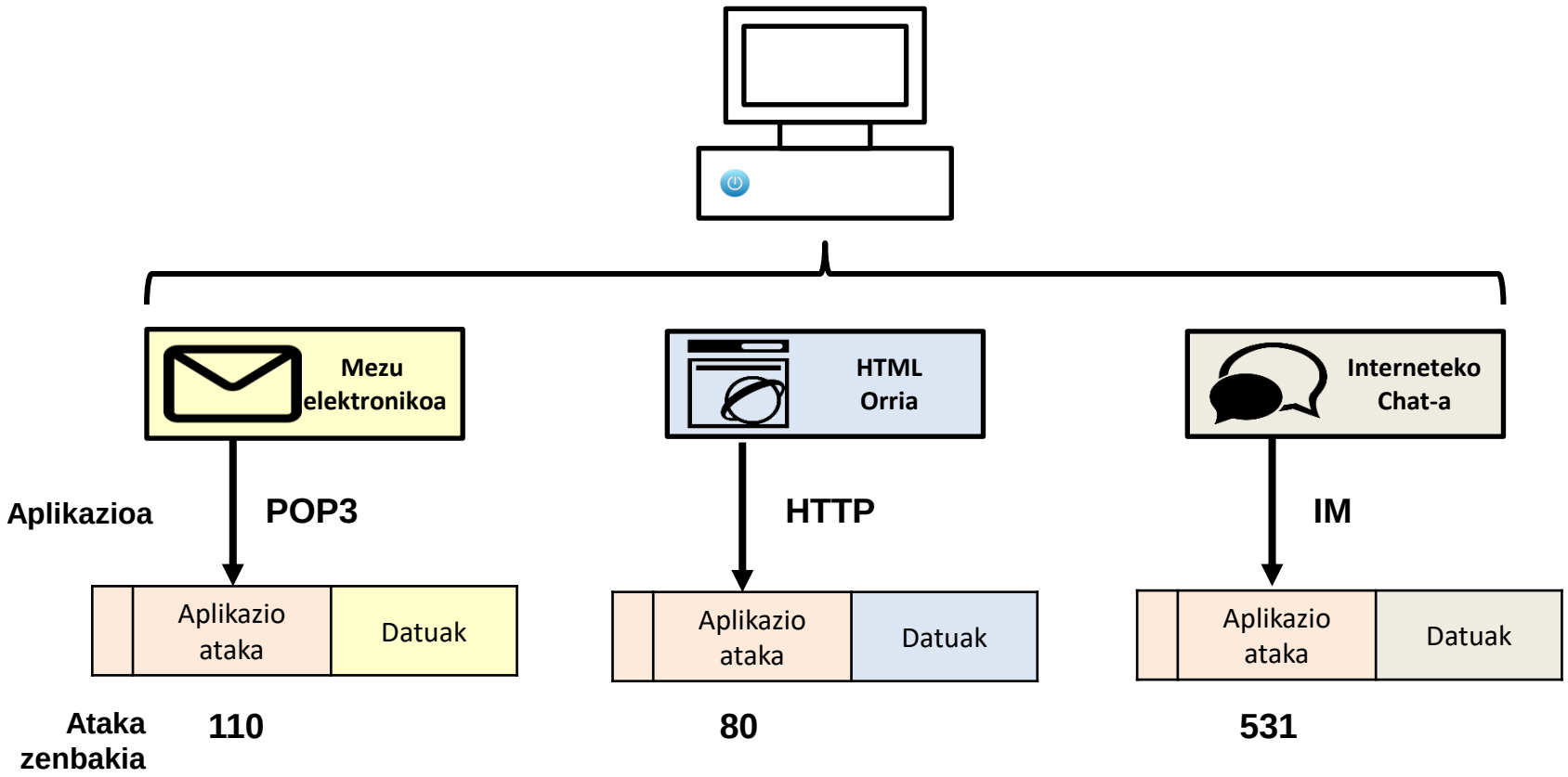
- **TCP eta UDP Komunikatzen diren aplikazioen JARRAIPENA** mantentzen dute**.**
- Aplikazio bakoitzera doazen Segmentuak (TCP) eta Datagramak (UDP) bereizteko, bai TCP eta UDP goiburuan aplikazio hauek identifikatu dezaketen informazioa daukate. Identifikatzaile bakar horiek **PORTU/ATAKA ZENBAKIAK** dira.
- **Multiplexazioa** portuaren bitartez egiten da, paketearen jatorri eta helmugako aplikazioa adierazten duen helbide lokal baten antzekoa da.



Host batean exekutatzen ari den prozesu bat zehazten duen garraio geruzako **ataka zenbakia** eta sareko geruzako **IP helbidearen** arteko konbinazioa da. Konbinazio honi **SOCKET** deritzo.
Socket pare bat (jatorri eta helmuga IP helbideak eta ataka zenbakiak) bi host-en arteko elkarrizketa identifikatzen du.

TCP / UDP PROTOKOLOA

APLIKAZIOEN MULTIPLEXAZIOA



TCP / UDP PROTOKOLOA

APLIKAZIOEN MULTIPLEXAZIOA

- **Ataka Zanbakia**: 16 bit (2 byte). Bere balioa 0 eta 65535 artean egongo da.
- Atakak HIRU TARTEETAN banatzen dira:
  - **0 - 1023** (*well known ports*). Hauek normalean **PROTOKOLO ESTANDARREN ZERBITZARIENTZAT (**HTTP → 80. ataka**)** erreserbatuta daude. Bakarrik super-erabiltzaile moduko baimenak dituzten prozesuak erabili ditzakete.
  - **1024 - 49151: ataka ERREGISTRATUAK**. Edozein aplikazioak erabili ditzake. IANA-ren Web orrian bakoitzak erabiltzen duen protokoloen zerrenda **eskuratu daiteke**.
  - **49152 - 65535:** ataka **DINAMIKOAK EDO PRIBATUAK**. Bakarrik bezeroak erabili ditzakete, ATAKA IRAGANKORRAK ere deitzen dira. Dinamikoki esleitzen dira aplikazio bezero batek konexioa hasten duenean.

Zerbitzua	Ataka	TCP	UDP
DayTime	13		X
FTP	21	X	
SSH	22	X	
TelNet	23	X	
SMTP	25	X	
DNS	53	X	X
BOOTP	67		X
TFTP	69		X
HTTP	80	X	
POP3	110	X	
NTP	123		X
SNMP	161		X
LDAP	389		X
HTTPS	443	X	
SIP	5060		X

Service Name and Transport Protocol Port Number Registry

TCP / UDP PROTOKOLOA

APLIKAZIOEN MULTIPLEXAZIOA

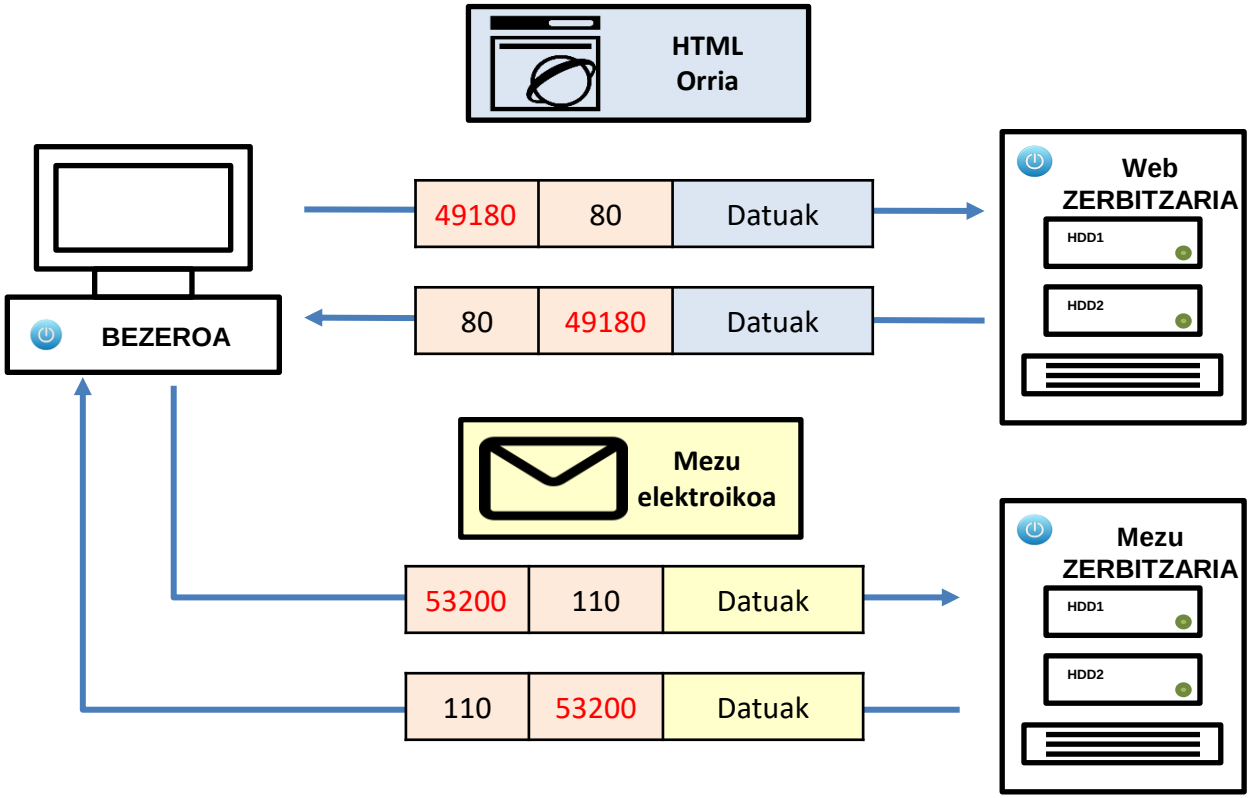
- Garraio eta aplikazio mailako zerbitzuak deskribatzeko BEZERO-ZERBITZARI eredua erabiltzen da:
  - **BEZEROA: konexioa HASTEN duena edo eskakizuna plazaratzen duena**.
  - **ZERBITZARIA: konexio eskaerak jasotzeko zain dagoena**.
  - 49152-tik 65535-ra: ataka dinamiko edo pribatuak. Bakarrik bezeroentzako erabiliak.
- Konexioa bai bezeroak bai zerbitzariak BUKATU dezake.

- **ATAKA ZENBAKIAREN ESLEIPENA:**
  - Esleipen modua aldatzen da, mezua eskaera (bezeroa) edo erantzuna (zerbitzaria) bada.
  - Prozesu **zerbitzariak** ataka **ESTATIKOAK** esleituta duten bitartean, **bezeroak DINAMIKOKI** hautatzen dute portu zenbakia elkarrizketa bakoitzean
  - **Bezeroak** urruneko zerbitzariarian prozesuari dagokion ataka zenbakia ezagutu behar du.
  - **Jatorrizko (Bezeroko) ataka modu ausazkoan sortzen da.**
  - Helmugako garraio geruzak **JARRAIPENA mantentzen du bezeroaren portu horretan,** eskaera gauzatu zuen aplikazioaren portuan, horrela erantzun bat itzultzen denean aplikazio zuzenera eraman ahal izateko. Eskaera egiten duen aplikazioaren portu zenbakia zerbitzariak itzultzen duen erantzunaren helmuga portu zenbaki bezala erabiltzen da.

***netstat** (network statistics) ordenagailuko konexio aktiboen zerrenda, sarrerakoak eta irteerakoak erakusten duen komando-lerroko tresna da*

TCP / UDP PROTOKOLOA

APLIKAZIOEN MULTIPLEXAZIOA



Bezeroaren ataka zenbakia **ausaz** esleitzen da.

Bezeroak prozesuan lotutako ataka zenbakia jakin behar du zerbitzarian

TCP / UDP PROTOKOLOA SEGMENTAZIOA

- **Segmentazioa**
  - Aplikazioaren datuak zatietan **banatzea** hauek transmisio bidearen mugapenen barruan transmitituko direla ziurtatzeko eta bidean ere multiplexatu ahal izango direla ziurtatzeko.
- TCP eta UDP-n segmentazio **DESBERDINA** erabiltzen da.

TCP

TCP-n, goiburu bakoitzak **SEKUENTZIA ZENBAKI** bat dauka. Honek helmugako garraio geruzako funtzioak mezuaren segmentuak transmititu ziren **orden berdinean** birmunta dezaten ahalegiten du.

Helmugako aplikazioa datuak jasotzen ditu igorleak planeatu zuen bezala.

UDP-k diseinu **sinplea** du eta TCP baino **karga gutxiago** sortzen du, hortaz datu transferentzia azkarragoa gauzatzen du.

UDP

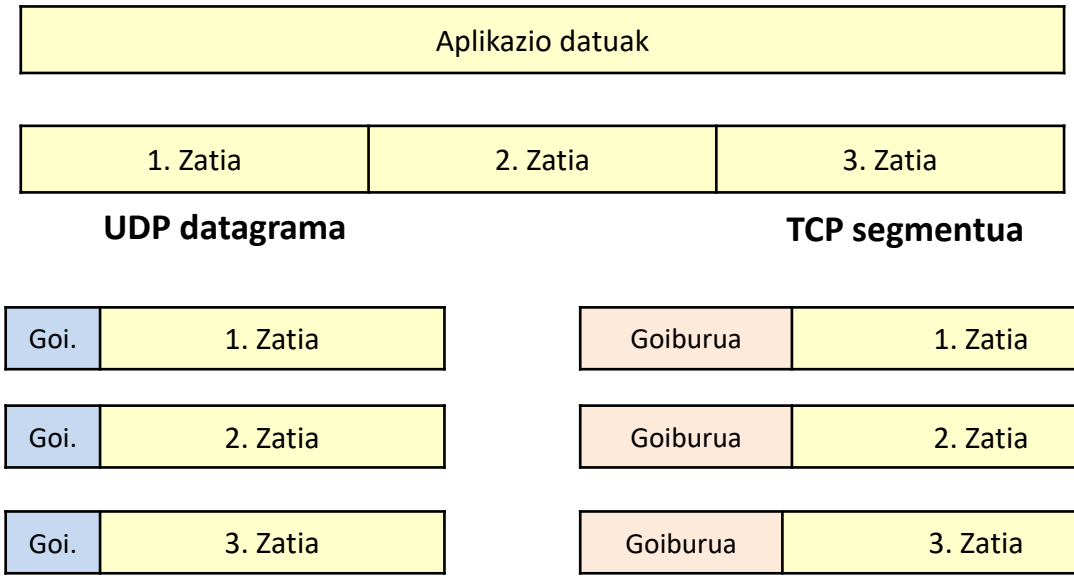
UDP **ez** da informazioa transmititu zen ordenaz edo konexioa mantentzeaz arduratzen.

Ez dago UDP goiburuan sekuentzia zenbakirik.

Informazioa orden desberdinean iritsi daiteke paketeak bide desberdinak hartu ditzaketelako sarean zehar.

UDP erabiltzen duen aplikazio batek datuak bidali ziren orden berdinean ez iristea onartu behar du.

TCP / UDP PROTOKOLOA SEGMENTAZIOA



UDP PROTOKOLOA

- UDP protokolo sinplea da, garraiorako oinarrizko ezaugarriak ditu, hau da egin behar duen gutxien egiten du. RFC 768 azaltzen da.
- **Konexiora ez Orientatua/bideratua. UDP Konexiorik gabeko protokoloa da**, aplikazio batek datuak partekatu behar dituenean zuzenki bidaltzen ditu.
- UDP erabiltzailearen datu unitateak **DATAGRAMA** deitzen dira.
- Datuen tamaina datagrama bat baino handiago bada, hauek datagrama anitzetan **zatitzen** dira.
- Helmugara bidean dauden UDP datagramek ibilbide desberdinak egin ditzakete eta ordenatuta daude. UDP-k **ezin ditu** helmugan **datagramak berrantolatu**, datuak jasotzen dituen ordenan hartzen ditu eta aplikaziora bidaltzen ditu. Datuen ordena aplikaziorako garrantzitsua bada, honek kudeatu egin behar ditu.
- Ez ditu baliabide asko erabiltzen.
- Garraio geruzako protokolo honek datuak " **AHALEGIN HOBERENA**" ("best-effort") bezala bidaltzen ditu.

Ahalik eta hoberen egiten dut baina ahal dudan denbora gutxien galduz.

UDP erabiltzen duten Aplikazioak:

DNS, Bideo Streaming **, Internet protokoloaren gaineko ahotsak** (voIP), Online jokoak…

UDP PROTOKOLOA

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

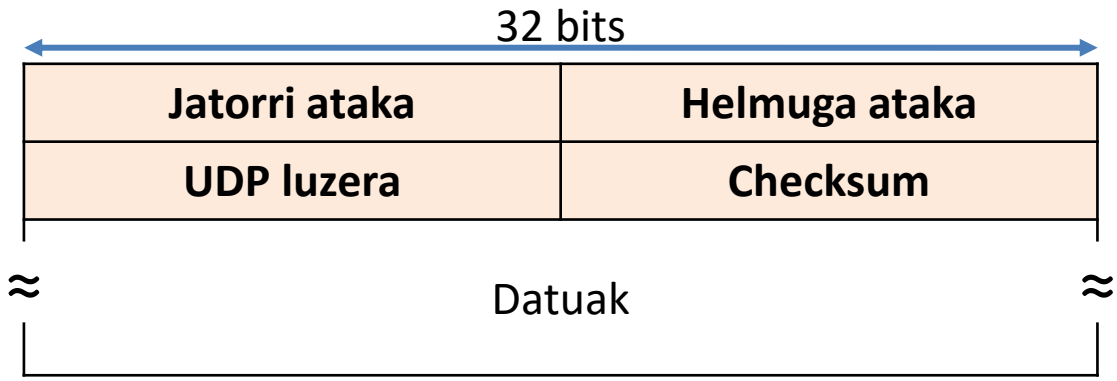
TCP AND UDP

LAYER 4 – TRANSPORT (SEGMENTS)

TCP	**Transmission Control Protocol** • Guaranteed Data Delivery • Error Detection using Sequence and ACK numbers • Windowing • Connection Oriented
UDP	**User Datagram Protocol** • "Best-effort" Delivery, *but no guarantee of delivery* • No Error Detection • No Windowing • Connectionless

UDP PROTOKOLOA

- **DATAGRAMAREN FORMATUA (UDP)**



Jatorri ataka: jatorrizko aplikazioa identifikatzen duen 16 biteko zenbakia (aukerazkoa).

Helmuga ataka: helmugako aplikazioa identifikatzen duen 16 biteko zenbakia.

Luzera: datagrama **osoaren** luzera bytetan (goiburua+datuak).

Checksum: erroreak kontrolatzeko atala. 16 bit. Hautazkoa

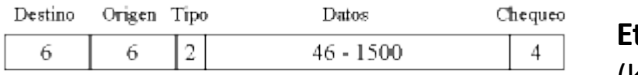
Datuak: Aplikazio geruzako datuak

Zein tamaina izan ahal dute datuek?

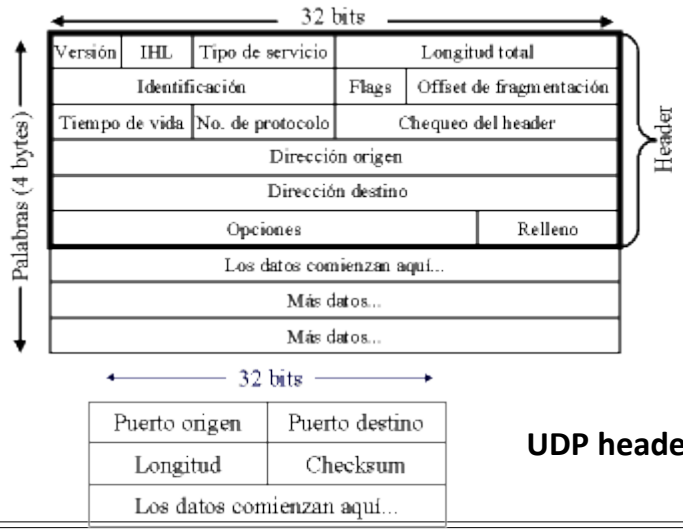
Ariketa: Checksum kalkulua.

UDP CHECKSUM

-
```
     Offset: Hexadecimal                                          ASCII
     ------- -----------                                          -----
      0: 0260 8ce8 5533 0800 2073 5ec6 0800 4500      .`..U3.. s^...E.
     16: 0047 adaa 4000 ff11 784f a8b0 0319 a8b0      .G..@...xO......
     32: 0132 e573 0035 0033 ____ 39f9 0100 0001      .2.s.5.3.|9ù....
     48: 0000 0000 0000 0235 3001 3103 3137 3603      .......50.1.176.
     64: 3136 3807 696e 2d61 6464 7204 6172 7061      168.in-addr.arpa
     80: 0000 0c00 01                                 .....
```



Ethernet frame formatua
(kanpuen tamaina byte-tan da)



IPv4 hader-aren formatua

UDP header-aren formatua

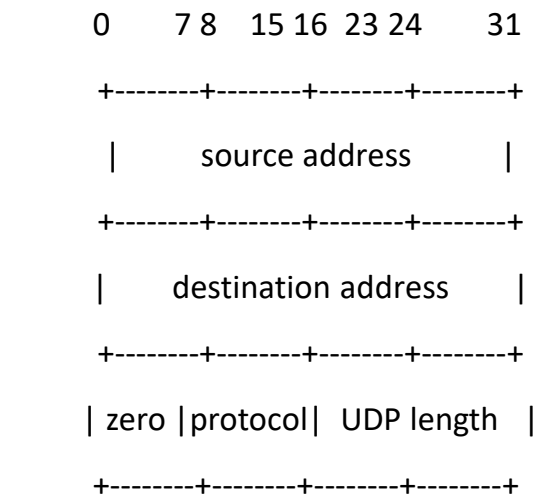
Protokolo zenbakia (hamartarra)	
6	**TCP**
17	**UDP**

UDP CHECKSUM

[...]Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

The pseudo header conceptually prefixed to the UDP header contains the source address, the destination address, the protocol, and the UDP length. This information gives protection against misrouted datagrams.

This checksum procedure is the same as is used in TCP.

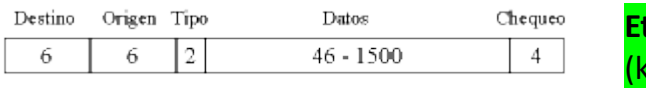


If the computed checksum is zero, it is transmitted as all ones (the equivalent in one's complement arithmetic). An all zero transmitted checksum value means that the transmitter generated no checksum (for debugging or for higher level protocols that don't care).[...]

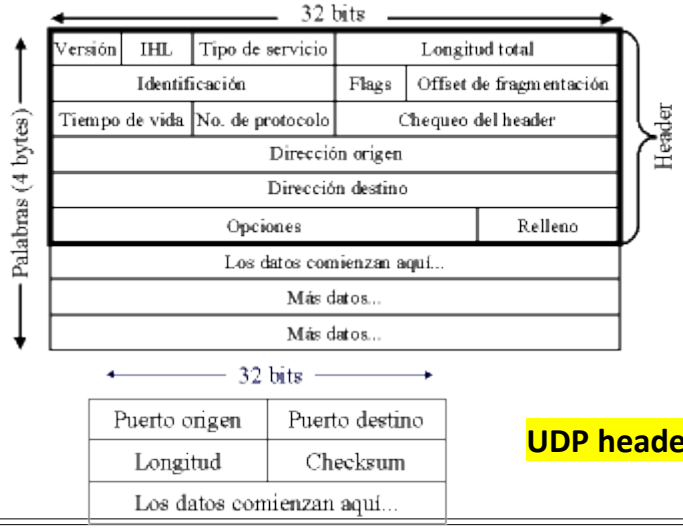
RFC 768

UDP CHECKSUM

```
     Offset: Hexadecimal                                          ASCII
     ------- -----------                                          -----
0:  0260 8ce8 5533 0800 2073 5ec6 0800 4500      .`..U3.. s^...E.
16:  0047 adaa 4000 ff11 784f a8b0 0319 a8b0     .G..@...xO......
32:  0132 e573 0035 0033 ____ 39f9 0100 0001     .2.s.5.3.|9ù....
48:  0000 0000 0000 0235 3001 3103 3137 3603     .......50.1.176.
64:  3136 3807 696e 2d61 6464 7204 6172 7061     168.in-addr.arpa
80:  0000 0c00 01                                .....
```



Ethernet frame formatua
(kanpuen tamaina byte-tan da)

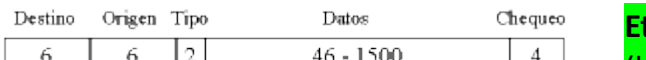


IPv4 hader-aren formatua

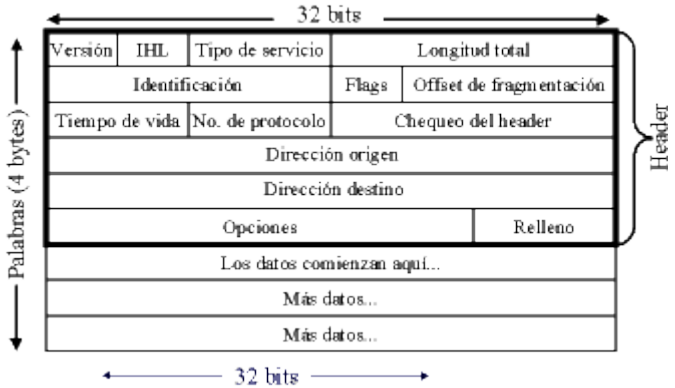
UDP header-aren formatua

UDP CHECKSUM

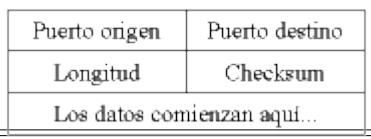
```
     Offset: Hexadecimal                                          ASCII
     ------- -----------                                          -----
0:  0260 8ce8 5533 0800 2073 5ec6 0800 4500      .`..U3.. s^...E.
16:  0047 adaa 4000 ff11 784f a8b0 0319 a8b0     .G..@...xO......
32:  0132 e573 0035 0033 ____ 39f9 0100 0001     .2.s.5.3.|9ù....
48:  0000 0000 0000 0235 3001 3103 3137 3603     .......50.1.176.
64:  3136 3807 696e 2d61 6464 7204 6172 7061     168.in-addr.arpa
80:  0000 0c00 01                                .....
```



Ethernet frame formatua
(kanpuen tamaina byte-tan da)



IPv4 hader-aren formatua



UDP header-aren formatua

seudoheader

a8b0 0319 ===> Jatorri IP helbidea
a8b0 0132 ===> Helmuga IP helbidea
0011 0033 ===> cero + protokolo zenbakia
(17 hamartar) + UDP luzera (51 hamartar, header eta datuak barne)

UDP CHECKSUM

seudoheader

a8b0 0319 ===> Jatorri IP helbidea
a8b0 0132 ===> Helmuga IP helbidea
0011 0033 ===> cero + protokolo zenbakia
(17 hamartar) + UDP luzera (51 hamartar, header eta datuak barne)

Orain batu behar dira, unitatera osagarri (complemento a uno) erabiltzen, seudo-header + UDP header + UDP datuak, 16 bits-eko hitzak bezala

Seudoheader-aren unitatera osagarri batura

[Hex]	Bitar
[a8b0]	1010100010110000
[0319]	0000001100011001
[a8b0]	1010100010110000
[0132]	0000000100110010
[0011]	0000000000010001
[0033]	0000000000110011
[155EF]	10101010111101111

NOTA: Hamaseitarretik hamartarretara:
0x12 = (1 * 16) + 2 = 18
0x3FB = (3 * 16^2) + (15 * 16) + 11 = 1019

Unitatera osagarria erabiltzean, operazio batek acarreo (*carry*) bat sortzen badu bit esanguratsuenean, emaitza gehitu behar da

[55EF]	0101010111101111
	1
[55F0]	0101010111110000

UDP CHECKSUM

UDP header-aren unitatera osagarri batura

Nota: Oraingoan cheksum zelaia 0000000000000000 –tzat hartuko da

[Hex]	Bitar
[e573]	1110010101110011
[0035]	0000000000110101
[0033]	0000000000110011
[0000]	0000000000000000
[E5DB]	1110010111011011

UDP CHECKSUM

UDP datuen unitatera osagarri batura

(Batuketak partzialak erabiliko dira *carry-ak* errazteko)

[Hex]	Bitar
[39f9]	0011100111111001
[0100]	0000000100000000
[0001]	0000000000000001
[0000]	0000000000000000
[0000]	0000000000000000
[0000]	0000000000000000
[0235]	0000001000110101
[3001]	0011000000000001
[3103]	0011000100000011
[3137]	0011000100110111
[3603]	0011011000000011
[1056D]	1000010101101101 -->

0000010101101101 + 1 = 0000010101101110 [56E]

UDP CHECKSUM

```
[3136]   0011000100110110
[3807]   0011100000000111
[696e]   0110100101101110
[2d61]   0010110101100001
[1000C]  1000000000001100  -->

0000000000001100 + 1 = 0000000000001101 [D]
```

```
[6464]   0110010001100100
[7204]   0111001000000100
[6172]   0110000101110010
[137DA]  1001101111011010  -->

0011011111011010 + 1 = 0011011111011011 [37DB]
```

```
[7061]   0111000001100001
[0000]   0000000000000000
[0c00]   0000110000000000
[0100]   0000000100000000
[7D61]   0111110101100001
```

UDP CHECKSUM

Orain header-aren totalak batzen dira

[Hex]	Bitar
[56E]	0000010101101110
[D]	0000000000001101
[37DB]	0011011111011011
[7D61]	0111110101100001
[BAB7]	1011101010110111

Gaizka Saldaña
Dpto. Ingeniería de Sistemas y Automática

UDP CHECKSUM

Eta azkenik, checksum-a kalkulatzen da:

Checksum-a kalklatzeko bi gauza egin behar dira, UDP seudo-header, header eta datuen emaitzak batu, eta azkenik, batera osagarria kalkulatu

1. Seudo-header, UDP header eta datuen batura:

[55F0] 0101010111110000
[E5DB] 1110010111011011
[BAB7] 1011101010110111
[1F682] 1111011010000010 -->

1111011010000010 + 1 = 1111011010000011 [F683]

Batera osagarria lortzeko zeroak bat bihurtu behar dira eta unitateak zero. Era honetan, 1111011010000011 -ren batera osagarria 0000100101111100 da. Au izango da *checksum-a:* 097C hamaseitarra.

Gaizka Saldaña
Dpto. Ingeniería de Sistemas y Automática

UDP CHECKSUM

Frame osoa hurrengoa da:

```
Offset: Hexadecimal                                          ASCII
 ------- -----------                                          -----
 0: 0260 8ce8 5533 0800 2073 5ec6 0800 4500     .`..U3.. s^...E.
16: 0047 adaa 4000 ff11 784f a8b0 0319 a8b0     .G..@...xO......
32: 0132 e573 0035 0033 097c 39f9 0100 0001     .2.s.5.3.|9ù....
48: 0000 0000 0000 0235 3001 3103 3137 3603     .......31.44.66.
64: 3136 3807 696e 2d61 6464 7204 6172 7061     88.in-addr.arpa.
80: 0000 0c00 01                                .....
```

TCP PROTOKOLOA

- **TCP** protokoloa UDP baino **fidagarriagoa eta konplexuagoa** da.
- **RFC 793**–n deskribatua **-** TCP Protocol Specification.
- **Konexiora Orientatutako/zuzendutako** protokoloa.
- TCP paketeak **SEGMENTUAK** deitzen dira (informazioa zatitzen da sarean bidaiatzeko).
- TCP gainontzeko baliabideak erabiltzen ditu hurrengo funtzioak betetzeko:
  - Emate Fidagarria.
  - Sekuentziaren birmuntatzea ordenatuta egiteko
  - Fluxua kontrolatzeko…

TCP segmentu bakoitzan aplikazio mailako datuak enkapsulatzen direnean goiburuan 20 byte ditu, UDP segmentuak bakarrik 8 byte sartzen dituen bitartean.

TCP erabiltzen duten aplikazioak:
Web arakatzaileak, posta elektronikoa, fitxategi transferentzia

TCP PROTOKOLOA

FIDAGARRITASUNA

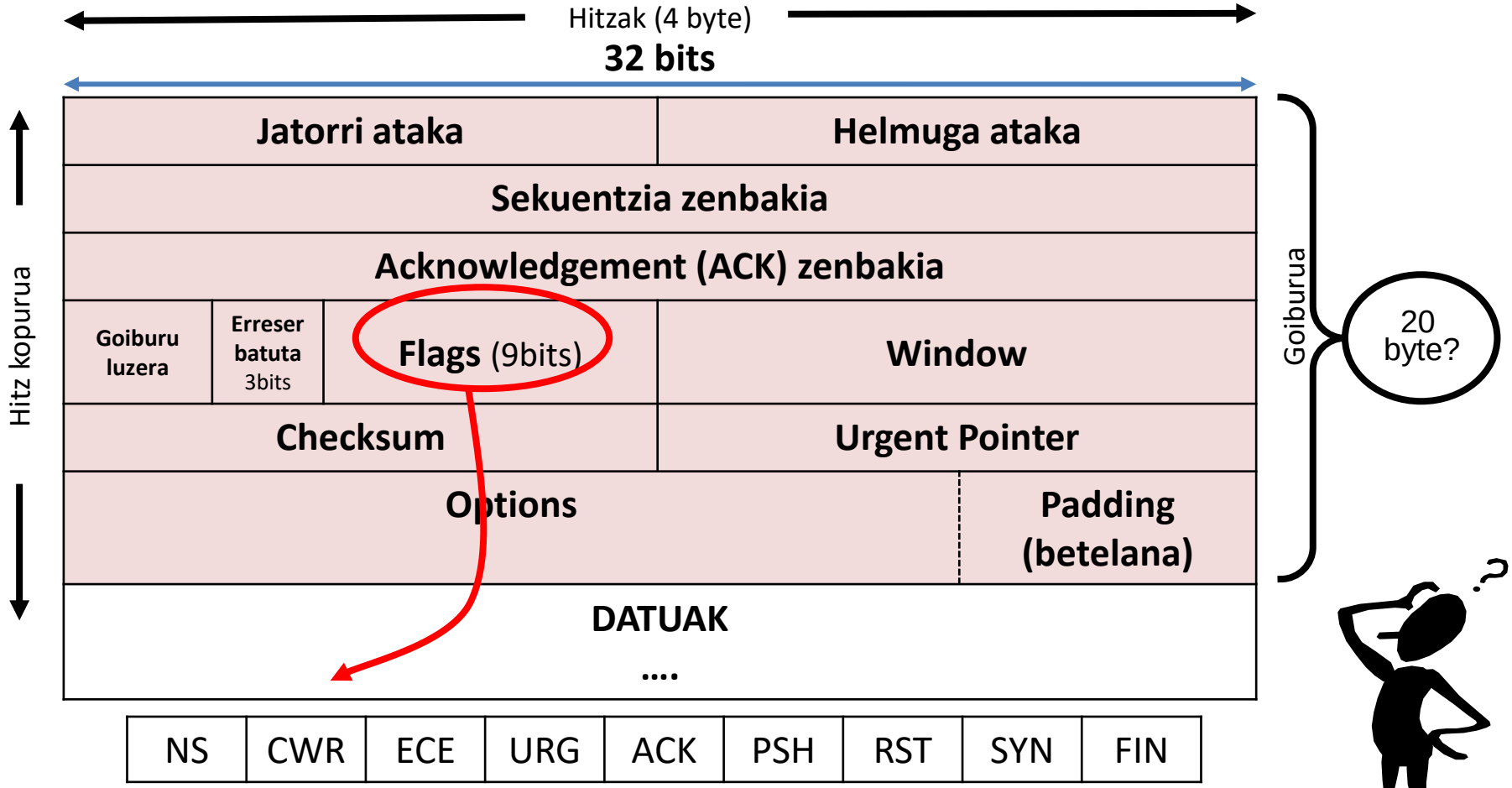
- Behar duen **fidagarritasuna** lortzeko TCP-k erabiltzen ditu:
  - **Konexiora orientatutako** saioak: Transmisio prozesua hasi baino lehen datuen jarraipena ahal egiten duen saioa ezartzen da.
  - Jasotako segmentuen **Hartze-agiriak/baieztapenak**.
  - Datuen **Birbidaltzea,** segmentuak ez badira jaso edota galdutzat eman badira.

TCP-ren gainezko karga/karga gehigarria

- **Sarean:** Saioen ezarpena **gehiegizko segmentuen trukaketa eta hartze-agiriak eta birbidaltzeak direla eta karga handiagoa sortzen du sarean**.
- **Host-etan** ere gainezko karga sortzen da hartze-agiriaren zain dauden segmentuen **erregistroa mantendu behar** delako eta **birbidaltze** prozesuagatik.

TCP PROTOKOLOA

GOIBURUA



¿Zein da TCP goiburu baten gutxienezko eta gehienezko tamaina ? ¿Zein da Options zelaiaren gehienezko tamaina?

TCP PROTOKOLOA
GOIBURUA

NS (1 bit) --***Nonce Sum.*** NS bandera aukerakoa da eta hartzaileak igorlearen segmentuak ezagutzen dituela erakusteko aukera ematen du
CWR (1 bit) -- ***Congestion Window Reduced.*** Bandera ostalari igorleak ezartzen du TCE segmentu bat jaso duela ECE bandarekin jarrita eta pilaketak kontrolatzeko mekanismoarekin erantzun duela adierazteko.
ECE (1 bit) -- ***Explicit Congestion Notification.*** Pilaketen jakinarazpena.
URG (1 bit) -- ***Urgent.*** Premiazko datu bloke bat definitzeko erabiltzen da. Premiazko erakuslearen eremua baliozkoa dela adierazten du. Erakusle honek premiazko datu horiek non amaitzen diren seinalatzen du.
ACK (1 bit) -- ***Acknowledgement.*** Baieztapen bandera
PSH (1 bit) -- ***Push.*** Hartzaileak datuak ahalik eta azkarren pasatu behar dizkio aplikazioari, datu gehiagoren zain egon beharrik izan gabe.
RST (1 bit) -- ***Reset.*** errorea gertatu da eta konexioa itxi beharko litzateke.
SYN (1 bit) -- ***Synchronice.*** Sinkronizatu sekuentzia zenbakiak konexioa hasteko. Konexioaren hasiera adierazten du.
FIN (1 bit) -- Eskatu konexioa askatzeko. Konexioaren amaiera adierazten du.

NS	CWR	ECE	URG	**ACK**	**PSH**	**RST**	**SYN**	**FIN**

TCP PROTOKOLOA

IZENBURUA

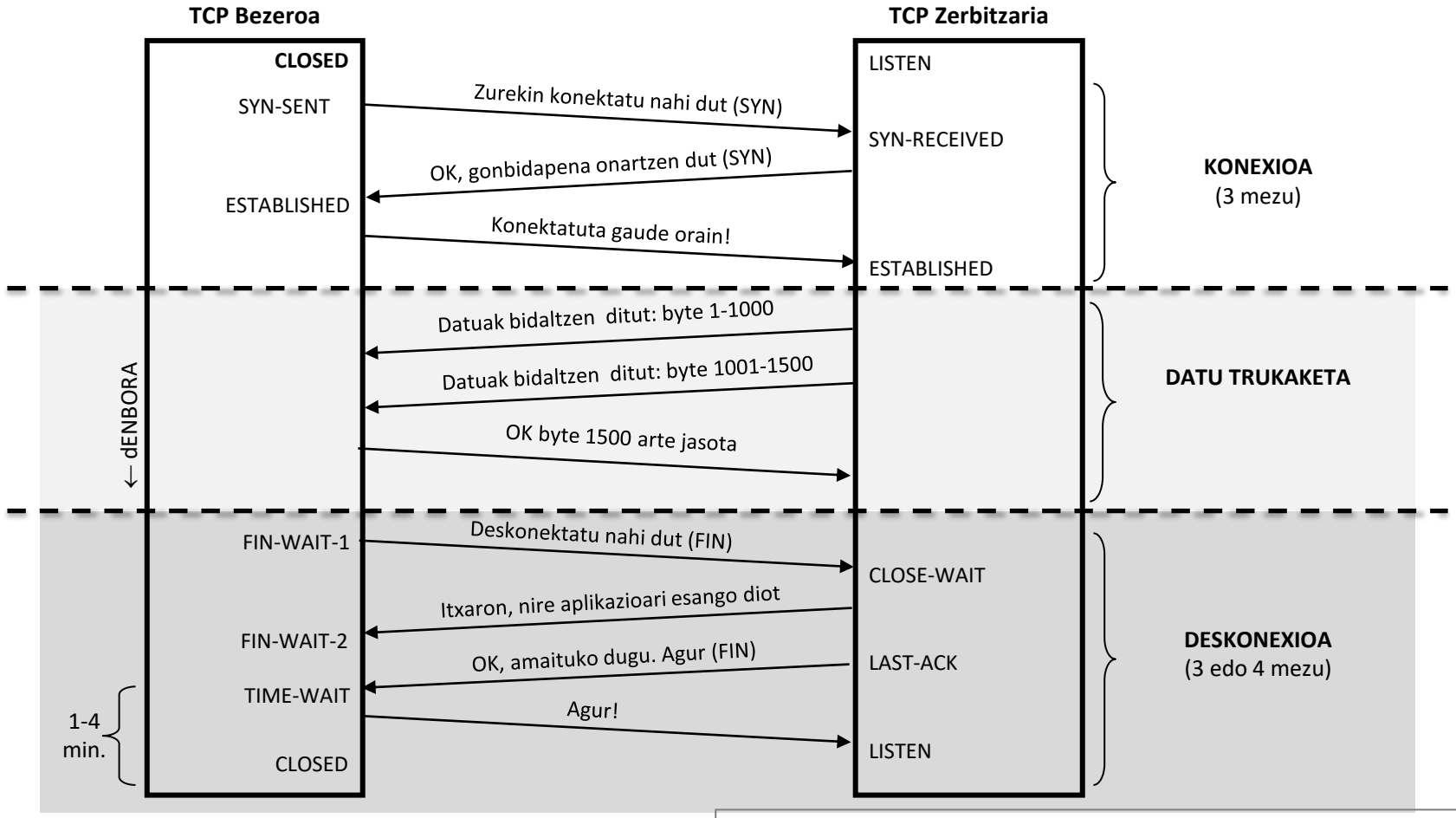
Aukeren eremua (Options): hiru zati, lehenengoa aukeraren eremuaren luzera, bigarrena erabiltzen den aukera eta hirugarrena erabiltzen diren aukerak. Gehien erabiltzen den aukeretako bat MSS da (segmentu gehieneko tamaina).

TCP Option Kind Numbers
(https://www.iana.org/assignments/tcp-parameters/tcp-parameters.xhtml#tcp-parameters-1)

Kind	Length	Meaning	Reference
0	-	End of Option List	[RFC793]
1	-	No-Operation	[RFC793]
2	4	Maximum Segment Size	[RFC793]
3	3	Window Scale	[RFC7323]
4	2	SACK Permitted	[RFC2018]
5	N	SACK	[RFC2018]
6	6	Echo (obsoleted by option 8)	[RFC1072][RFC6247]
7	6	Echo Reply (obsoleted by option 8)	[RFC1072][RFC6247]
8	10	Timestamps	[RFC7323]
9	2	Partial Order Connection Permitted (obsolete)	[RFC1693][RFC6247]
10	3	Partial Order Service Profile (obsolete)	[RFC1693][RFC6247]
11		CC (obsolete)	[RFC1644][RFC6247]
12		CC.NEW (obsolete)	[RFC1644][RFC6247]
13		CC.ECHO (obsolete)	[RFC1644][RFC6247]
14	3	TCP Alternate Checksum Request (obsolete)	[RFC1146][RFC6247]
15	N	TCP Alternate Checksum Data (obsolete)	[RFC1146][RFC6247]
16		Skeeter	[Stev_Knowles]
17		Bubba	[Stev_Knowles]
…	…	…	…

TCP PROTOKOLOA

KONEXIOARI BEGIRA SAIOAK



TCP Saioa

Flag **SYN= 1→** konexio baten **hasiera adierazten du**.
Flag **FIN =1→ konexio baten amaieraadierazten du**

TCP PROTOKOLOA
KONEXIOAREN EZARTZEA

- **THREE-WAY HANDSHAKE – "HIRU BIDEKO AGURRA"**
  - konexioa ezartzeko TCP protokoloan gehien erabiltzen den mekanismoa da. 3 mezuen trukaketan oinarrituta dago, horregatik izen hori ematen zaio:

Segmentu 1: (bezeroa) → Bezeroak zerbitzariari konektatzeko gonbidapen bat bidaltzen dio. '**active open**' bat egiten duela esaten da.

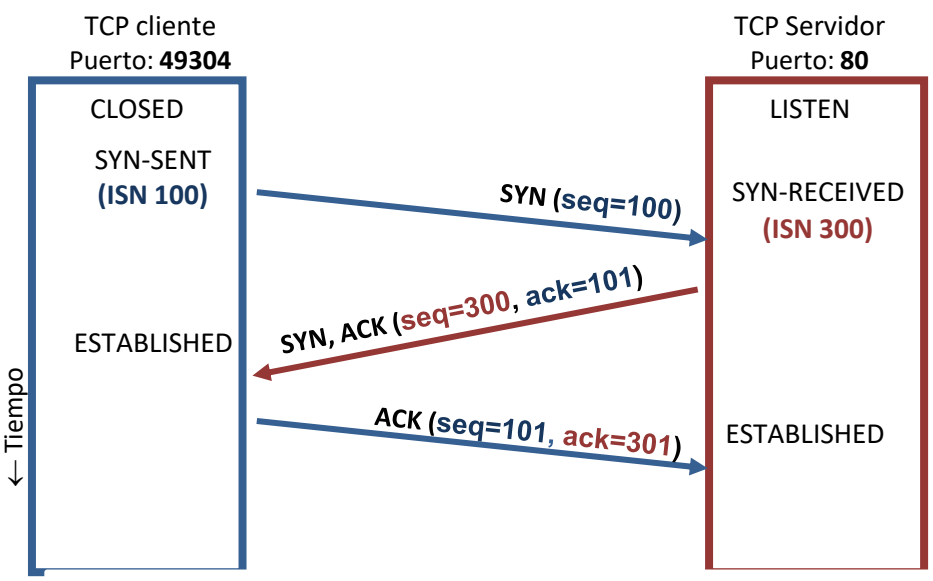
Segmentu 2: (zebitzaria) → Zerbitzariak gonbidapena jasotzen duenean bezeroari onartze mezu bat bidaltzen dio erantzun moduan. '**passive open**' bat egiten duela esaten da.

Segmentu 3: (bezeroa) → Erantzuna **jasotzen duenean bezeroak** konexioa ezarritzat hartzen du eta hirugarren mezu bat bidaltzen du, aurrekoaren **hartze-agiria.**

Zerbitzariak **ezarritzat hartzen du konexioa** hirugarren mezu hori **jasotzen duenean.**

TCP PROTOKOLOA

KONEXIOAREN EZARTZEA



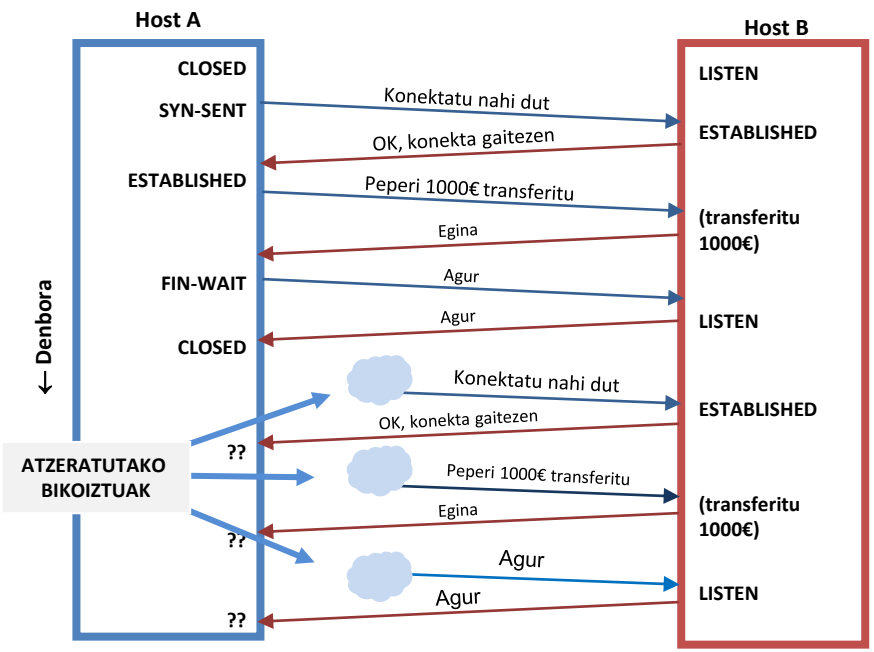
- TCP-k erabiltzen duen konexio mekanismoa hiru segmenturen trukean oinarritzen da, horregatik hiru norabidetako esku-emate gisa ezagutzen da **(three way handshake).**
  - **Bezeroak** zerbitzarira konektatzeko gonbidapena bidaltzen du. "active open" egiten duela esaten dugu. SYN flag = 1.
  - Gonbidapena jasotzen duenean, zerbitzariak bezeroari onartzen dion erantzuna emango dio. "passive open" egiten du. SYN flag = 1 eta ACK = 1.
  - Erantzuna jasotzean, **bezeroak** konexioa ezarrita dagoela uste du eta hirugarren aitorpen eta jasotze mezu bat bidaltzen du. ACK = 1. **Zerbitzariak** hirugarren mezu hau jasotzen duenean konexioa wzarritzat emten du.

- TCP konexio bat ezarri behar denean identifikatzaile bat aukeratzen da konexio horretarako (**ISN** - *Initial Sequence Number*).
- TCP konexio batean bakarrik 2 nodo parte hartzen dute. Bakoitzak bere kabuz konexiorako erabiliko duen ISN-a aukeratzen du, beraz **konexio baterako beti 2 ISN** daude, bat konexioko eremu bakoitzerako.
- ISN-a aurretik ezarritako konexio baten **atzeratutako bikoizturen** bat iristea ekiditen du, honek konexio engainakorra gauzatuko luke.

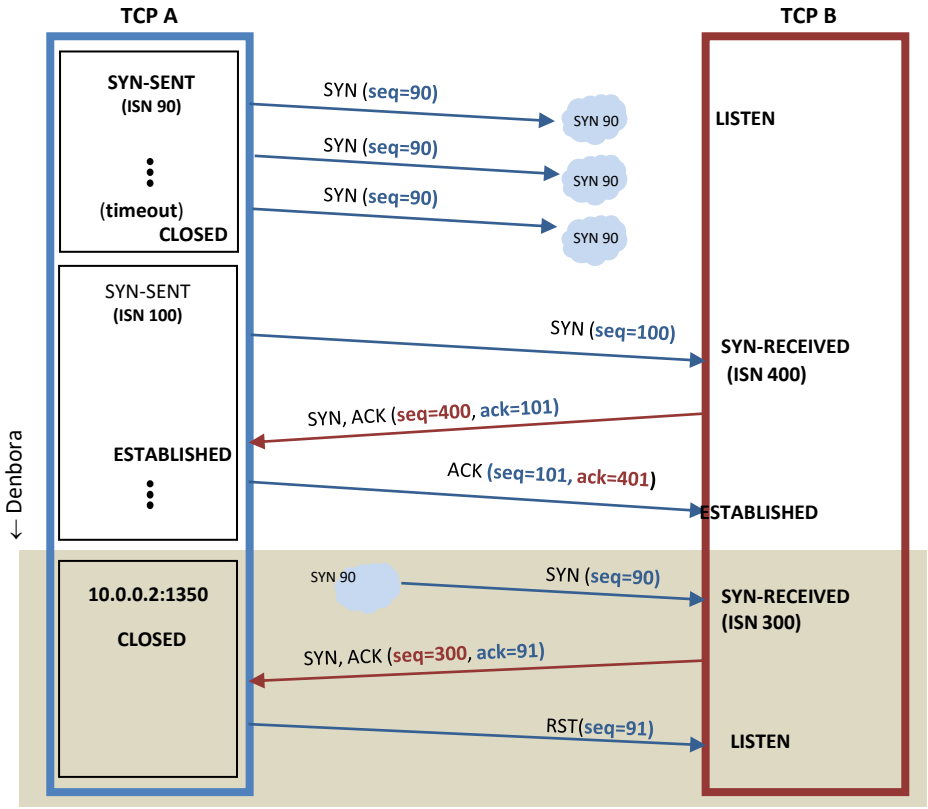
ISN-ren HAUTAKETA (RFC 793-ren arabera). 32 bit-eko zenbaki oso bat izan behar du, **zeinurik gabekoa eta 4 mikrosegundora unitate batean inkrementatzen dena.** ISN bat 4 ordutara agertu daiteke, **denbora nahikoa ISN berdina duen edozein atzeratutako bikoizturen agerpena ekiditzeko**. Sistema eragileak algoritmo errazak erabiltzen dituzte ISN-ak eraikitzeko, **CPU gutxiago kontsumitzeko asmoz.**

TCP PROTOKOLOA

KONEXIOAREN EZARTZEA



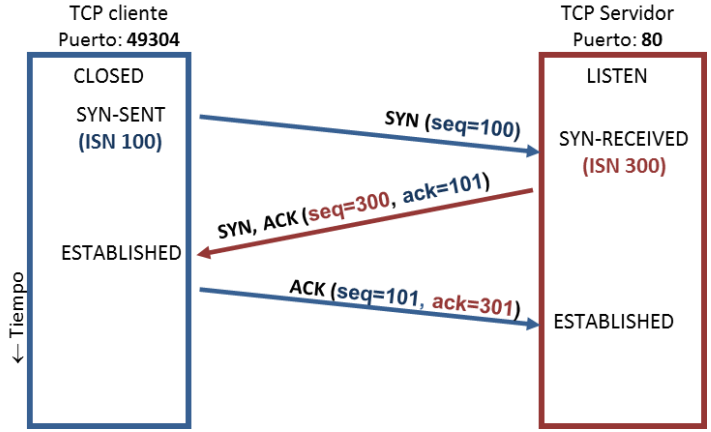
Soluzioa: Atzeratutako SYN baten agerpena



RST: Konexioa amaitzen du erantzunik itxaron gabe.

TCP PROTOKOLOA

DATU TRUKAKETA

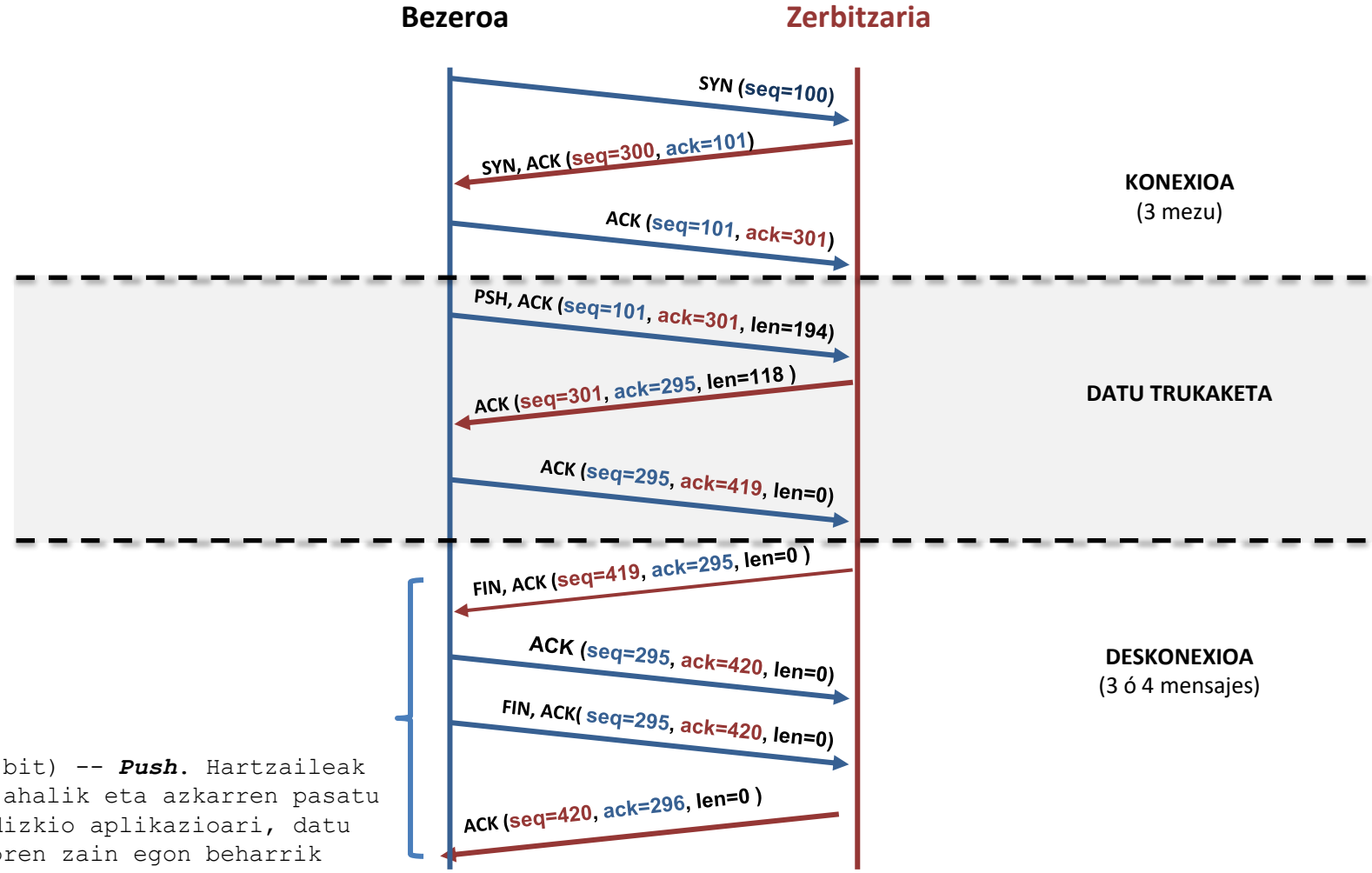


- **Zekuentzia zenbakiak eta SYN flag**
  - TCP-k konexio batean transmititutako byte guztiak **ZENBATZEN** ditu.
  - **SEKUENTZIA ZENBAKIA (SEQ)** atala bidalitako datuen **lehenengo byte-a** identifikatzen du. TCP ez ditu segmentuak zenbatzek, byte-ak baizik.
  - **ISN = hasierako balioa**, ikusi dugun bezala TCP protokoloa aukeratzen du SYN bidaltzen duenean, konexioaren **identifikatzaile** bezala erabiltzen da hiru-bideko agurra konexioan. Hortaz, sekuentzia zenbakia datuen lehenengo byte identifikatzen du.
  - **Flag SYN** jarrita dagoenean sekuentzia zenbakia unitate 1-ean inkrementatzen da. SYN segmentua datu “birtual” bat bezala ikusi dezakegu.’
- **Hartze-agiriak eta ACK flag**
  - Hartze-agiriek beste TCP-ri datuak zuzen jaso direla adierazteko balio dute.
  - **Flag ACK** ‘ACK zenbakia’ atalean dagoena zentzuduna dela adierazten du.
  - **ACK ZENBAKIAK** hurrengo TCP segmentuan jasotzea espero den lehen byte-a zehazten du. Hau da, ACK -1 jaso eta **antzemandako azken byte-ren zenbakia adierazten du.**
  - TCP-k **piggybacking** (itzuliko den trafikoa-mezuaz laguntzen da informazio antzematea egiteko) deritzon teknika erabiltzen du datuen antzematea egiteko. Teknika honen bidez hartze-agiri bat trama bakar batean bidali ordez (ACK mezua soilik) bidaliko den hurrengo paketearen amaieran jarriko du informazioa.
  - TCP trukatzen diren mezu guztiek, lehenengoa izan ezik, ACK Flag jarrita dute..
  - **ACK** banderaren presentziak **EZ du sekuentzia kopurua handitzen**.

Konexio bakoitza norabide bakarreko bi saio dira.
Sekuentzia zenbakiak eta aitorpen zenbakiak bi noranzkoetan trukatzen dira.

TCP PROTOKOLOA

DATU TRUKAKETA

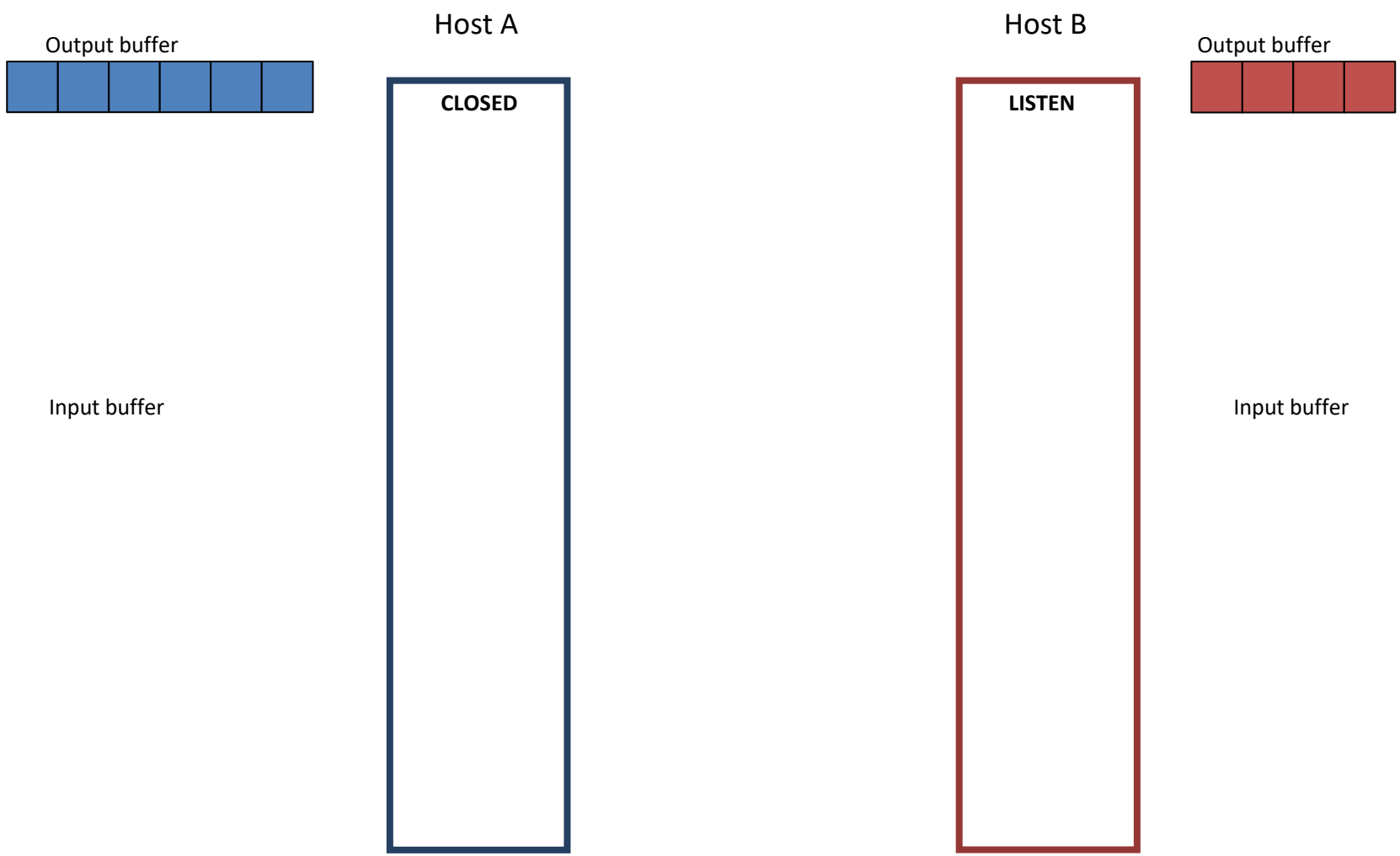


PSH (1 bit) -- ***Push***. Hartzaileak datuak ahalik eta azkarren pasatu behar dizkio aplikazioari, datu gehiagoren zain egon beharrik izan gabe.

Gaizka Saldaña
Dpto. Ingeniería de Sistemas y Automática

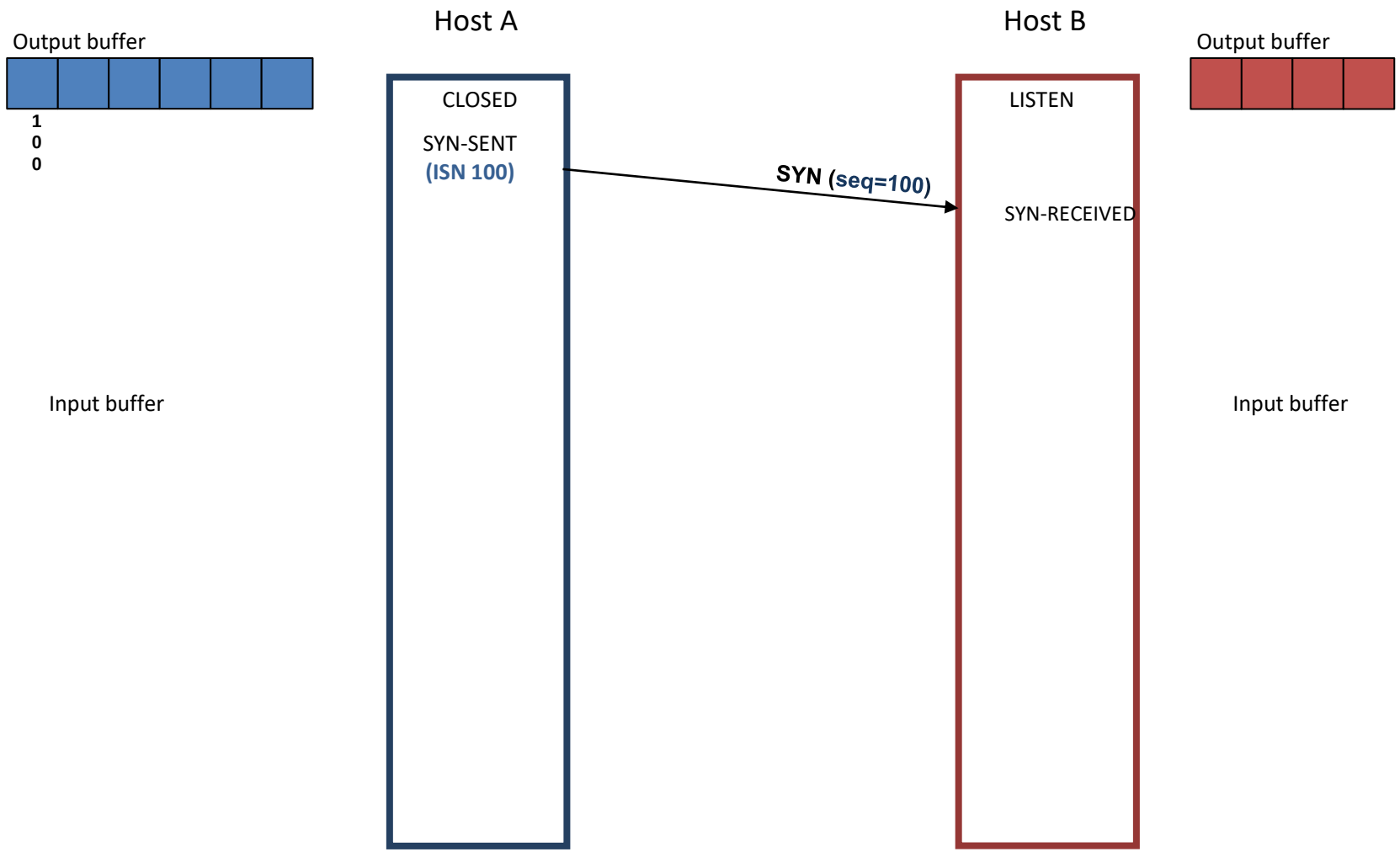
TCP PROTOKOLOA

DATU TRUKAKETA



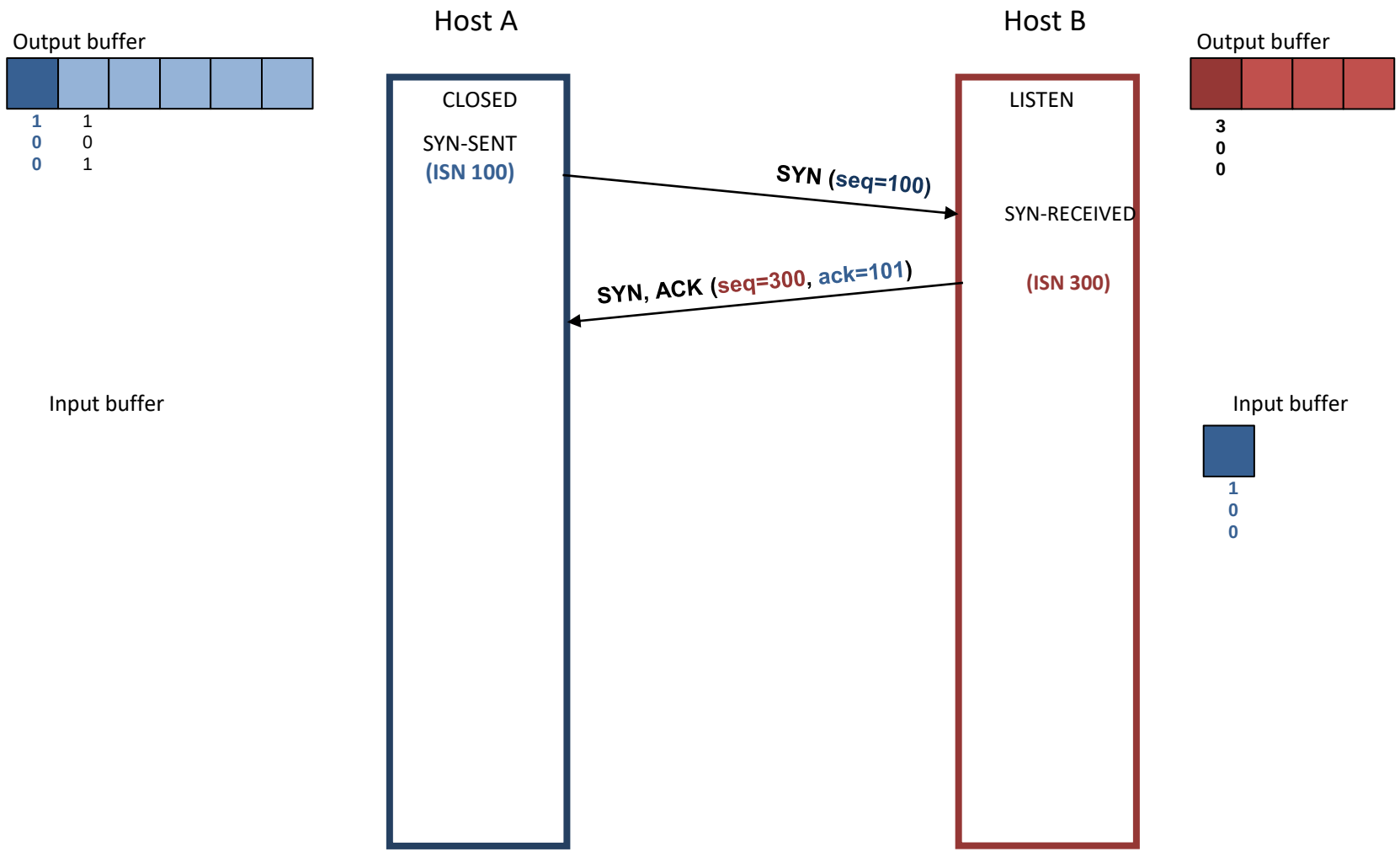
TCP PROTOKOLOA

DATU TRUKAKETA



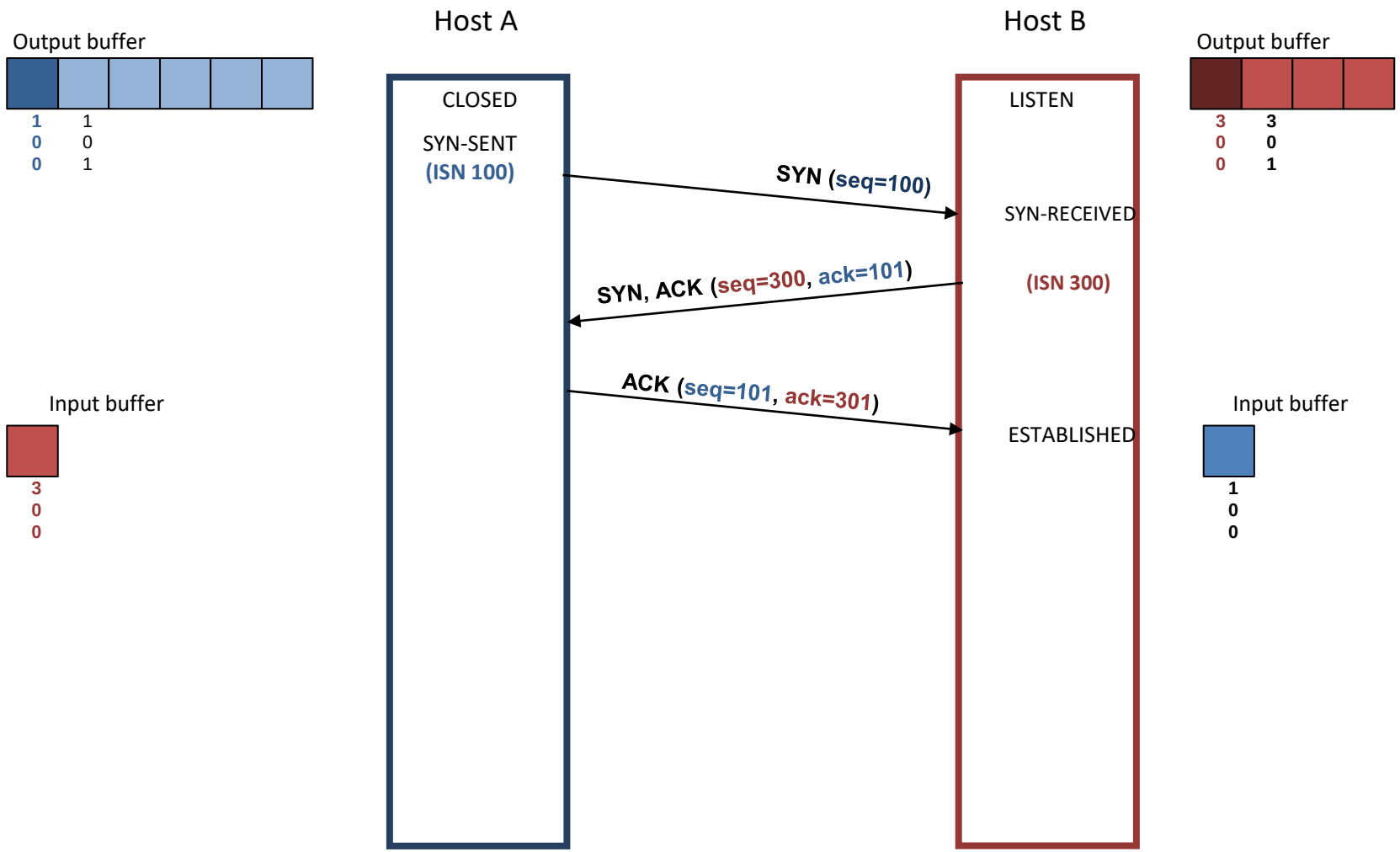
TCP PROTOKOLOA

DATU TRUKAKETA



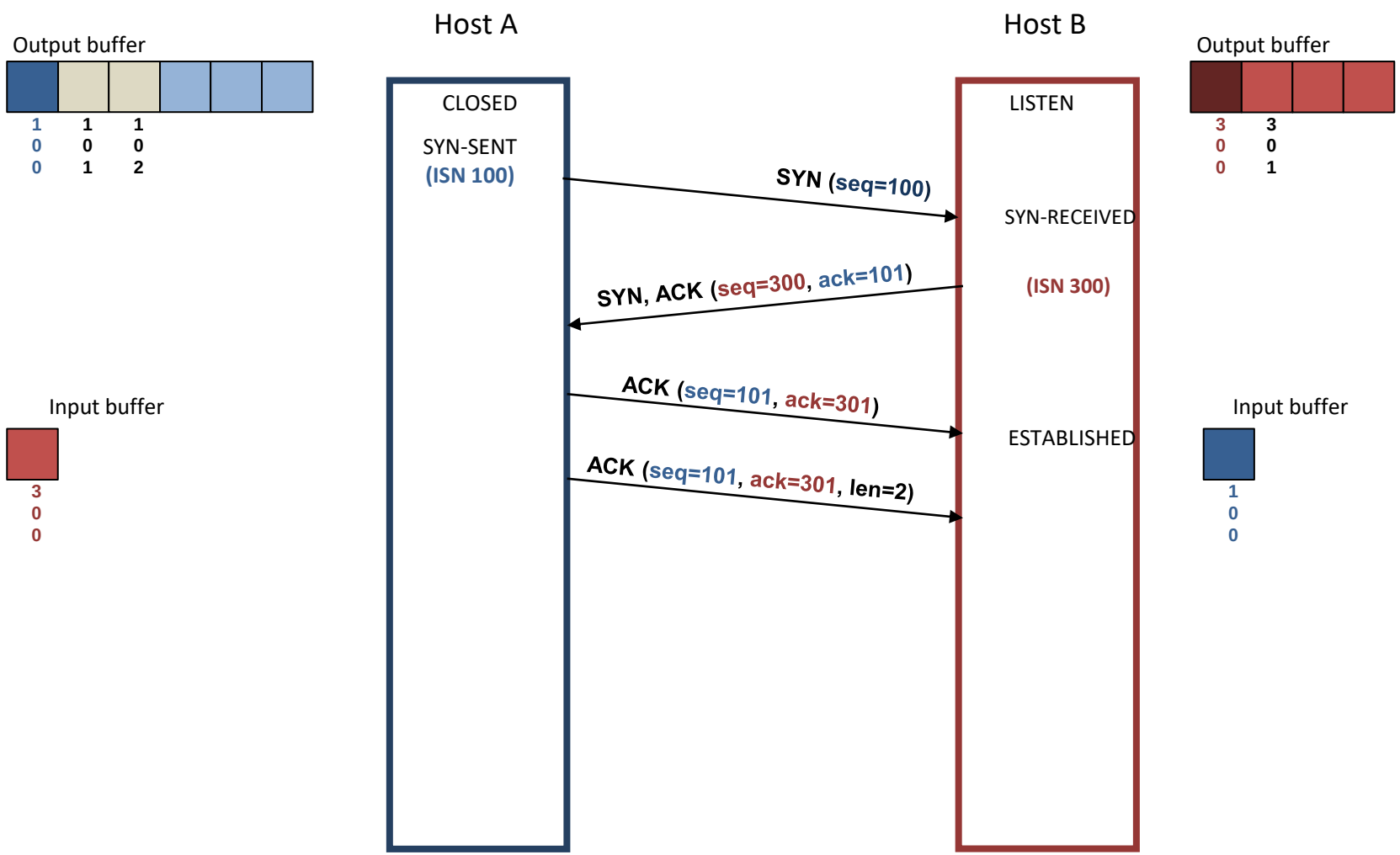
TCP PROTOKOLOA

DATU TRUKAKETA



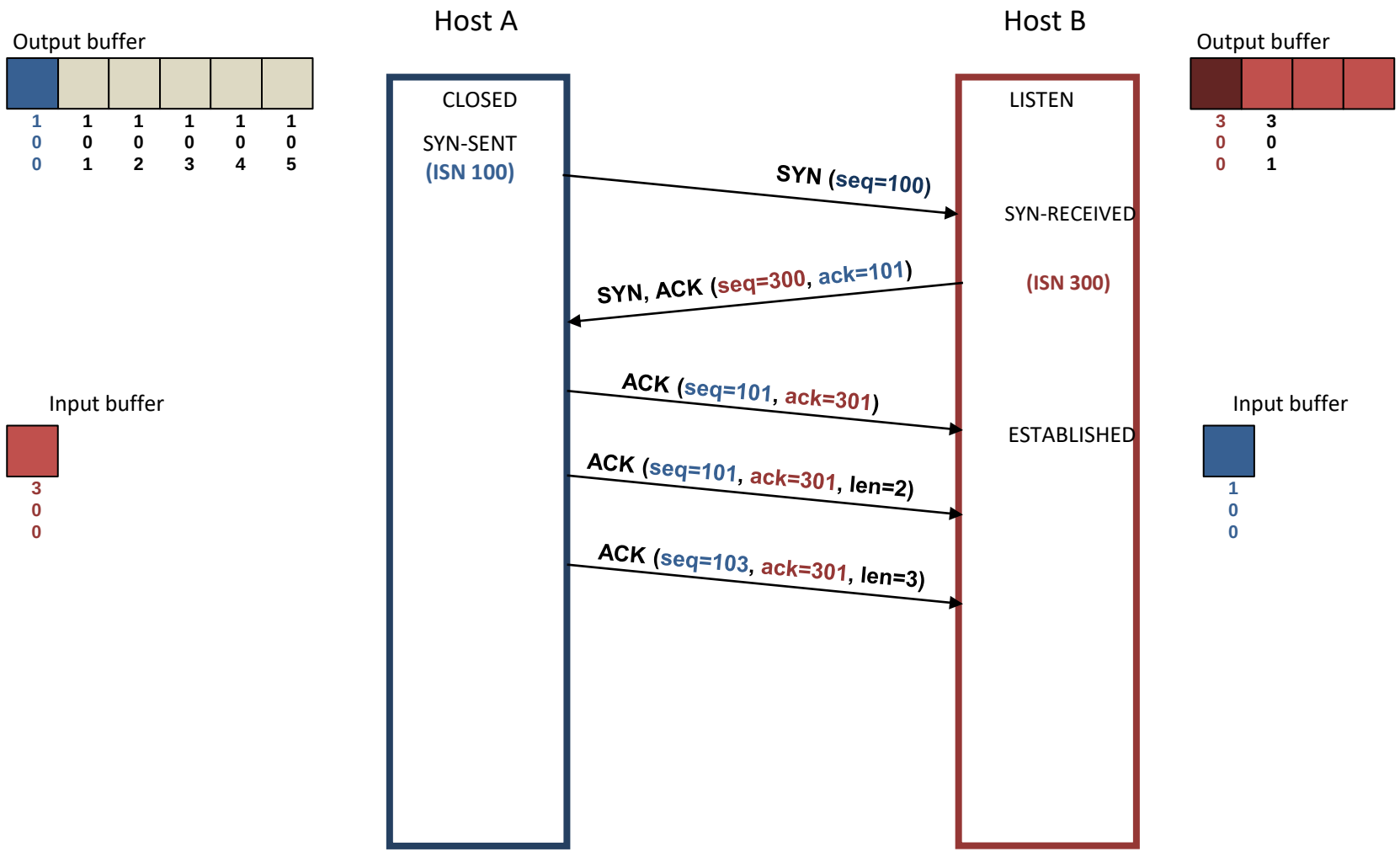
TCP PROTOKOLOA

DATU TRUKAKETA



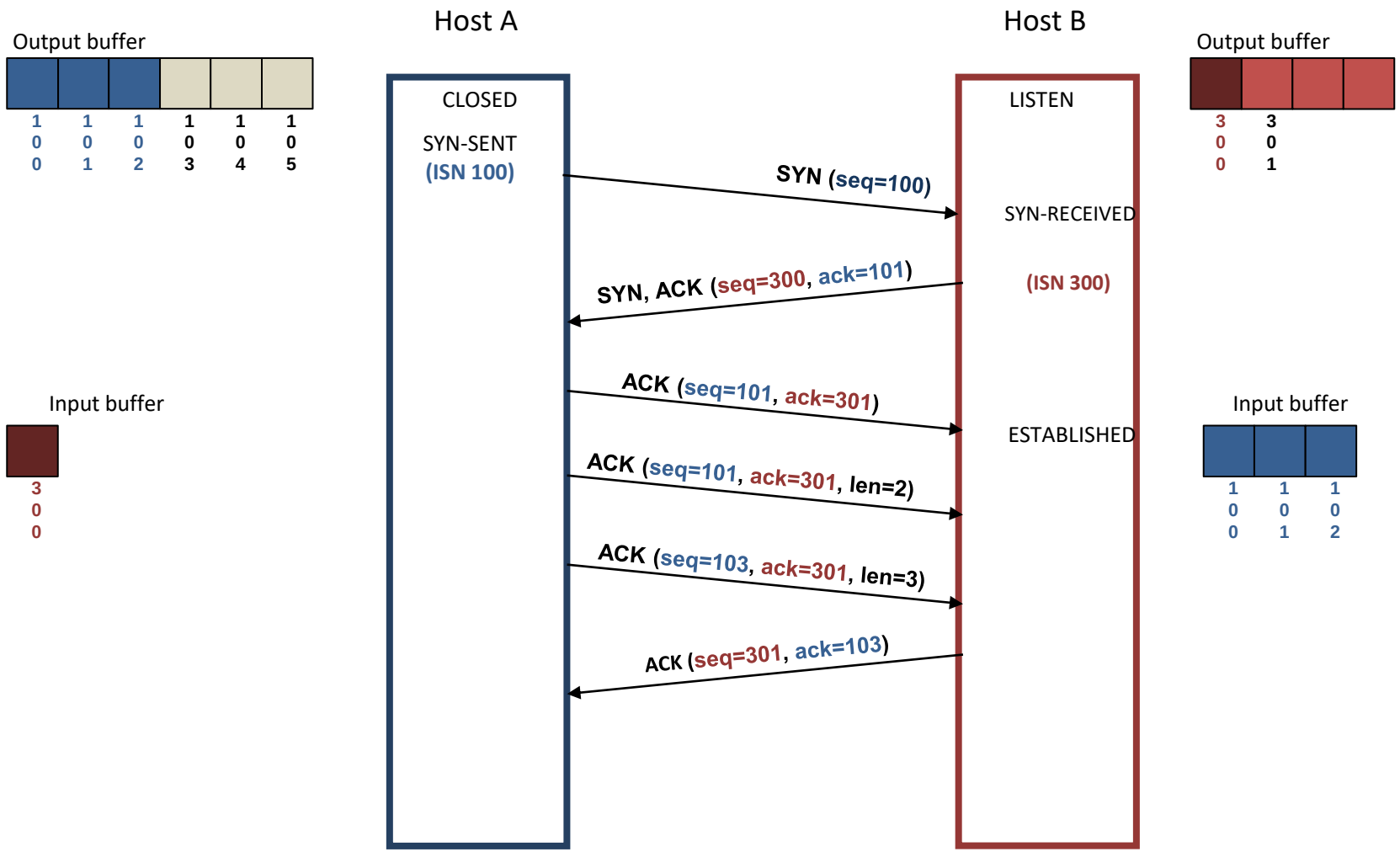
TCP PROTOKOLOA

DATU TRUKAKETA



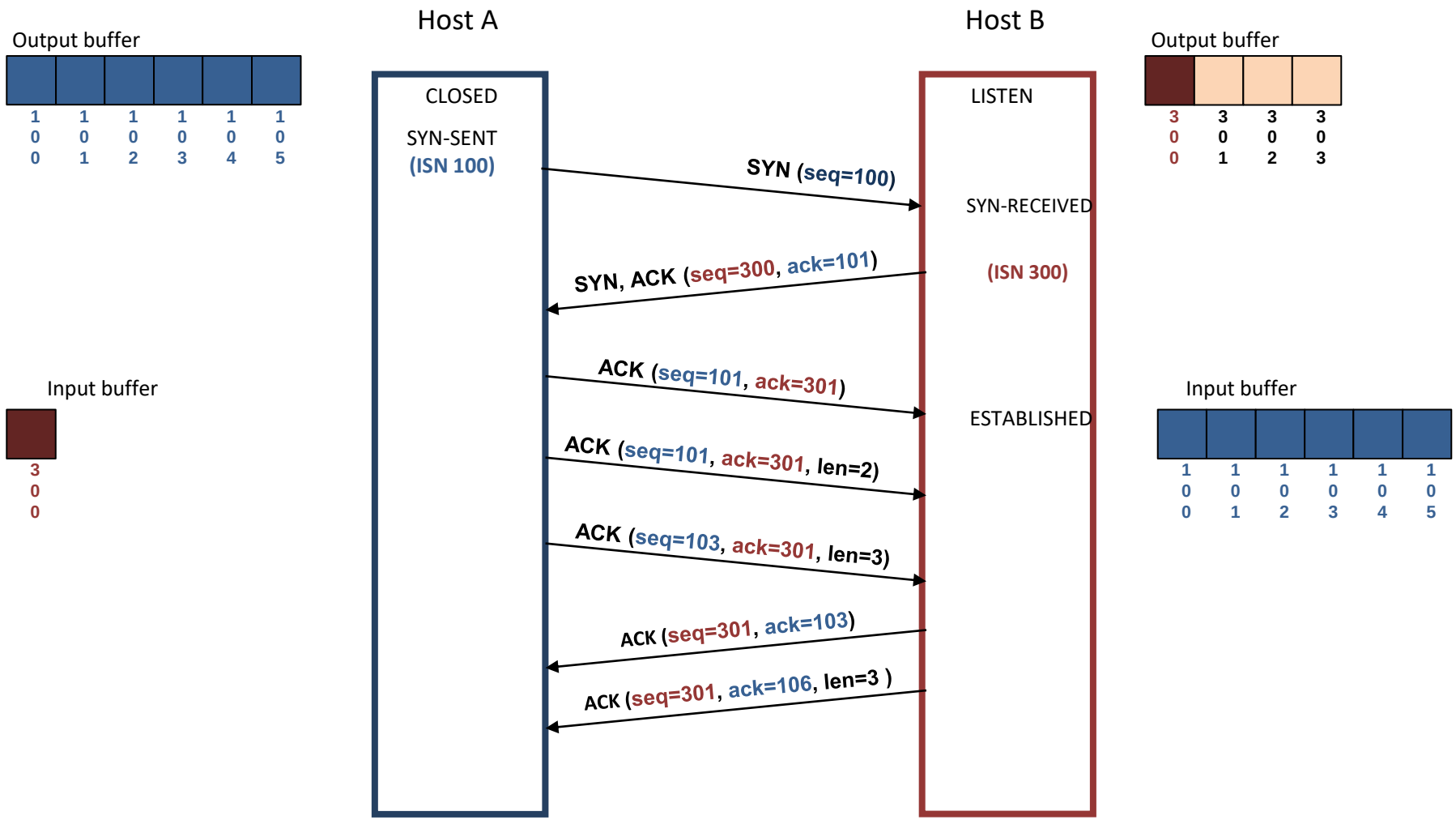
TCP PROTOKOLOA

DATU TRUKAKETA



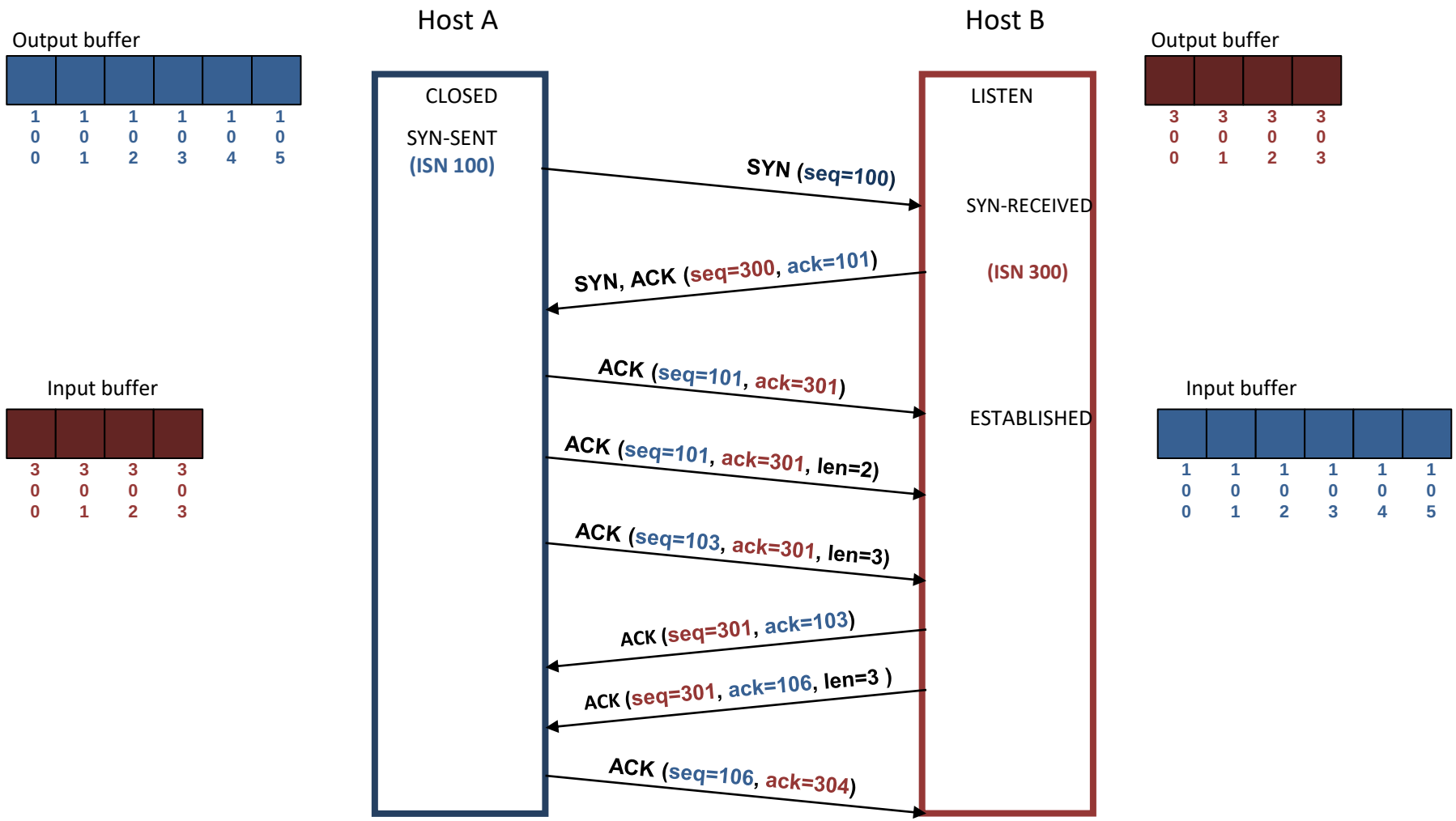
TCP PROTOKOLOA

DATU TRUKAKETA



TCP PROTOKOLOA

DATU TRUKAKETA



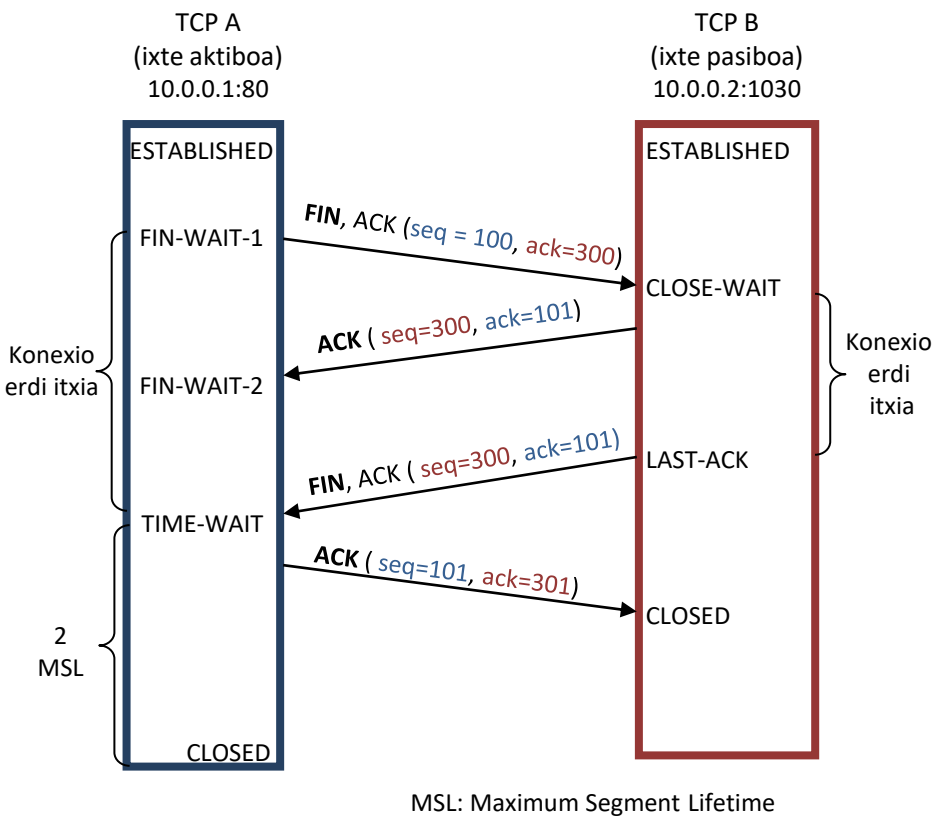
TCP PROTOKOLOA

DESKONEXIOA

- Kontuan hartu behar da **bai bezeroa bai zerbitzaria konexioa eten edo itxi dezaketela**.
- El mecanismo de desconexión utilizado por TCP se basa en el intercambio de cuatro mensajes
- **Bezeroa deskonexioa hasten badu** hurrengo urratsak jarraituko dira:
  1. Bezeroak bidaltzeko datu gehiago ez dituenean **FIN** flag aktibatuta duen segmentu bat bidaltzen du.
  2. Zerbitzariak **ACK** bidaltzen du bezerotik zerbitzarira bidalitako **FIN** hori jaso duela adierazteko.
  3. Zerbitzariak datu gehiago ez duenean bidaltzeko, **FIN** bat bidaltzen dio bezeroari saioa bukatzeko.
  4. Bezeroak **ACK** batekin erantzuten du zerbitzariaren **FIN** jaso duela adierazteko.
- Hiru-bideko lotura baten bidez konexioa bukatzea posible da:
  - Bezeroak bidaltzeko datu gehiago ez dituenean zerbitzariari **FIN** bat bidaltzen dio.
  - zerbitzariak datu gehiago ez baditu **FIN** eta **ACK**-rekin erantzungo dio biak pakete bakarrean konbinatuz.
  - bezeroak **ACK** batekin erantzungo dio.

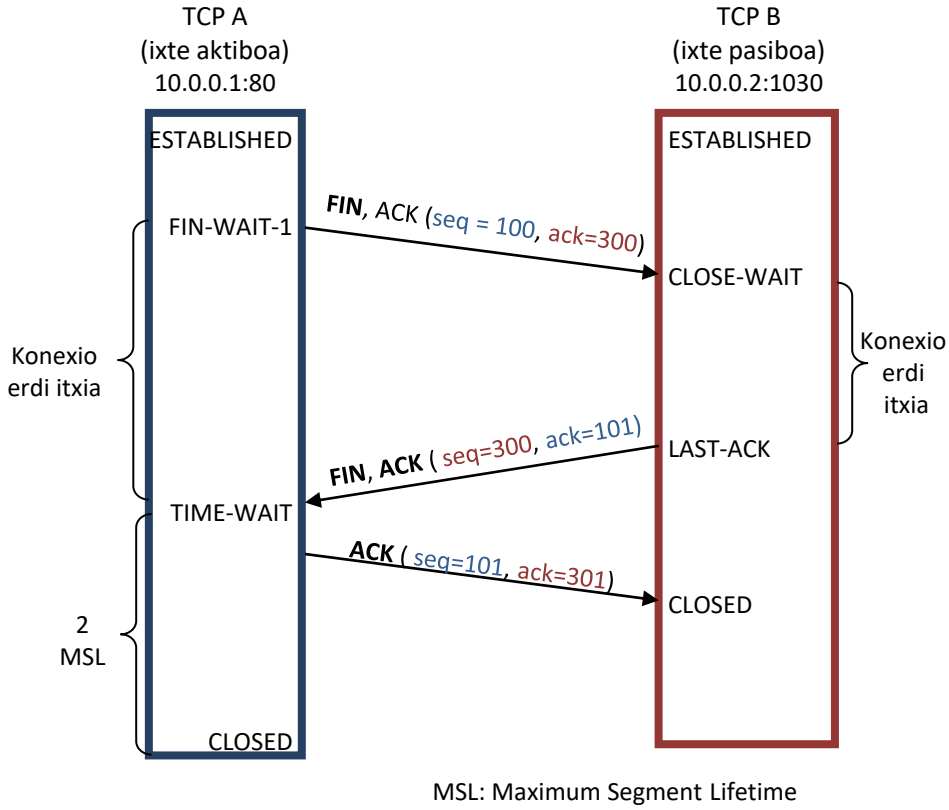
TCP PROTOKOLOA
DESKONEXIOA

4 mezuren bidez eginiko deskonexioa



MSL: Maximum Segment Lifetime

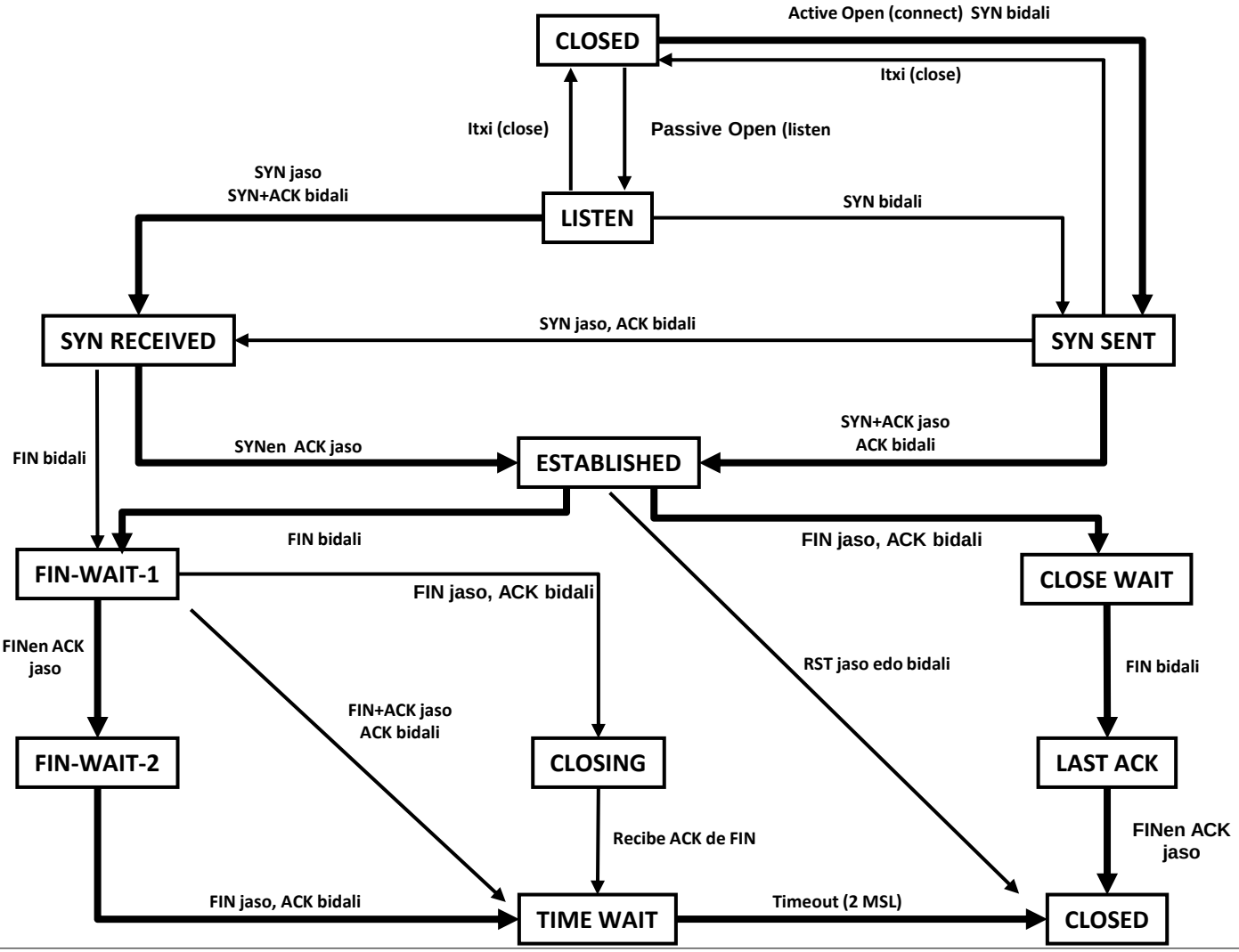
3mezuren bidez eginiko deskonexioa



MSL: Maximum Segment Lifetime

TCP PROTOKOLOA

ESTATU DIAGRAMA

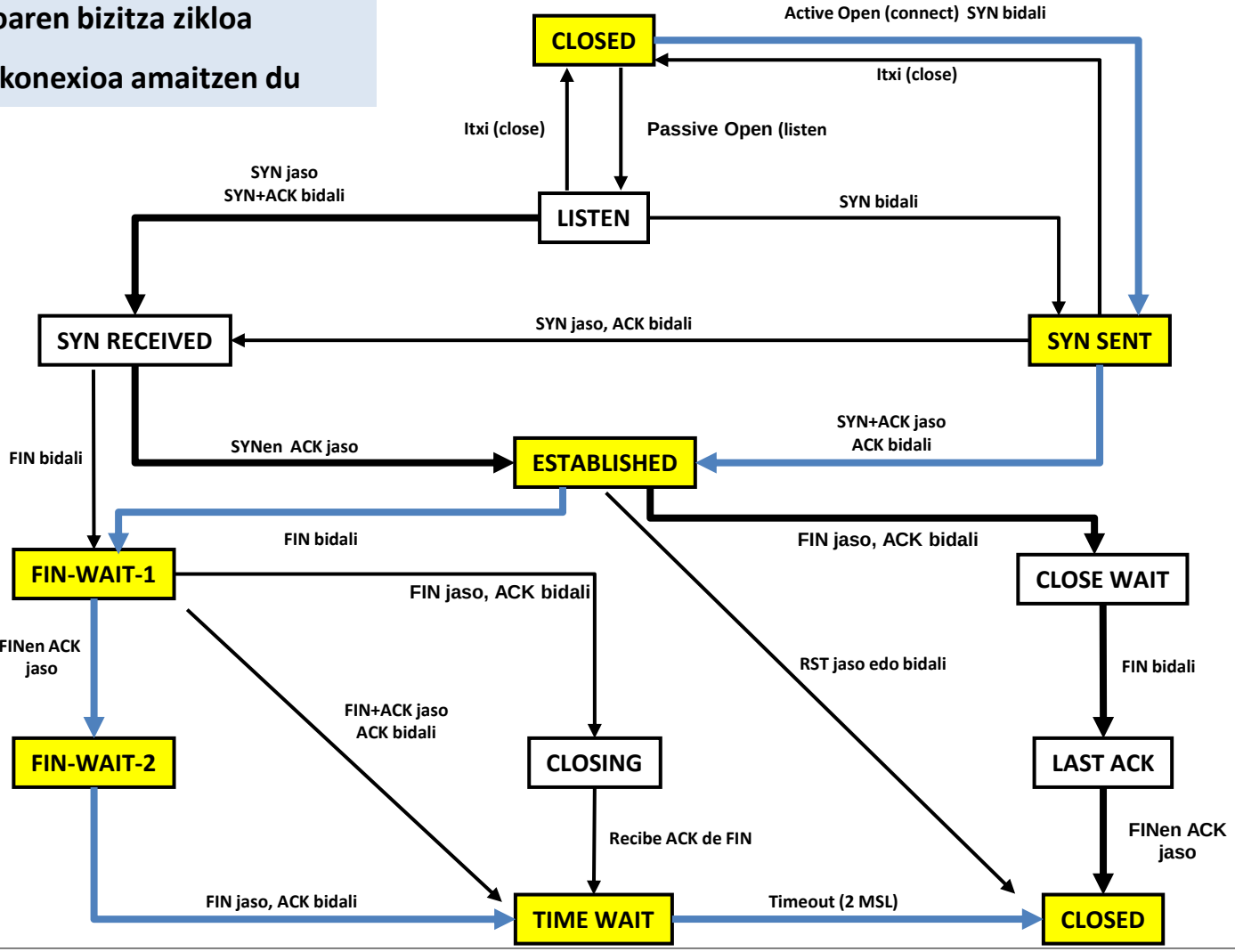


TCP PROTOKOLOA

ESTATU DIAGRAMA

Bezeroaren bizitza zikloa

Bezeroak konexioa amaitzen du

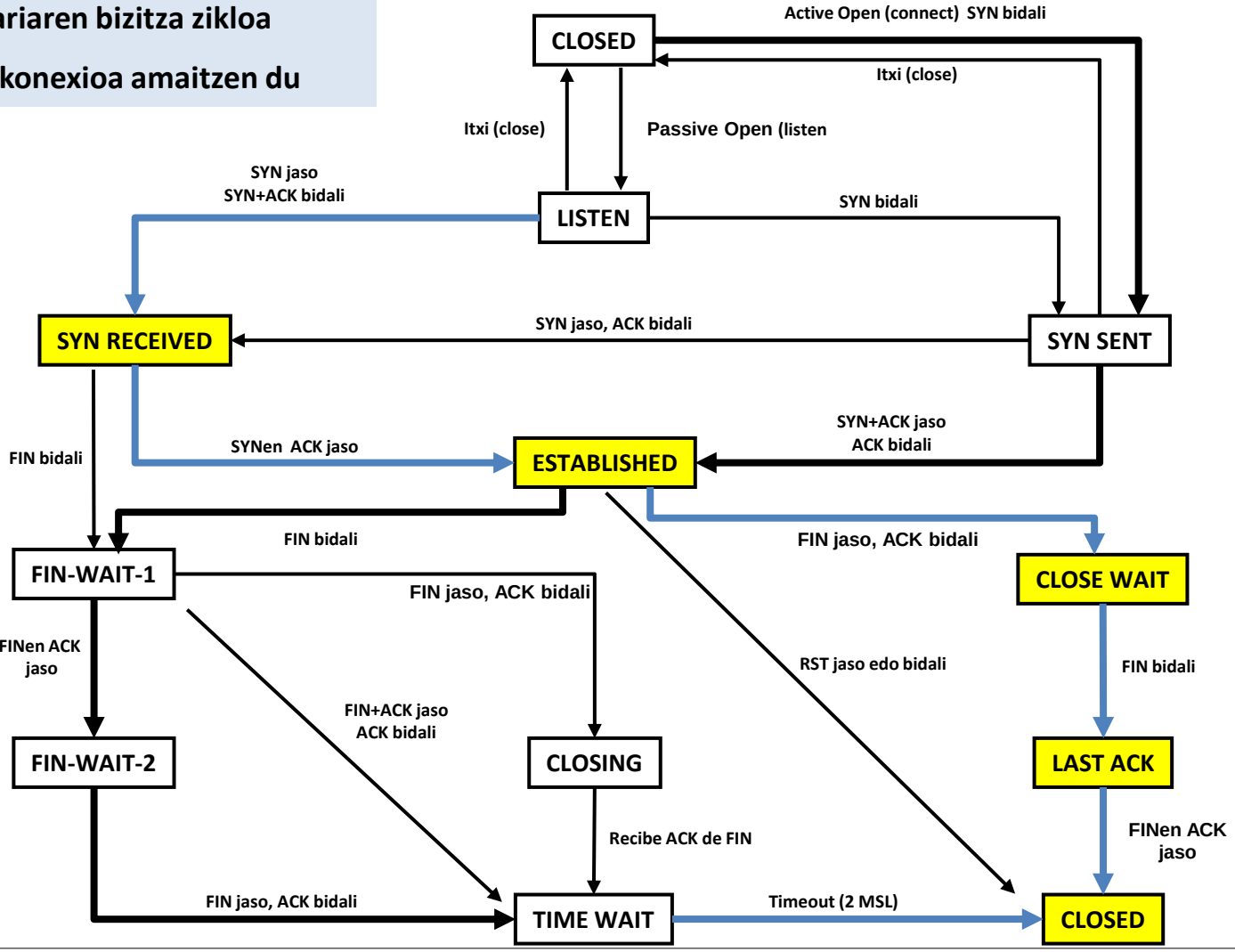


TCP PROTOKOLOA

ESTATU DIAGRAMA

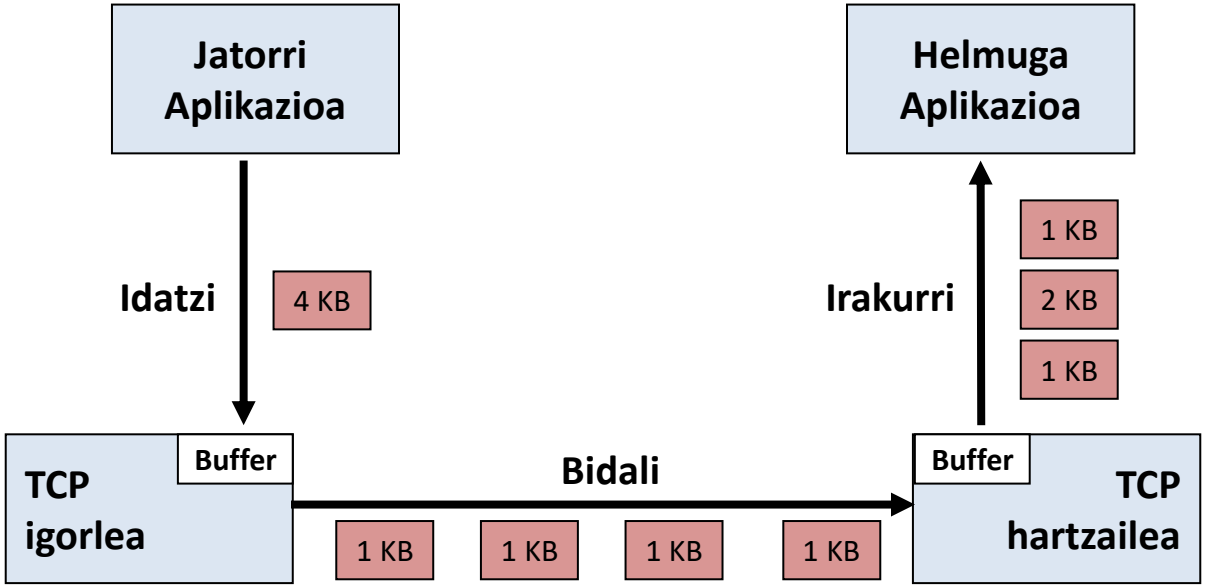
Zerbitzariaren bizitza zikloa

Bezeroak konexioa amaitzen du



TCP PROTOKOLOA

- **Jatorri aplikazioa→ TCP**
  - **Aplikazioa** TCPri datuak bidaltzen dizkio nahi duenean..
- **TCP → Helmuga Aplikazioa**
  - Aplikazioa TCP bufferretik irakurtzen du nahi duenean eta nahi duena (datu urgenteak izan ezik)



Zer gertatzen da segmentuak ordenatuta iristen ez badira?
Zer gertatzen da bufferra beteta badago?
Zer gertatzen da nodo batek segmentu tamaina onartzen ez badu?

TCP PROTOKOLOA

SEGMENTUEN REERAIKIERA

TCPrekin segmentuak desordenatuta iritsi daitezke helmugara.

- Hartzailea mezu originala ulertu dezan datu segmentuak **BIRMUNTATZEN** dira **jatorrizko ordenean**.
- Hori lortzeko **SEKUENTZIA ZENBAKIA (SEQ)** erabiltzen dira pakete bakoitzaren goiburuan.
  - Saioaren konfigurazioan **ISN** bat ezartzen da**.**
  - Zenbaki honek saio horretan bidaliko diren byten hasiera adierazten dio aplikazio hartzaileari.
  - Saioan zehar datuak transmititu ahala **SEKUENTZIA ZENBAKIA TRANSMITITU eta jakinarazi DIREN BYTEtan INKREMENTATZEN DA**.
  - Byten jarraipen hori **SEGMENTU BAKOITZA IDENTIFIKATU ETA BERE HARTZE-AGIRIA (ACK) BIDALTZEA** modu bakar batean egitea posible egiten du. Galdutako segmentuak identifikatu daitezke.
- Sekuentzia zenbakiak **fidagarritasuna** ematen dute segmentua nola birmuntatu eta nola ordenatu behar diren adierazten duelako.

TCP hartzaileak datu segmentuak hartze buffer batean jartzen ditu. Segmentuak bere sekuentzia zenbakien arabera orden zuzenean jartzen dira eta aplikazio mailara pasatzen dira birmuntatuta.

TCP PROTOKOLOA
TCP RENTRANSMISIOA

Galdutako paketeen kudeaketa

- ¿Zer gertzatzen da jatorri edo helmuga *buffera* beteta badago?
  - Segmentuak galtzen dira.
- TCP segmentu galduak kudeatzeko metodoak ditu. Hauen artean **HARTZE-AGIRIA EZ DUTEN SEGMENTUAK BIRBIDALTZEA** da.
- TCP erabiltzen duen helmuga hosteko zerbitzu batek normalean **bakarrik sekuentzi zenbaki jarraituak dituzten byten hartze-agiria ematea da**.
  - *Adib. Sekuentzia zenbaki 1500-tik 3000-tara jaso badira eta 3400-tik 3500-tara, hartze agiria 3001 izango litzateke. Horrela 3001-tik 3399-tara sekuentzia zenbakia duten segmentuak ez direla jaso esango du.*
- Jatorri hosteko TCP **denboraldi batean hartze-agiri bat jasotzen ez duenean**, jaso duen azkeneko hartze-agiria itzuliko du eta **puntu hortik aurrera berriro bidaliko** ditu datuak.

- Gaur egun, host-ak “hartze agiri selektiboa” deritzon hautazko funtzio bat ere erabili dezakete. Host-ak **HARTZE-AGIRI SELEKTIBO** onartzen badute, hartzaileak segmentu ez jarraituen byteak antzeman ditzake eta orduan bakarrik galdutako datuak birbidali behar izango dira

Ariketa X (Hartze-agiri selektibo barik eta izanda)

TCP PROTOKOLOA

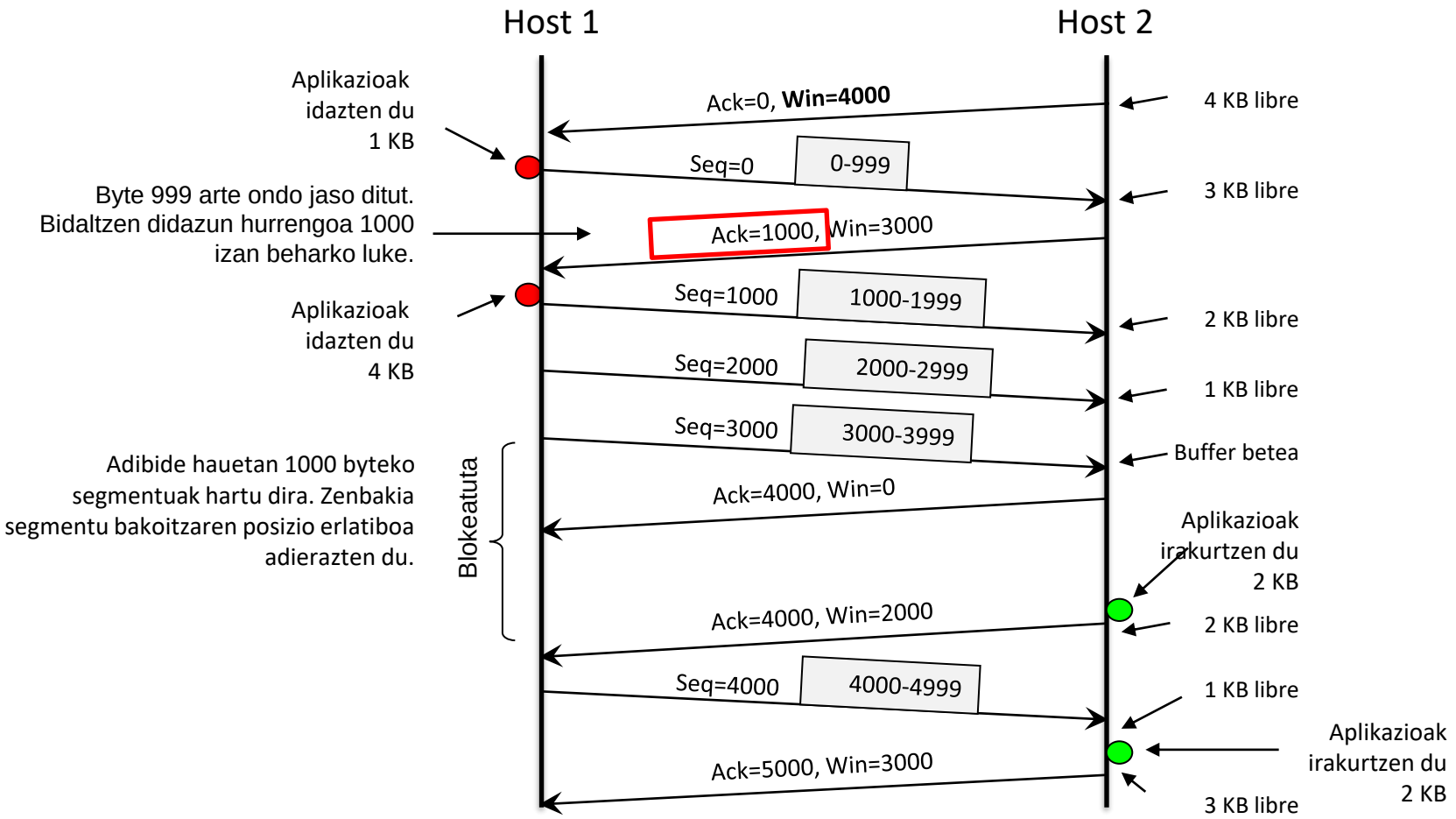
FLUXU KONTROLA

- Jasotako aplikazioaren datuak irakurtzeko abiadura igorlearen transmisio abiadura baino txikiagoa bada, gerta daiteke harrera bufferra saturatuta egotea eta segmentuak galtzea.
- TCP Goiburuko **WINDOW** elementua hartze-agiria jaso baino lehen transmititu daitekeen datu kopurua ezartzen du.
  - Hasierako **leihoaren tamaina saioaren hasieran ezartzen da** hiru-bideko agurra erabiliz.
  - **Adibidea:** *Hasierako tamaina TCP saio baterako 3000 byte bezala ezartzen da. Igorleak 3000 byte bidali dituenean hauen hartze-agirien zain geratzen da segmentu gehiago transmititu baino lehen. Behin hartze-agiriak jaso dituenean beste 3000 byte bidali ditzake.*

- **LEIHOAREN NEURRI DINAMIKOAK.**
  - Helmugak komunikazio abiadura **moteldu** behar badu memoria buffer mugatua duelako, leihoaren tamainaren balio txikiagoa bidal diezaioke iturriari **aitorpen (jakinarazpen)** batean.
  - Transmisioaren abiadura modu eraginkorrean **murrizten** da, **iturriak datuak maizago aitortzea espero duelako.**
  - Datu galerarik edo baliabide murritzik gabeko transmisio aldien ondoren, helmugak leihoaren tamaina handitu dezake. Aitorpen gutxiago eskatzeak sareko gainkarga murrizten du. Leihoaren tamaina handitzen joango da datuak galtzen diren arte, eta horrek leihoaren tamaina txikituko du.

TCP PROTOKOLOA

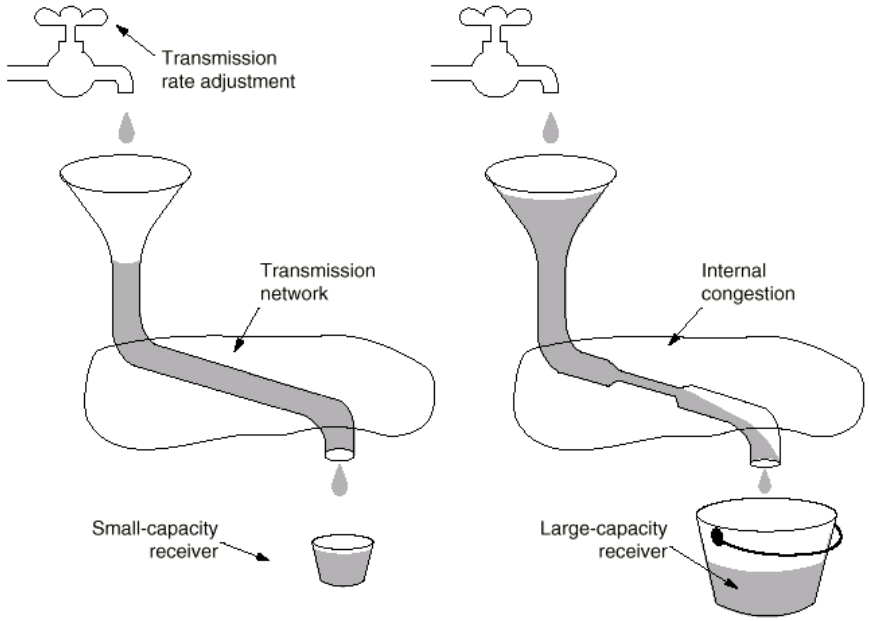
FLUXU KONTROLA



TCP PROTOKOLOA

FLUXU KONTROLA VS KONGESTIOA

- **Fluxu kontrola.**
  - Abiadura desberdinarekin informazioa sortarazi/prozesatzen bi entitateen arteko informazio bidaltzea sinkronizatzea posible egiten duen teknika da.
- **Itomen/Kongestio kontrola.**
  - Fluxu kontrola baino kontzeptu zabalagoa da. Teknika hauek sare oso bat erabili ahal ez denean agertu daitekeen akatsa edo erroreak detektatu eta zuzentzea posible egiten dute, zerbitzu kalitate ikuspuntutik beharrezkoak diren berandutzeekin adibidez. Hortaz, kontzeptu global bat da, sare osoa kontutan hartzen du eta ez bakarrik igorle eta hartzailea fluxu kontrolean gertatzen den bezala



Fluxu kontrola itomenari aurre egiteko erabiltzen den teknika bat da. Berarekin sarera trafiko gehiegi botatzen dituzten jatorriak geldiarazten dira. Baina beste mekanismo batzuk ere aurkitu daitezke.

PRAKTIKAK ETA AKTIBITATEAK

TCP / UDP

NETSTAT

Aktibitatea

Zure ordenagailuan aplikazioak aktibatu daitezen ataka zenbakiak bilatu

NETSTAT-ek

- Ezarrita dauzkagun TCP konexioak ikustarazten digu.
- erabiltzen ari den protokoloa, ataka lokalaren helbidea eta zenbakia, kanpoko ataken helbide eta zenbakia (socket) eta konexioaren egoera ere adierazten ditu.

Firewalla babesmen modu arrunta da beharrezkoak ez diren atakak blokeatzen duena, hau da, irekita dagoen zerbitzuen ataka zenbakiei ez dagozkion paketeak sartzen ez utzi.

Gainera, TCP konexio ez beharrekoak sistemaren baliabide garrantzitsuak erabili ditzakete hostaren errendimenda murriztuz

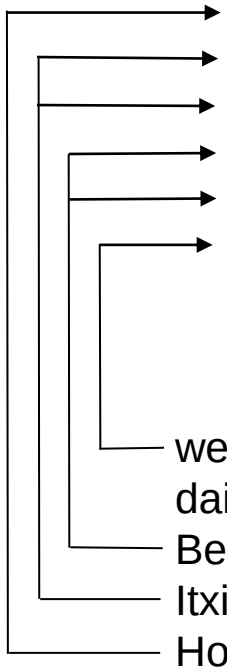
TCP / UDP

NETSTAT

```
C:\>netstat -n
Conexiones activas

Proto Dirección local    Dirección remota          Estado

  TCP  10.0.1.25:3719     10.0.1.60:21           ESTABLISHED
  TCP  10.0.1.25:4111     10.0.1.50:110          TIME_WAIT
  TCP  10.0.1.25:4113     10.0.1.50:110          TIME_WAIT
  TCP  10.0.2.13:80       10.0.2.40:1056         ESTABLISHED
  TCP  10.0.1.25:80       10.0.1.30:2312         ESTABLISHED
  TCP  *:80               *:*                    LISTEN
C:\>
```



web zerbitzariak entzuten host honetan. Bere IP guztiak (*.80) 80 atakan konektatu daitezke edozein **IP helbide eta atakatik (*.*)**.

Bezeroen konexioak web zerbitzariarekin.

Itxi gabeko saioa posta bezero bezala 10.0.1.50-tik.

Host honen konexioa ftp bezero bezala 10.0.1.60-tik.

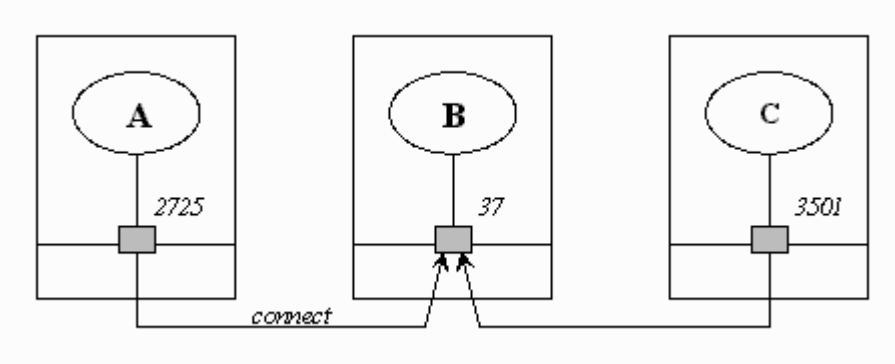
"-n" aukera ezartzen ez bada, netstat IP helbide guztiak eta ataka guztiak izenak bilakatzen saiatzen da ("pop3" **jarriko luke 110 ordez**).

GALDERAK

1.- Demagun 3 host ditugula: A, B eta C. Host A eta C-n 2725 eta 2501 atakak hurrenez hurren erabiltzen dituzten bi bezero ditugu. B hostan TCP zerbitzari konkurrentea daukagu, 37 ataka erabiltzen du zerbitzua eskaintzeko (irudian ikusten den bezala). A bezeroak B zerbitzariarekin konektatzen da eta honek konexioa onartzen du. Gero, C bezeroa B zerbitzariarekin konektatzen da.

(a) Nola daki B bi konexiak nola bereiztu biak ataka berdinaren bidez (37) kudeatuta badaude?

(b) Zer gertatuko litzateke C bezeroaren ataka lokala 2725 izango balitz?



TCP CHECKSUMA

Erakutsitako informazioa Ethernet V2 fotograma baten barruan dago eta TCP goiburuko checksum-a ez da erakusten, ezta CRC ezta Ethernet-en hitzaurrea ere. Segmentu honek ez ditu datuak eramaten, TCP *three-way handshake* lehenengo segmentua da.

```
Offset:  Hexadecimal                                              ASCII
-------  -----------                                              -----
 0:  0800  2073  5ec6  00e0  1ede  72d6  0800  4500        ..  s^.....r...E.
16:  0030  6e1e  4000  7f06  35c4  a8b0  036c  a8b0        .0n.@...5....l..
32:  0319  0593  0015  008a  ca9b  0000  0000  7002        ..............p.
48: 2000  ____  0000 0204 0218 0101 0402                 .>..............
```

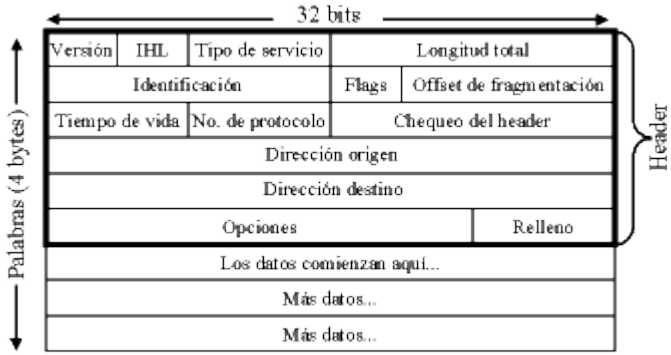
Ethernet frame formatua
(tamainak bytetan emanda)

Destino	Origen	Tipo	Datos	Chequeo
6	6	2	46 - 1500	4

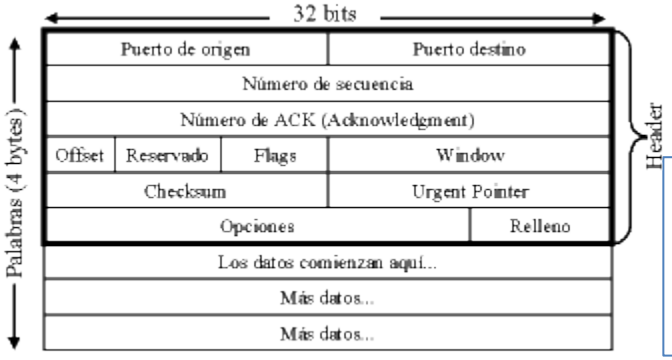
BINARIO	0000	0001	0010	0011	0100	0101	0110	0111
HEXADECIMAL	0	1	2	3	4	5	6	7

BINARIO	1000	1001	1010	1011	1100	1101	1110	1111
HEXADECIMAL	8	9	A	B	C	D	E	F

IPv4 header formatua



· **TCP header formatua**



Protokolo zenbakia (hamartarra)	
6	TCP
17	UDP

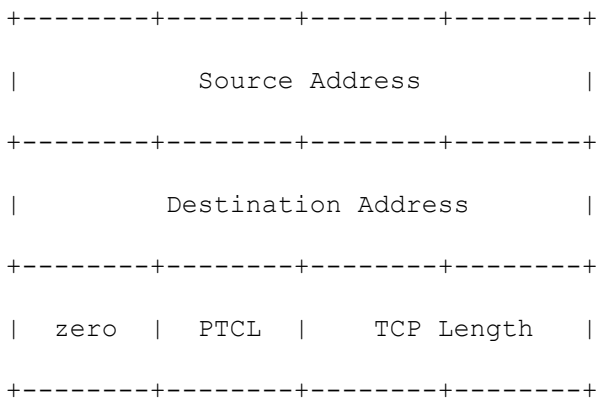
TCP CHECKSUMA

[...]

Checksum: 16 bits

The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header and text. If a segment contains an odd number of header and text octets to be checksummed, the last octet is padded on the right with zeros to form a 16 bit word for checksum purposes. The pad is not transmitted as part of the segment. While computing the checksum, the checksum field itself is replaced with zeros.

The checksum also covers a 96 bit pseudo header conceptually prefixed to the TCP header. This pseudo header contains the Source Address, the Destination Address, the Protocol, and TCP length. This gives the TCP protection against misrouted segments. This information is carried in the Internet Protocol and is transferred across the TCP/Network interface in the arguments or results of calls by the TCP on the IP.



The TCP Length is the TCP header length plus the data length in octets (this is not an explicitly transmitted quantity, but is computed), and it does not count the 12 octets of the pseudo header.

[...]

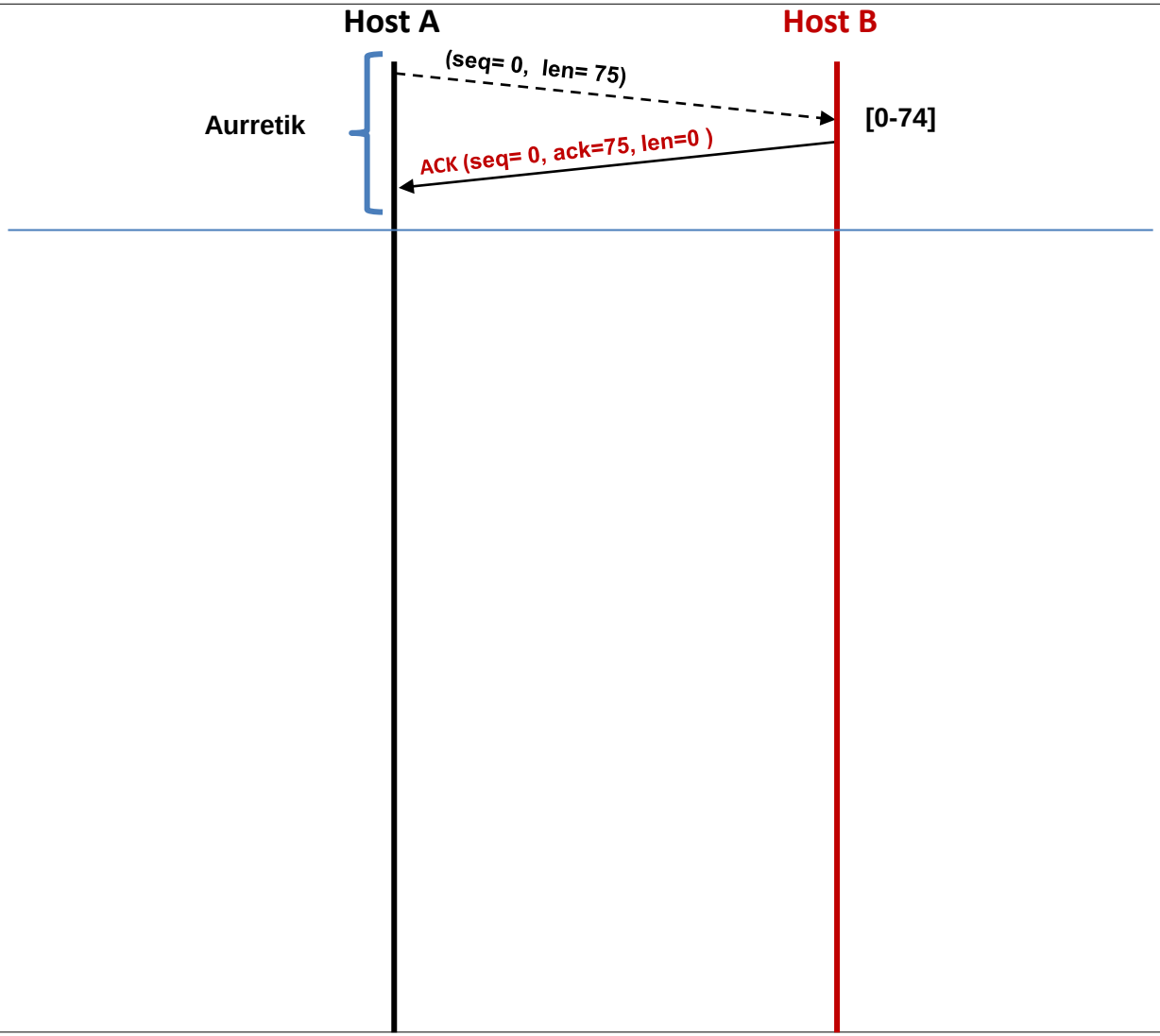
X ARIKETA

- Adierazi TCP konexio hau, **transmisio eta harrera-leihoak nahiko handiak direla** suposatuz, tenporizadoreak ere nahiko handiak direla jarraian 5 datu segmentu eta dagozkien ACK ostalari hartzaileak (B ostalariak) eta igorleak (A ostalariak), hurrenez hurren, jaso ahal izateko (kanalean galerarik ez badago), **eta aitorpen selektiborik gabe**.

- Ostalari hartzaileak (B) dagoeneko datuen lehen 75 byte (0tik 74ra bitartekoak) arrakastaz jaso ditu. Demagun A ostalariak 10 byteko 5 datu segmentu berriak bidaltzen dituela jarraian B ostalariari, eta lehenengo segmentua galtzen dela (bidaltzen den lehen aldia). Azkenean, 5 datu segmentuak B ostalariak ongi jaso ditu.

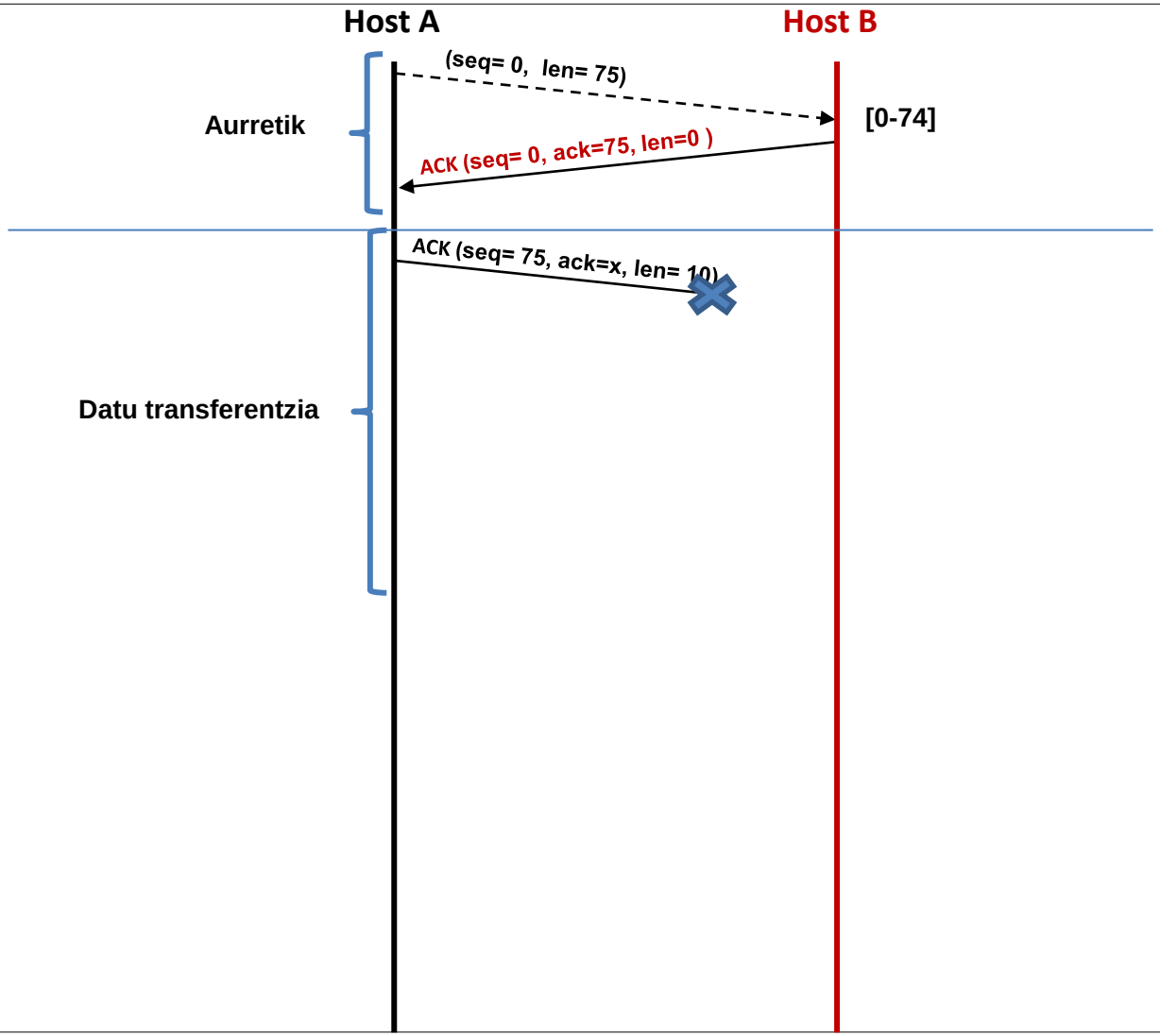
- Segmentu eta ACK trukaketa grafikoki adierazi, beti segmentuen sekuentzia zenbakiak eta ACK zenbakiak jarriz, baita tenporizadoreak ere. Jasotako segmentuak harreran nola kudeatzen diren ere adierazi (adibidez, bikoiztuak badaude, zer egin)

X ARIKETA



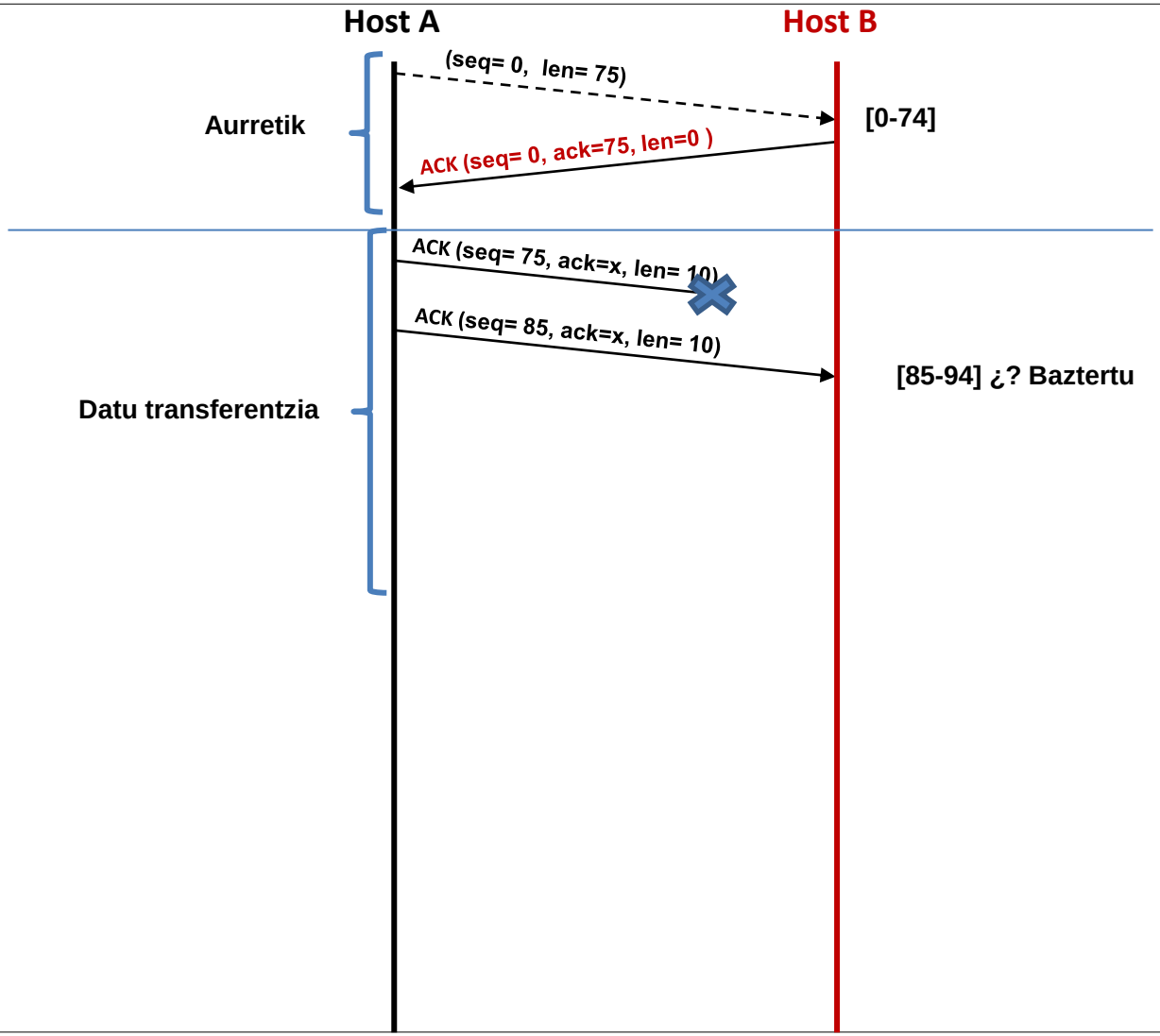
Gaizka Saldaña
Dpto. Ingeniería de Sistemas y Automática

X ARIKETA

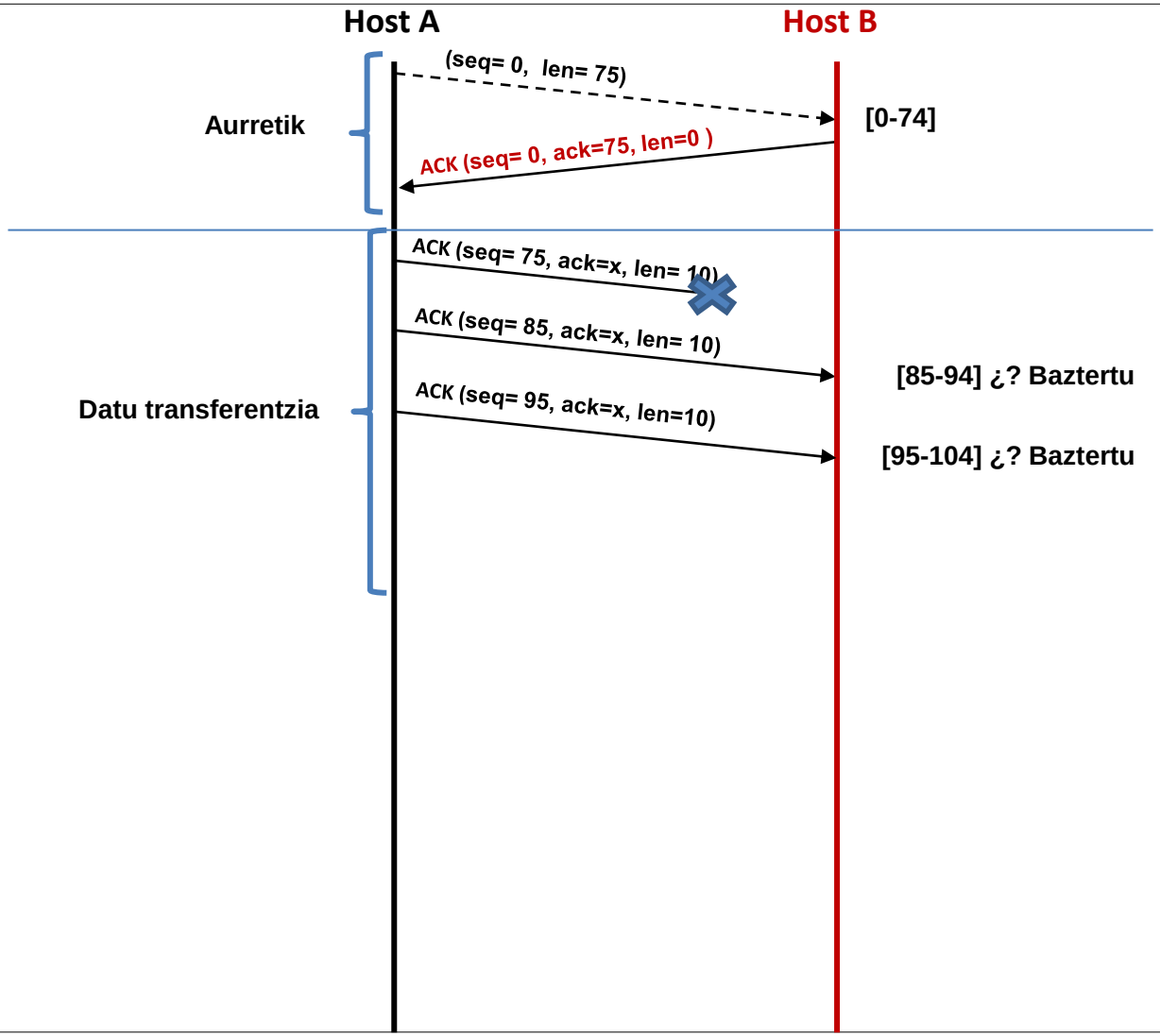


Gaizka Saldaña
Dpto. Ingeniería de Sistemas y Automática

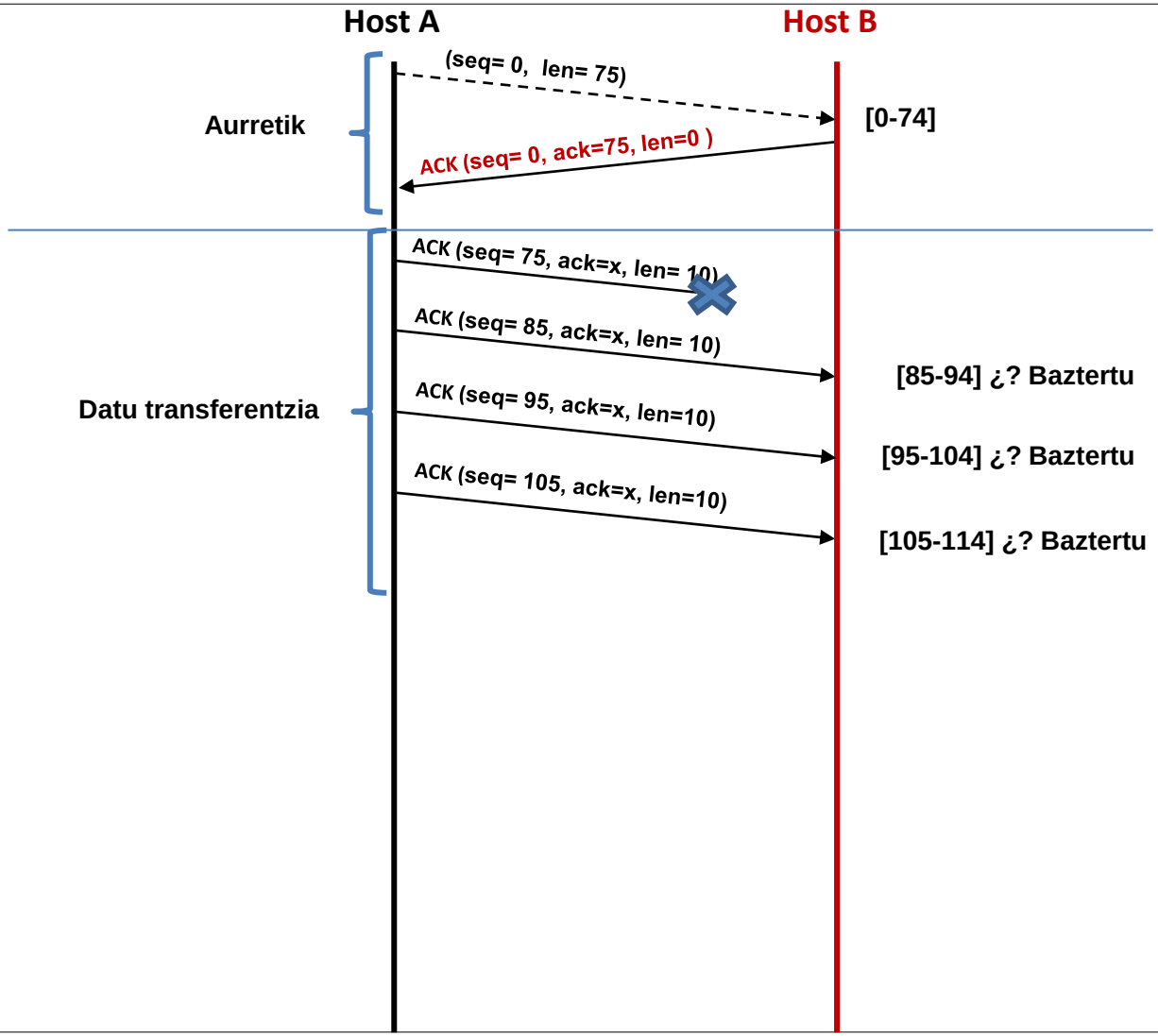
X ARIKETA


Host A
Host B
(seq= 0, len= 75)
[0-74]
Aurretik
ACK (seq= 0, ack=75, len=0)
ACK (seq= 75, ack=x, len= 10)
ACK (seq= 85, ack=x, len= 10)
[85-94] ¿? Baztertu
Datu transferentzia

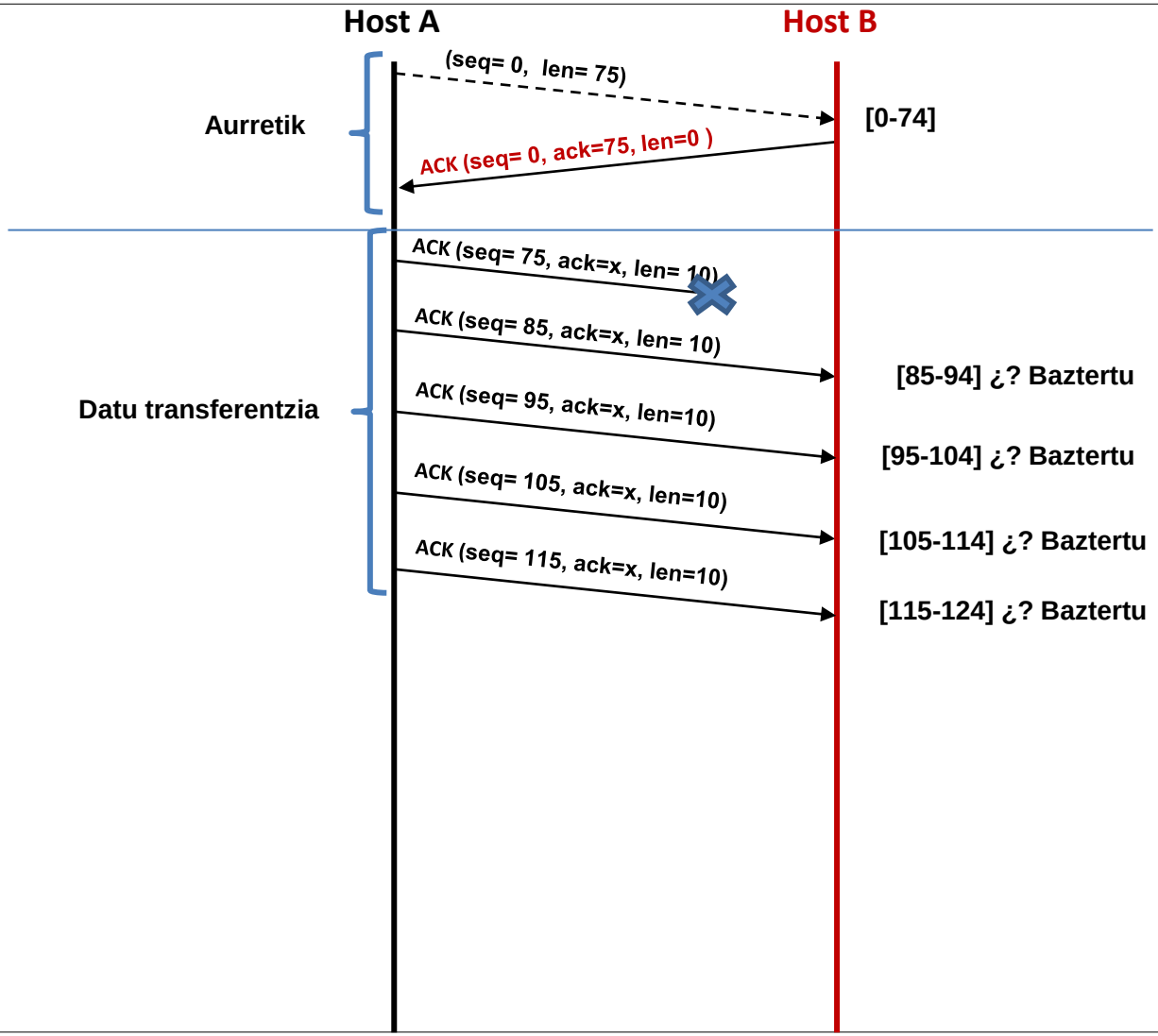
X ARIKETA


Host A
Host B
(seq= 0, len= 75)
[0-74]
Aurretik
ACK (seq= 0, ack=75, len=0)
ACK (seq= 75, ack=x, len= 10)
ACK (seq= 85, ack=x, len= 10)
[85-94] ¿? Baztertu
Datu transferentzia
ACK (seq= 95, ack=x, len=10)
[95-104] ¿? Baztertu

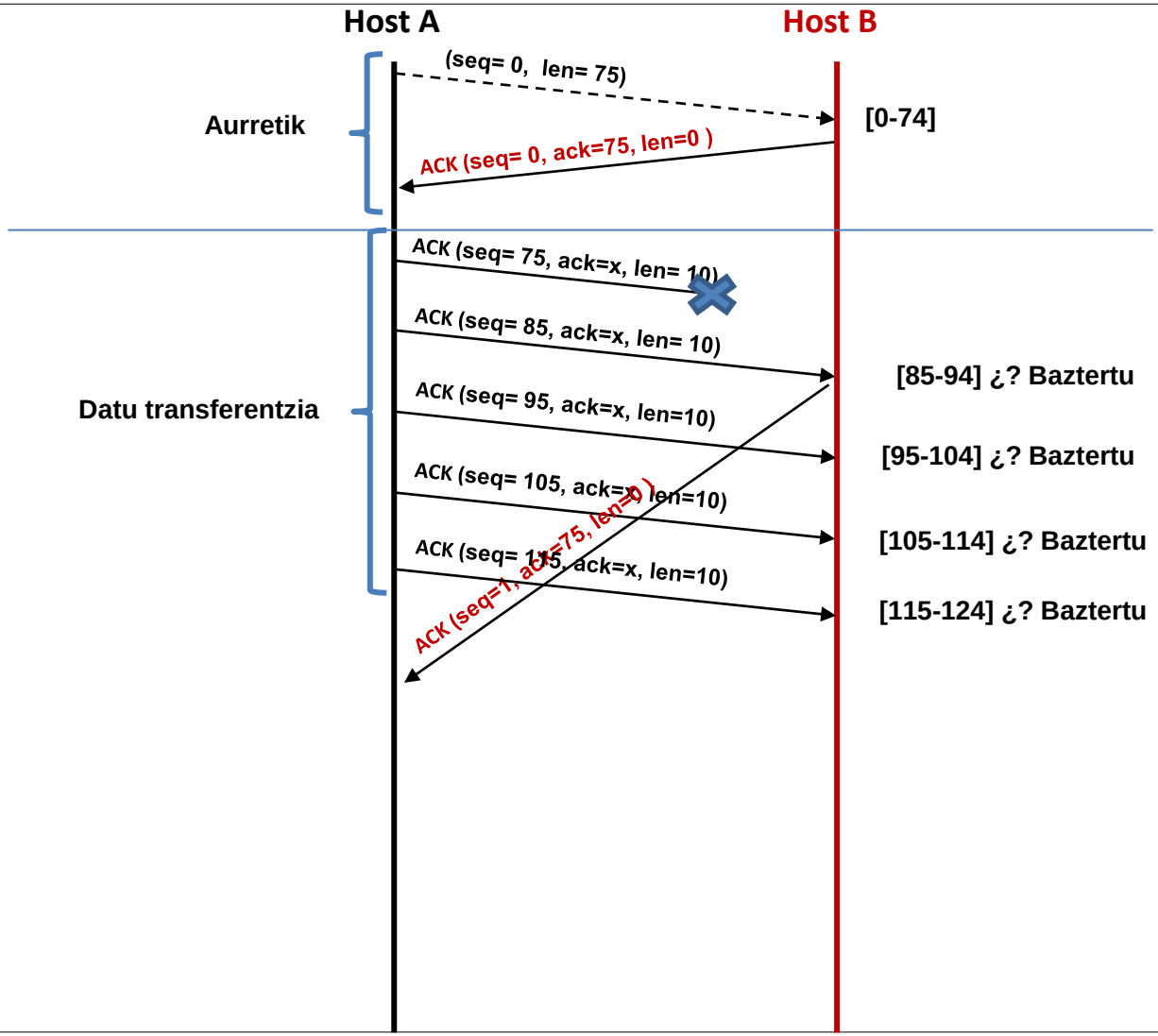
X ARIKETA


Host A
Host B
(seq= 0, len= 75)
[0-74]
Aurretik
ACK (seq= 0, ack=75, len=0)
ACK (seq= 75, ack=x, len= 10)
ACK (seq= 85, ack=x, len= 10)
[85-94] ¿? Baztertu
Datu transferentzia
ACK (seq= 95, ack=x, len=10)
[95-104] ¿? Baztertu
ACK (seq= 105, ack=x, len=10)
[105-114] ¿? Baztertu

X ARIKETA

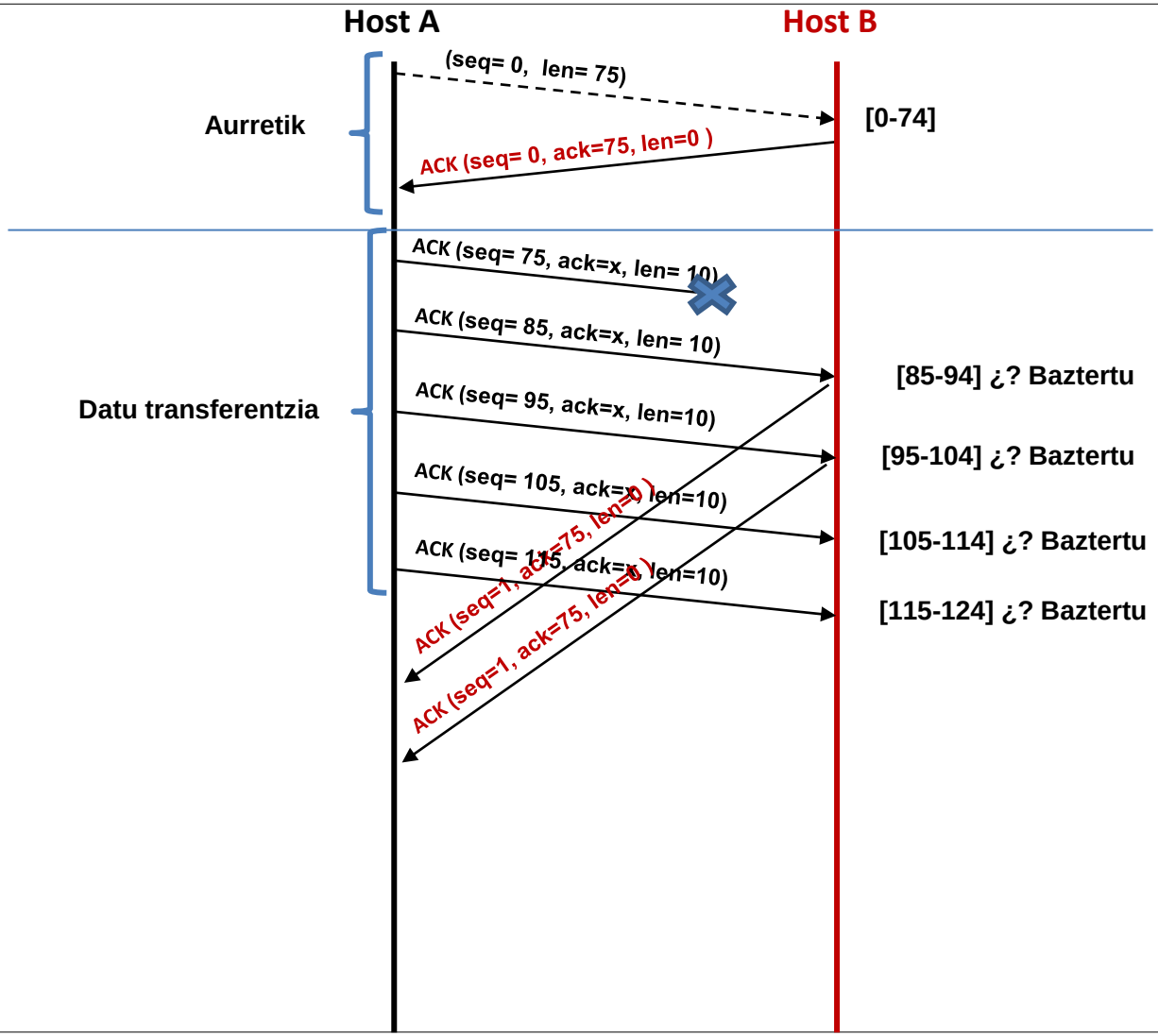

Host A
Host B
(seq= 0, len= 75)
[0-74]
Aurretik
ACK (seq= 0, ack=75, len=0)
ACK (seq= 75, ack=x, len= 10)
ACK (seq= 85, ack=x, len= 10)
[85-94] ¿? Baztertu
Datu transferentzia
ACK (seq= 95, ack=x, len=10)
[95-104] ¿? Baztertu
ACK (seq= 105, ack=x, len=10)
[105-114] ¿? Baztertu
ACK (seq= 115, ack=x, len=10)
[115-124] ¿? Baztertu

X ARIKETA

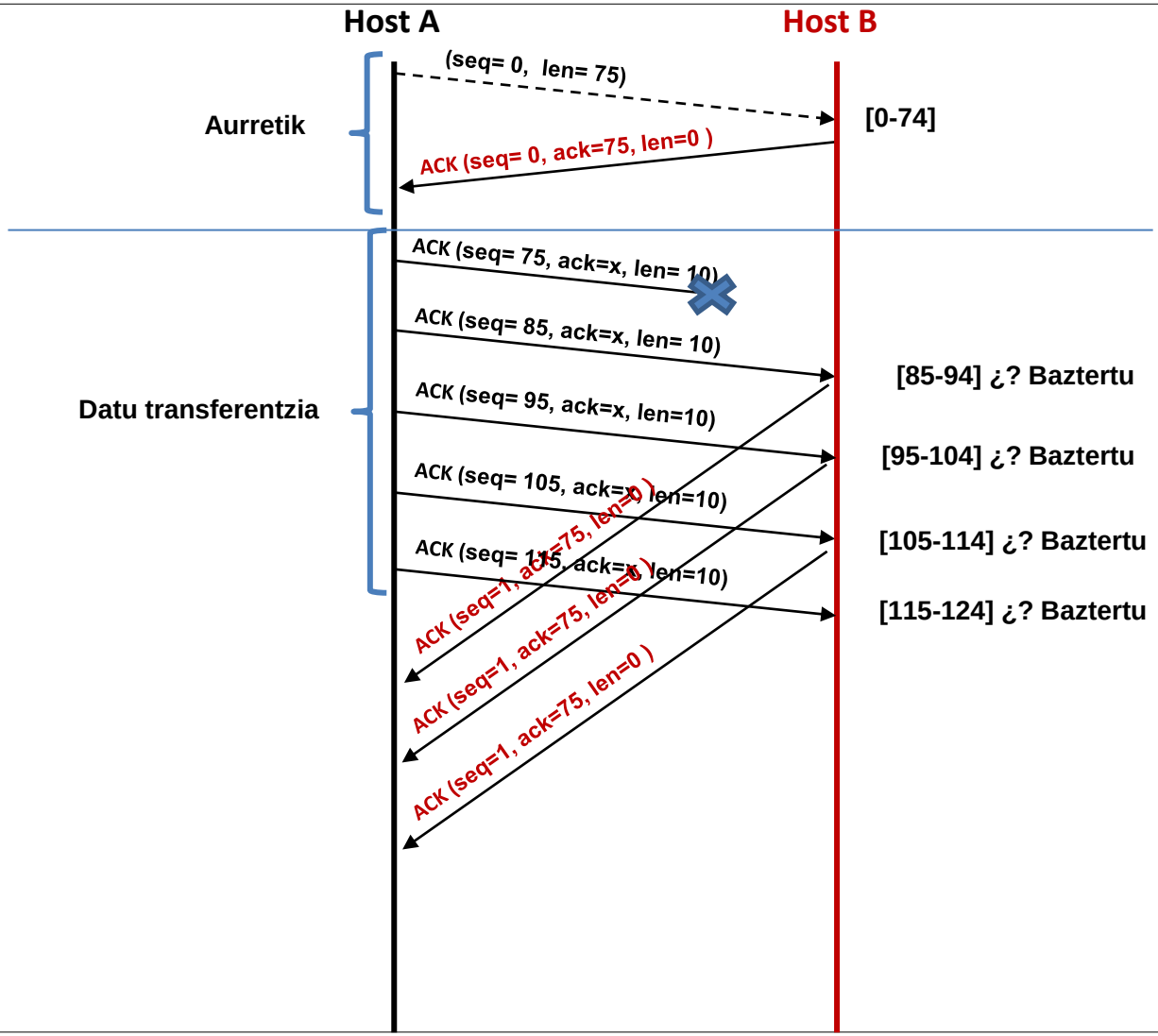


Gaizka Saldaña
Dpto. Ingeniería de Sistemas y Automática

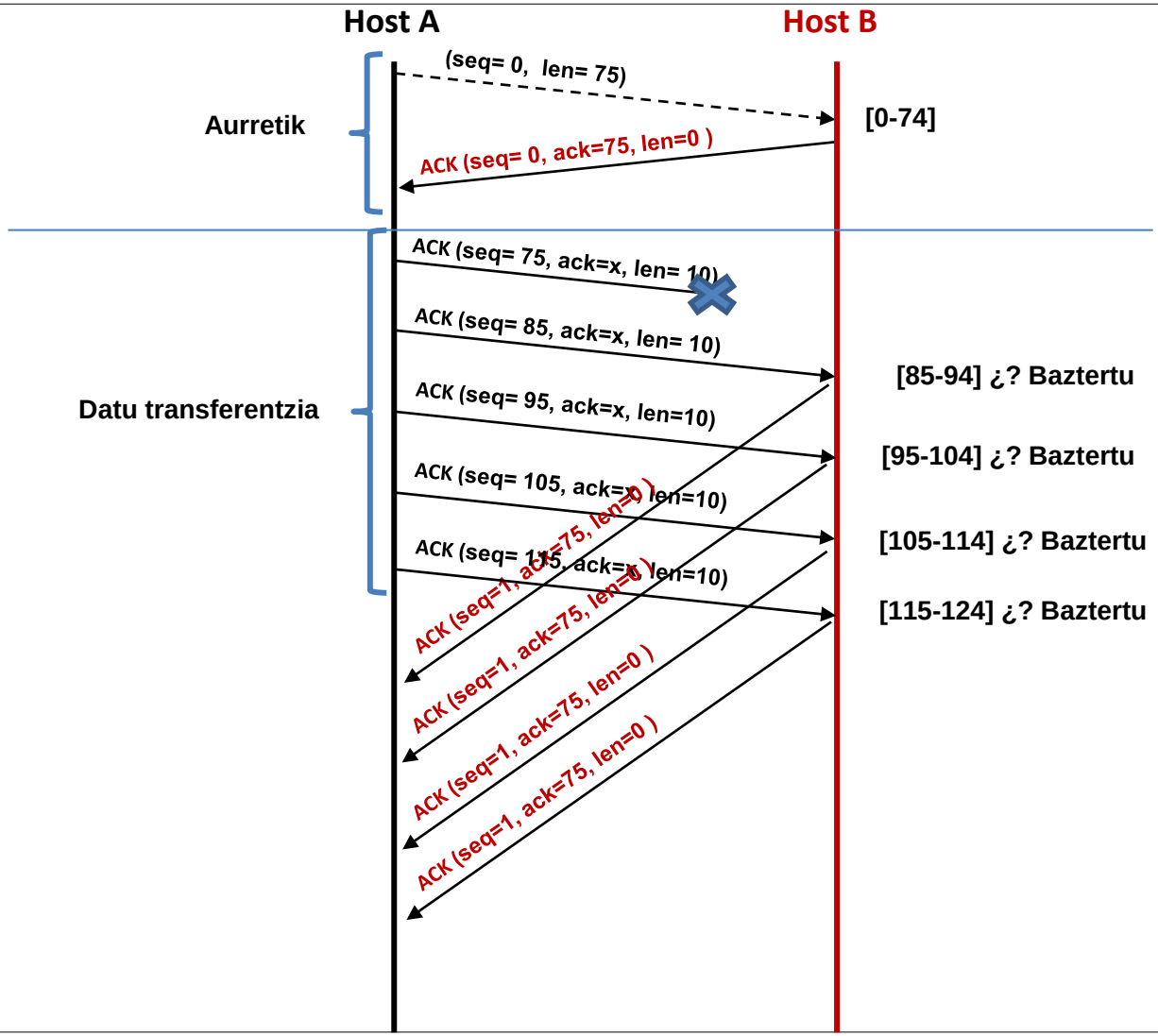
X ARIKETA


Host A
Host B
(seq= 0, len= 75)
[0-74]
Aurretik
ACK (seq= 0, ack=75, len=0)
ACK (seq= 75, ack=x, len= 10)
ACK (seq= 85, ack=x, len= 10)
[85-94] ¿? Baztertu
Datu transferentzia
ACK (seq= 95, ack=x, len=10)
[95-104] ¿? Baztertu
ACK (seq= 105, ack=x, len=10)
[105-114] ¿? Baztertu
ACK (seq= 115, ack=x, len=10)
[115-124] ¿? Baztertu
ACK (seq=1, ack=75, len=0)
ACK (seq=1, ack=75, len=0)

X ARIKETA

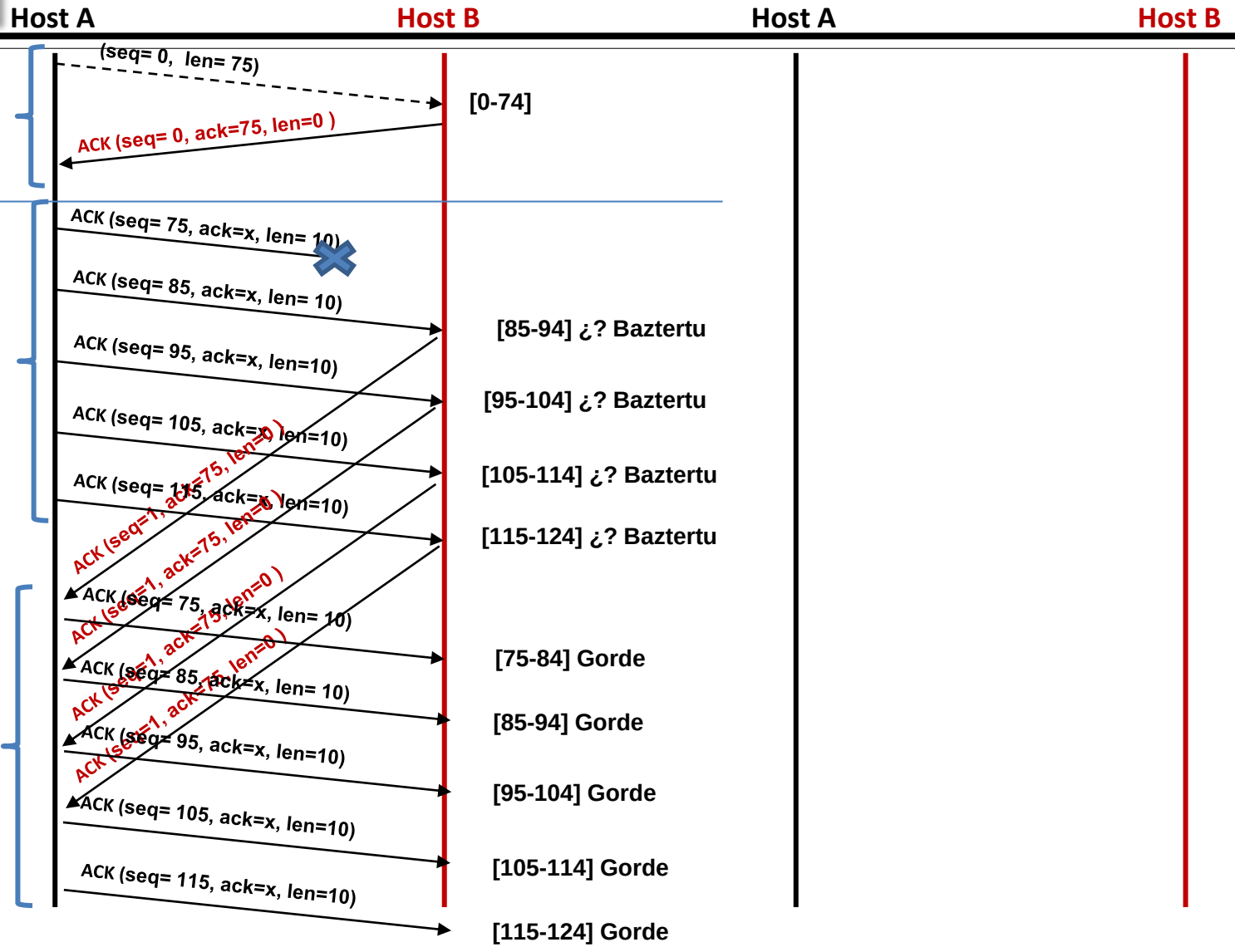

Host A
Host B
(seq= 0, len= 75)
[0-74]
Aurretik
ACK (seq= 0, ack=75, len=0)
ACK (seq= 75, ack=x, len= 10)
ACK (seq= 85, ack=x, len= 10)
[85-94] ¿? Baztertu
Datu transferentzia
ACK (seq= 95, ack=x, len=10)
[95-104] ¿? Baztertu
ACK (seq= 105, ack=x, len=10)
[105-114] ¿? Baztertu
ACK (seq= 115, ack=x, len=10)
[115-124] ¿? Baztertu
ACK (seq=1, ack=75, len=0)
ACK (seq=1, ack=75, len=0)
ACK (seq=1, ack=75, len=0)

X ARIKETA



Gaizka Saldaña
Dpto. Ingeniería de Sistemas y Automática

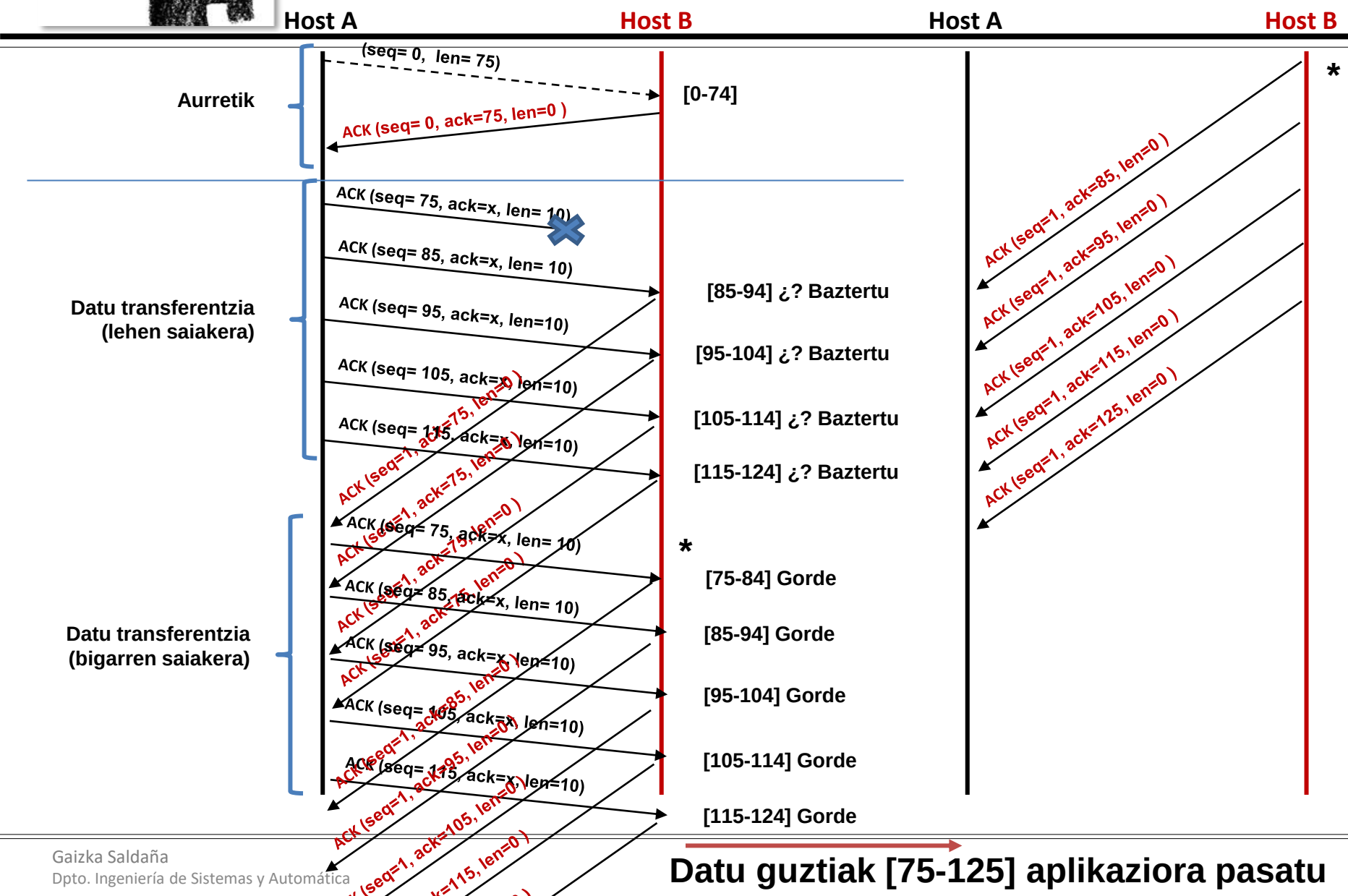
X ARIKETA



Pasa todos los datos [75-125] a aplicación

Gaizka Saldaña
Dpto. Ingeniería de Sistemas y Automática

X ARIKETA



Datu guztiak [75-125] aplikaziora pasatu

Gaizka Saldaña
Dpto. Ingeniería de Sistemas y Automática

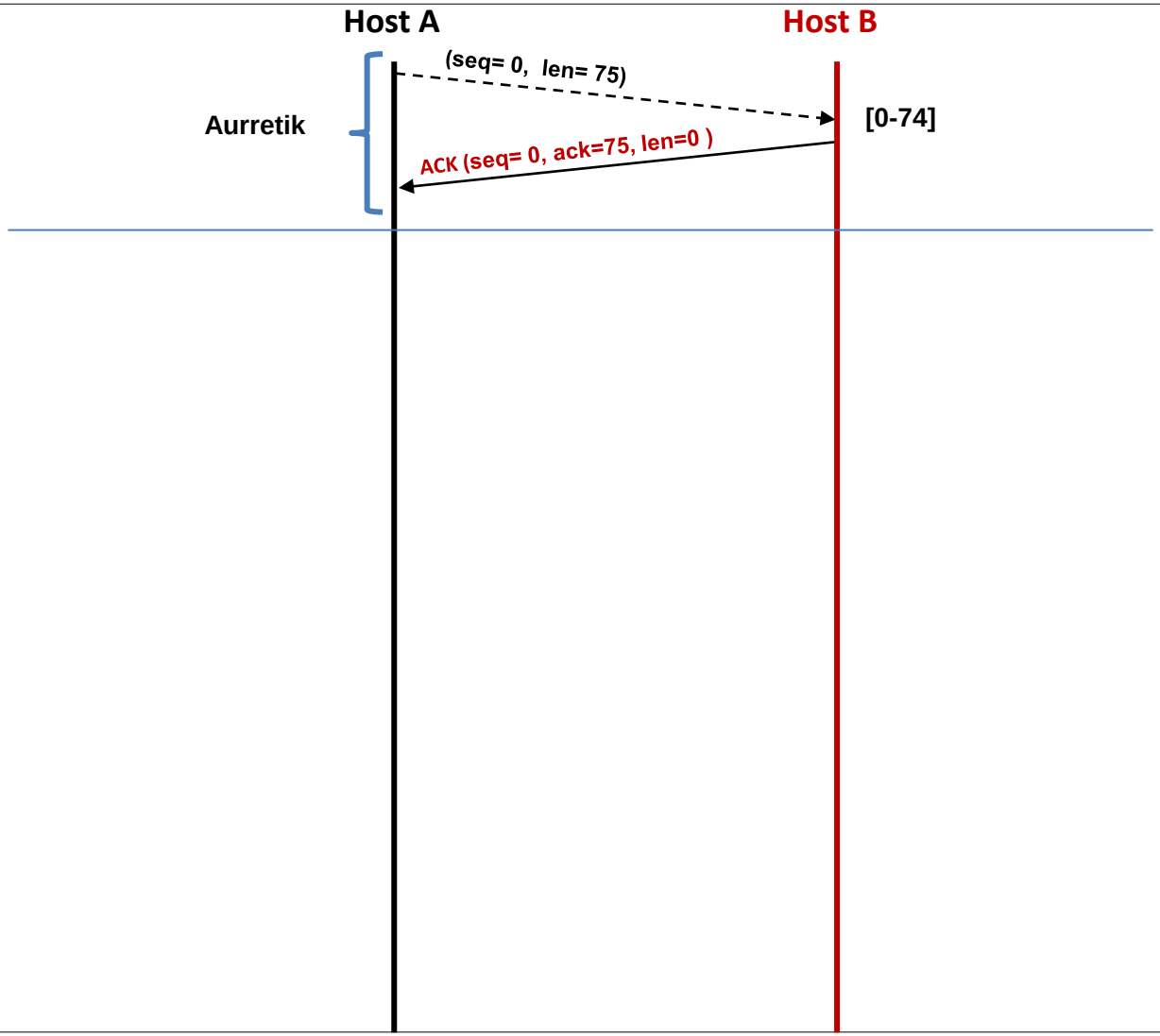
X ARIKETA

- Adierazi TCP konexio hau, **transmisio eta harrera-leihoak nahiko handiak direla** suposatuz, tenporizadoreak ere nahiko handiak direla jarraian 5 datu segmentu eta dagozkien ACK ostalari hartzaileak (B ostalariak) eta igorleak (A ostalariak), hurrenez hurren, jaso ahal izateko (kanalean galerarik ez badago), **eta aitorpen selektibo**rekin.

- Ostalari hartzaileak (B) dagoeneko datuen lehen 75 byte (0tik 74ra bitartekoak) arrakastaz jaso ditu. Demagun A ostalariak 10 byteko 5 datu segmentu berriak bidaltzen dituela jarraian B ostalariari, eta lehenengo segmentua galtzen dela (bidaltzen den lehen aldia). Azkenean, 5 datu segmentuak B ostalariak ongi jaso ditu.

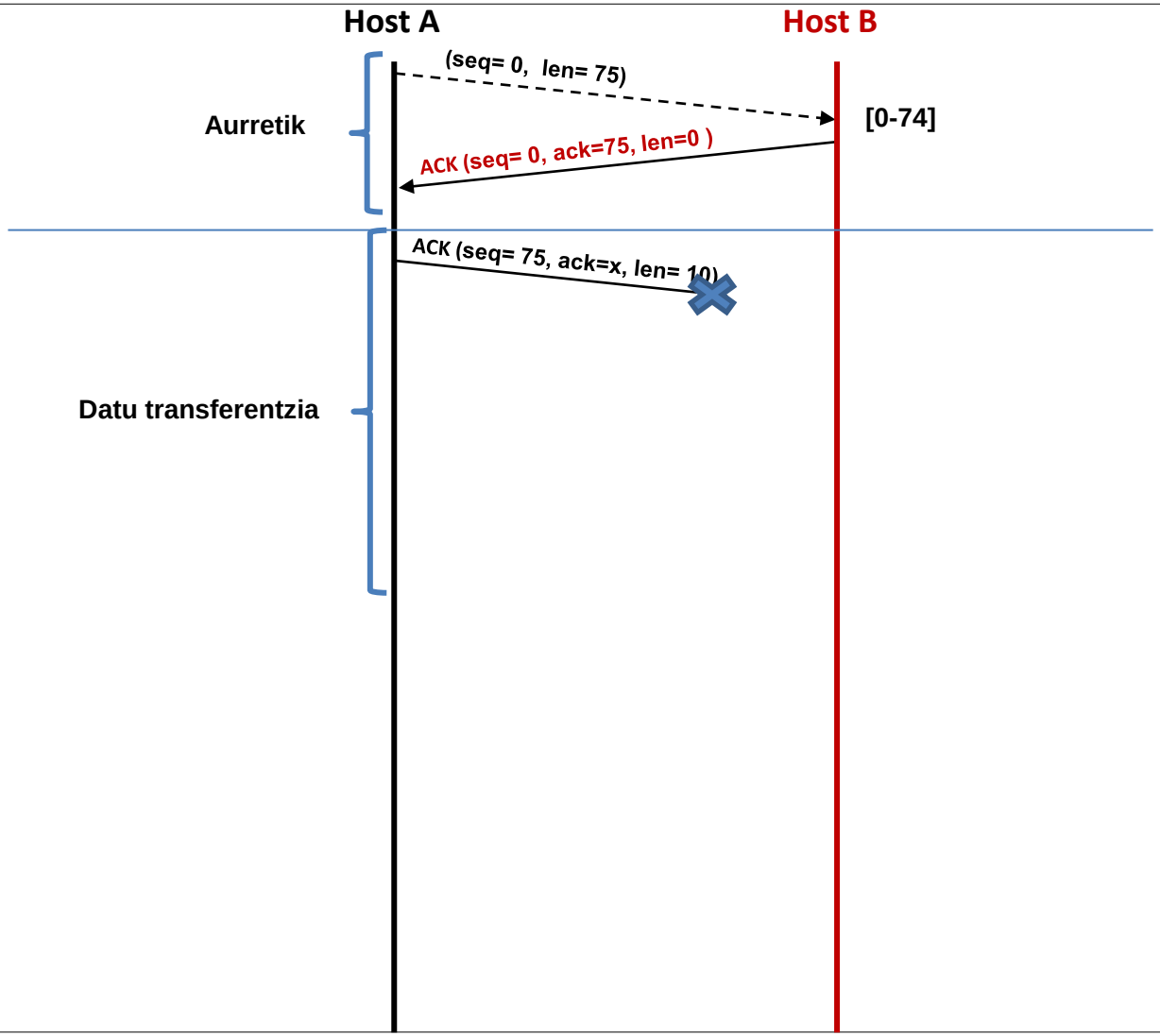
- Segmentu eta ACK trukaketa grafikoki adierazi, beti segmentuen sekuentzia zenbakiak eta ACK zenbakiak jarriz, baita tenporizadoreak ere. Jasotako segmentuak harreran nola kudeatzen diren ere adierazi (adibidez, bikoiztuak badaude, zer egin)

X ARIKETA



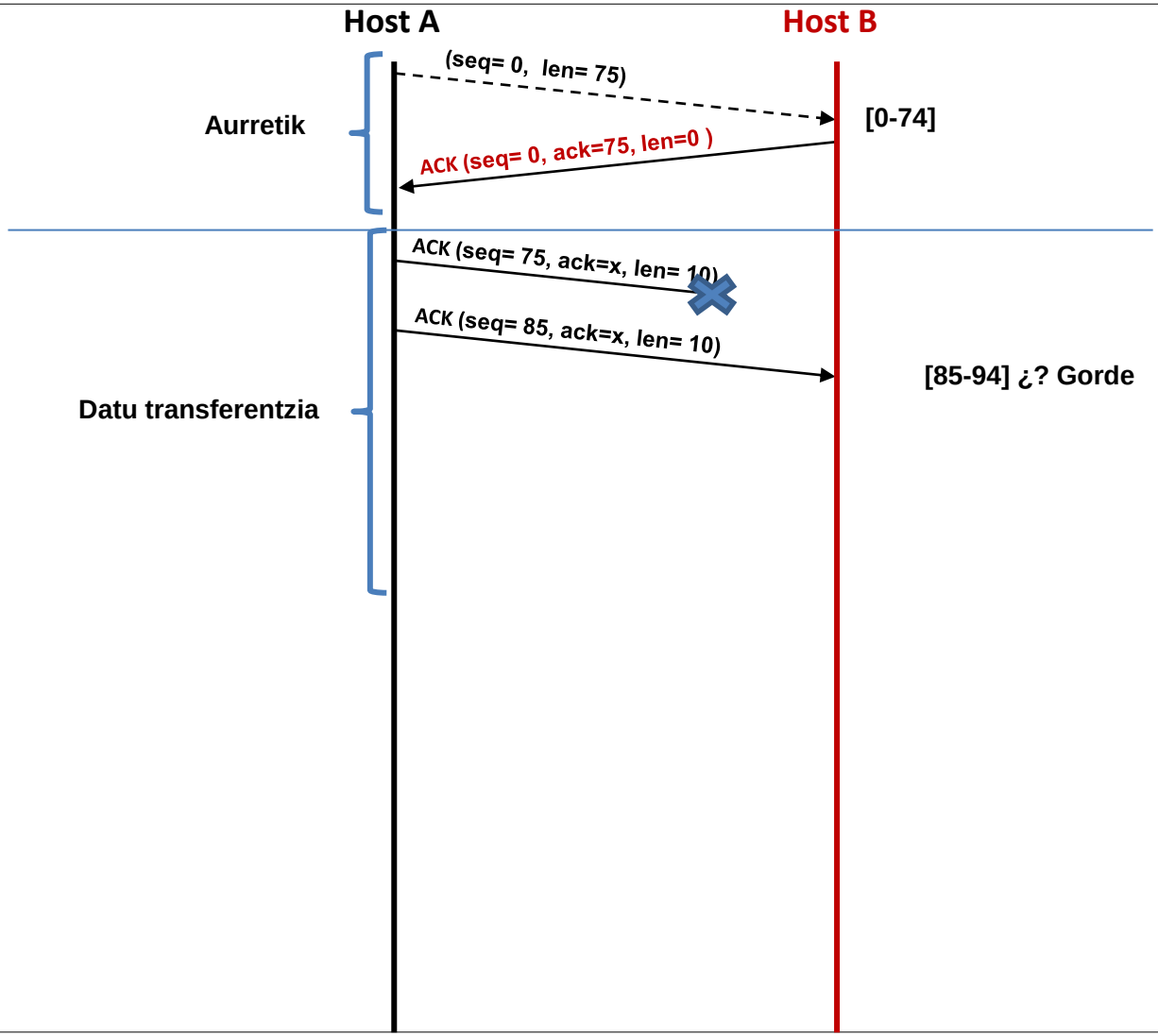
Gaizka Saldaña
Dpto. Ingeniería de Sistemas y Automática

X ARIKETA

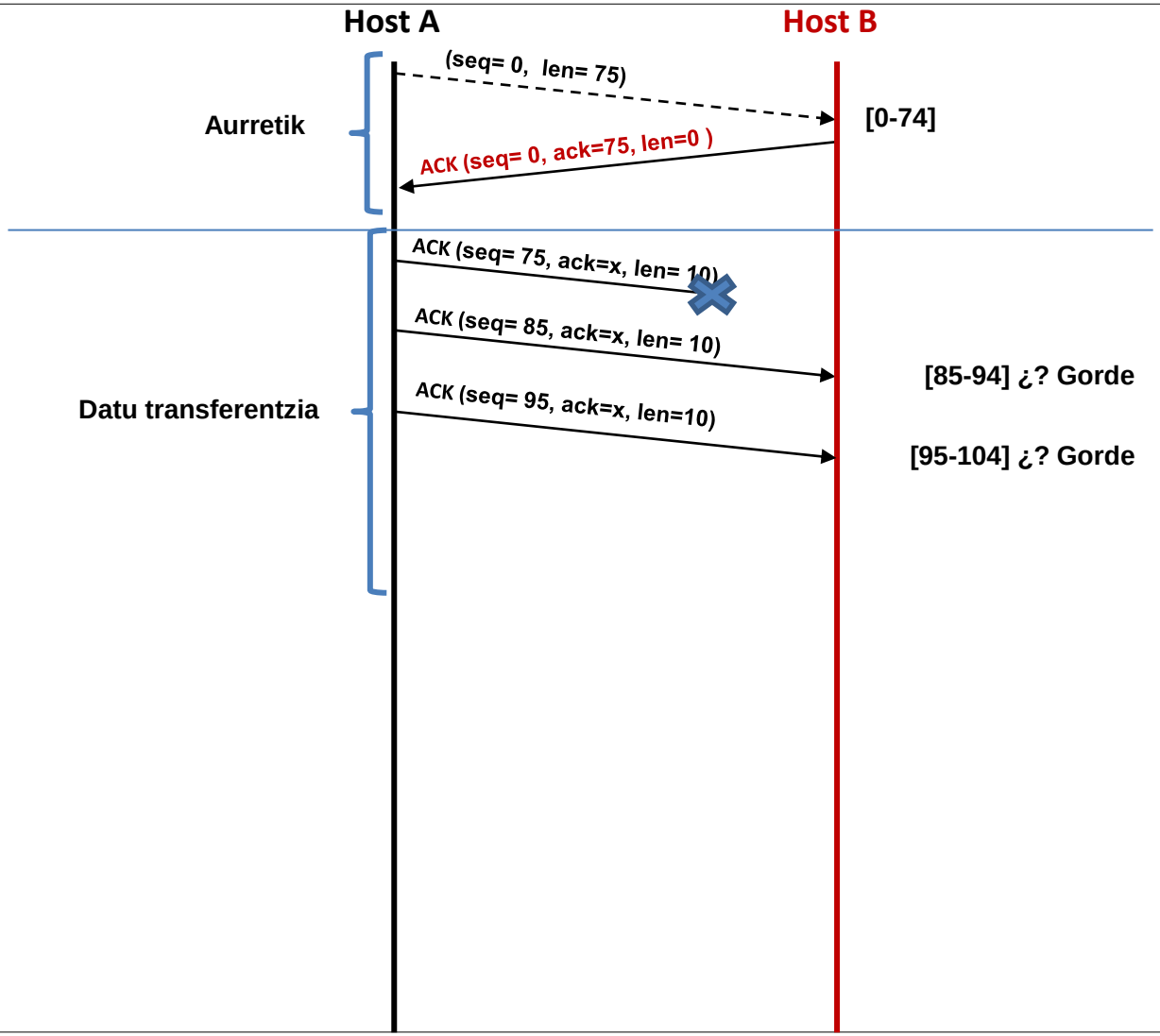


Gaizka Saldaña
Dpto. Ingeniería de Sistemas y Automática

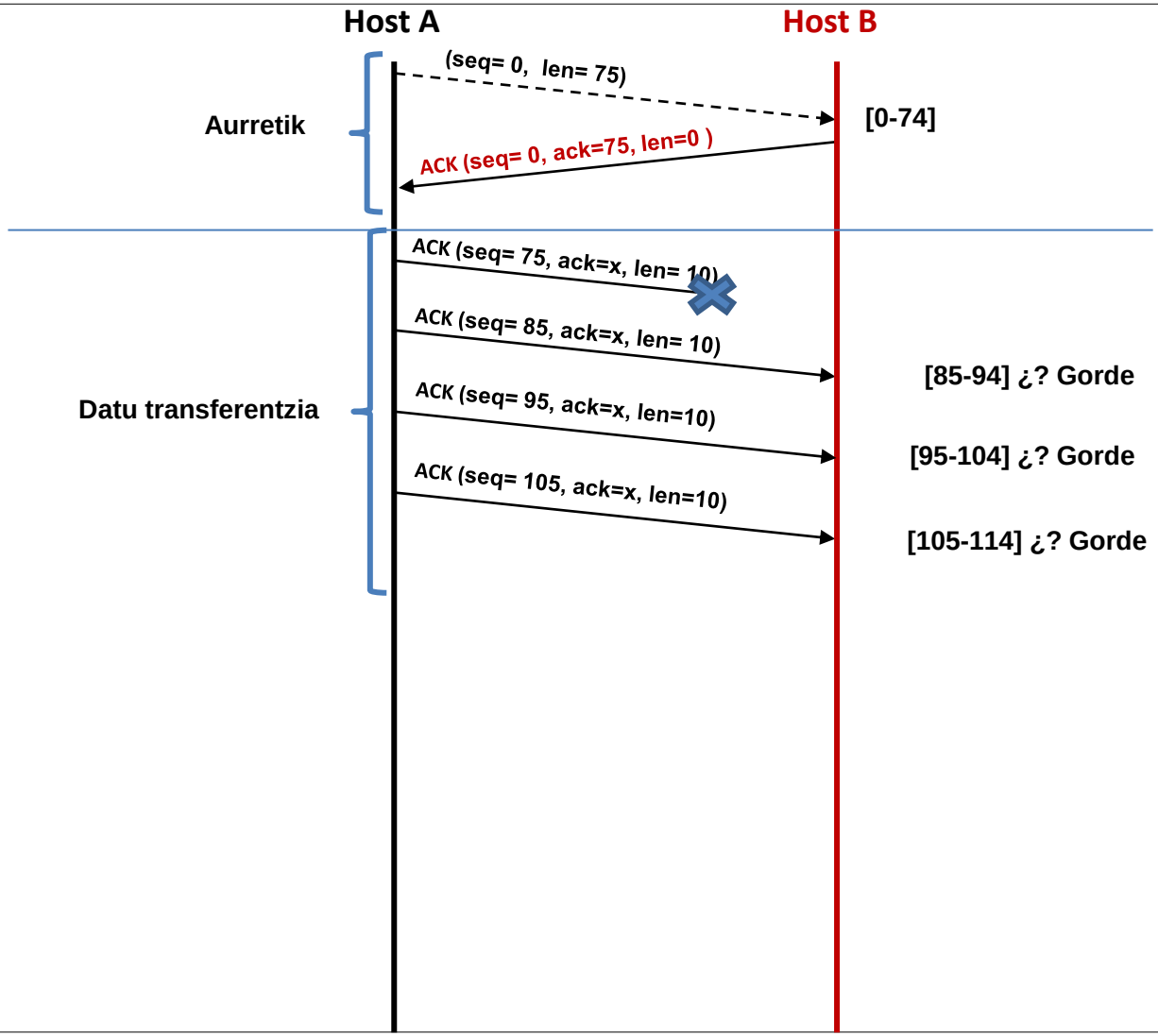
X ARIKETA


Host A
Host B
(seq= 0, len= 75)
[0-74]
Aurretik
ACK (seq= 0, ack=75, len=0)
ACK (seq= 75, ack=x, len= 10)
ACK (seq= 85, ack=x, len= 10)
[85-94] ¿? Gorde
Datu transferentzia

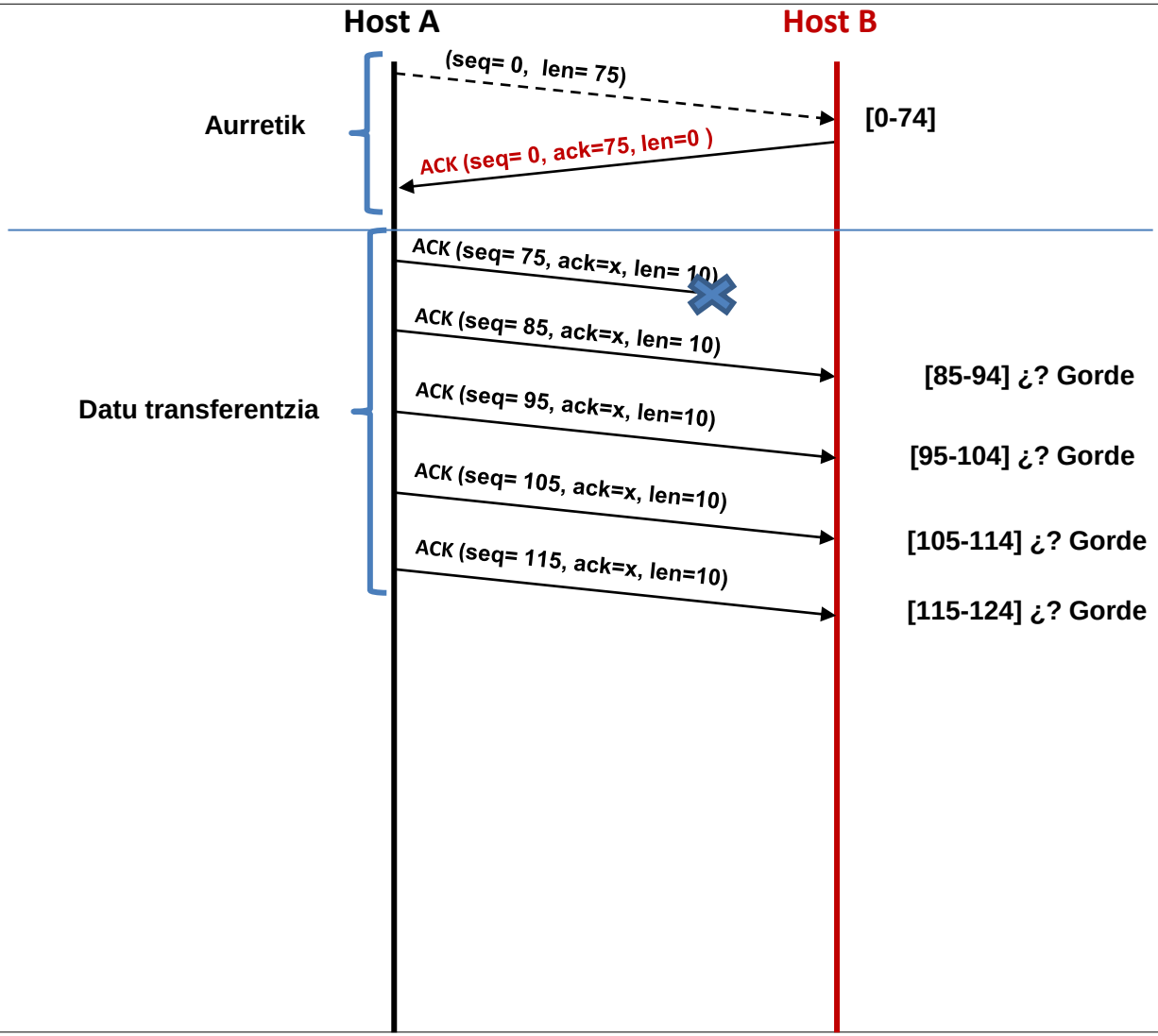
X ARIKETA


Host A
Host B
(seq= 0, len= 75)
[0-74]
Aurretik
ACK (seq= 0, ack=75, len=0)
ACK (seq= 75, ack=x, len= 10)
ACK (seq= 85, ack=x, len= 10)
[85-94] ¿? Gorde
Datu transferentzia
ACK (seq= 95, ack=x, len=10)
[95-104] ¿? Gorde

X ARIKETA

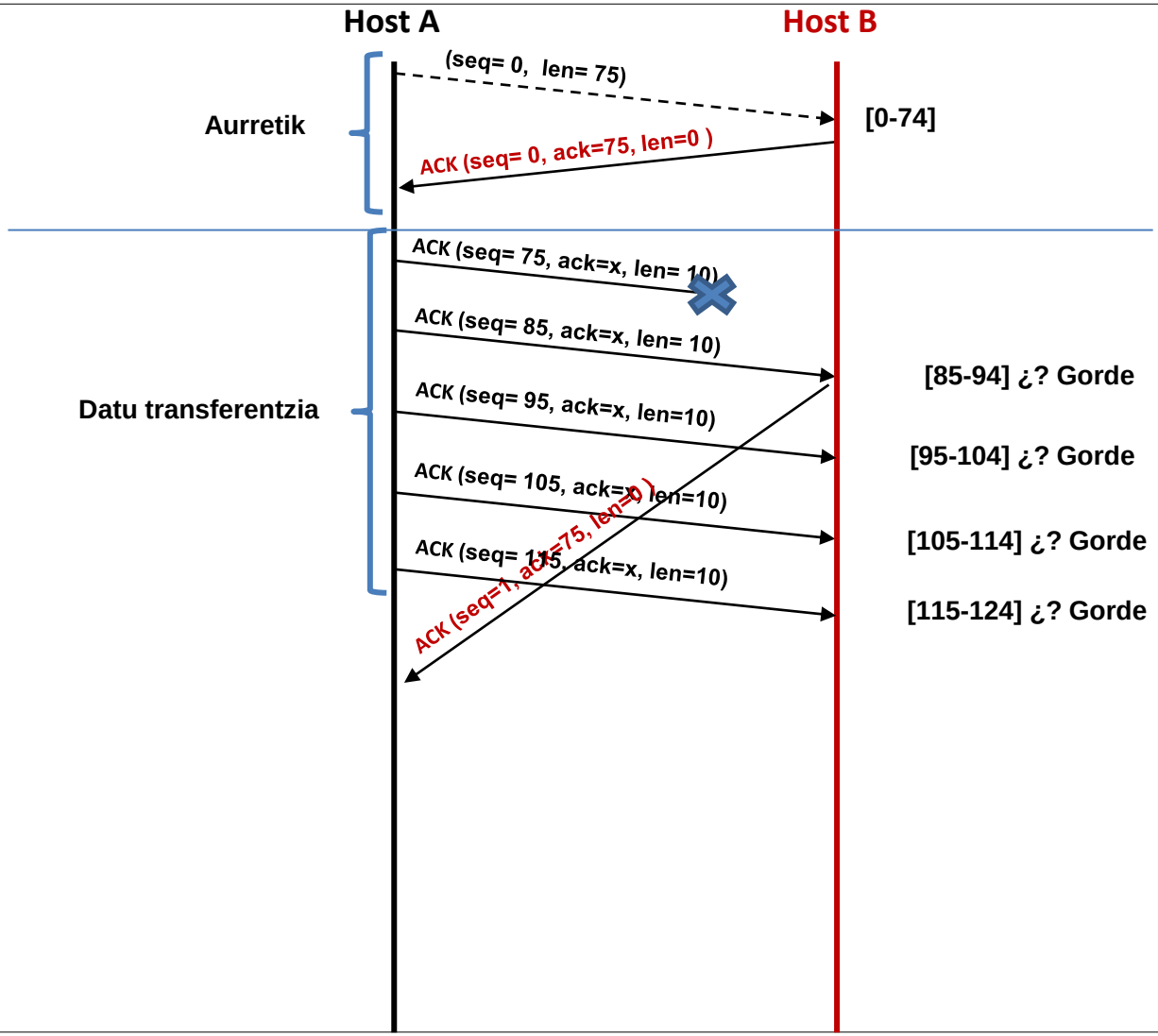

Host A
Host B
(seq= 0, len= 75)
[0-74]
Aurretik
ACK (seq= 0, ack=75, len=0)
ACK (seq= 75, ack=x, len= 10)
ACK (seq= 85, ack=x, len= 10)
[85-94] ¿? Gorde
Datu transferentzia
ACK (seq= 95, ack=x, len=10)
[95-104] ¿? Gorde
ACK (seq= 105, ack=x, len=10)
[105-114] ¿? Gorde

X ARIKETA



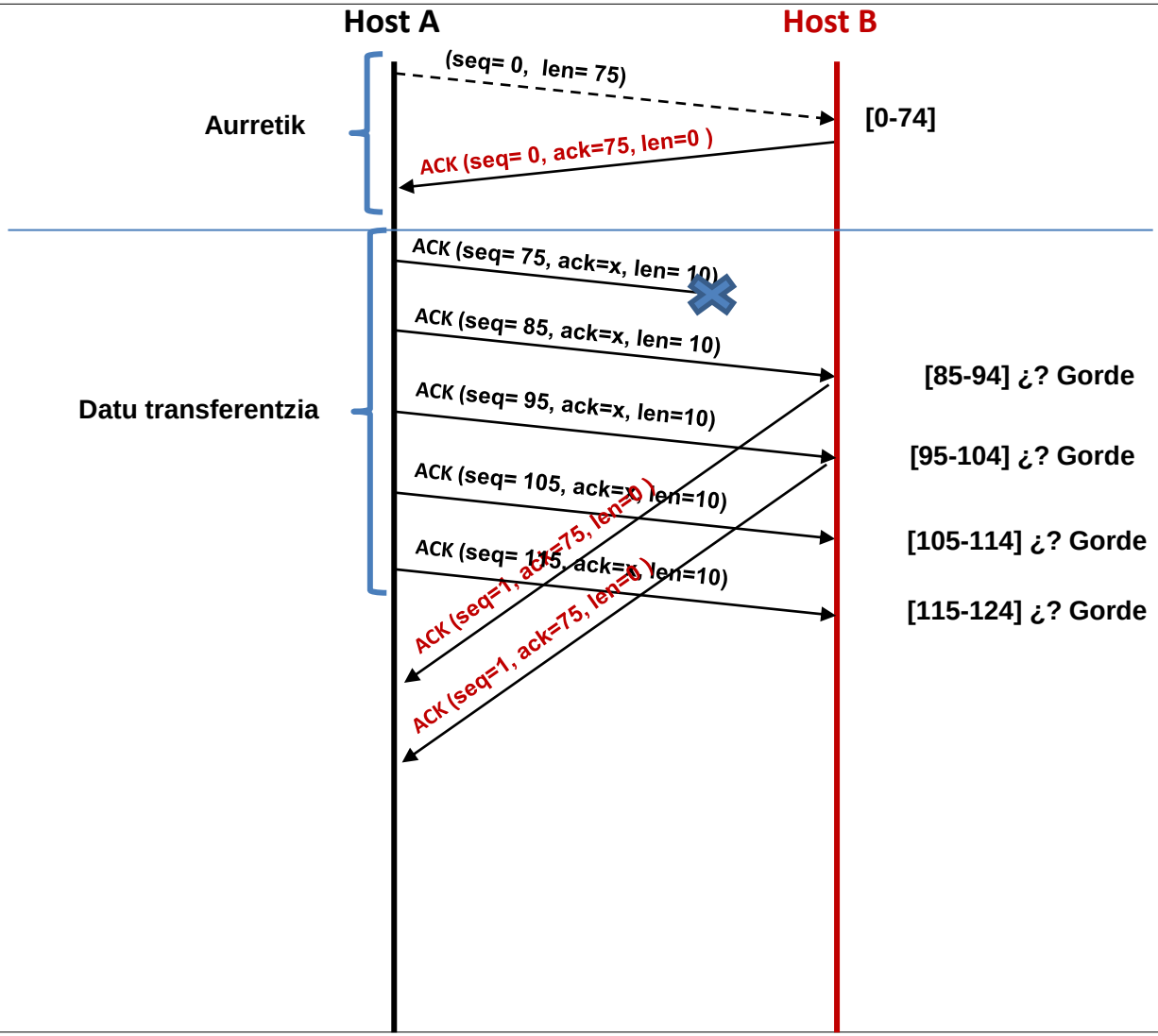
Gaizka Saldaña
Dpto. Ingeniería de Sistemas y Automática

X ARIKETA



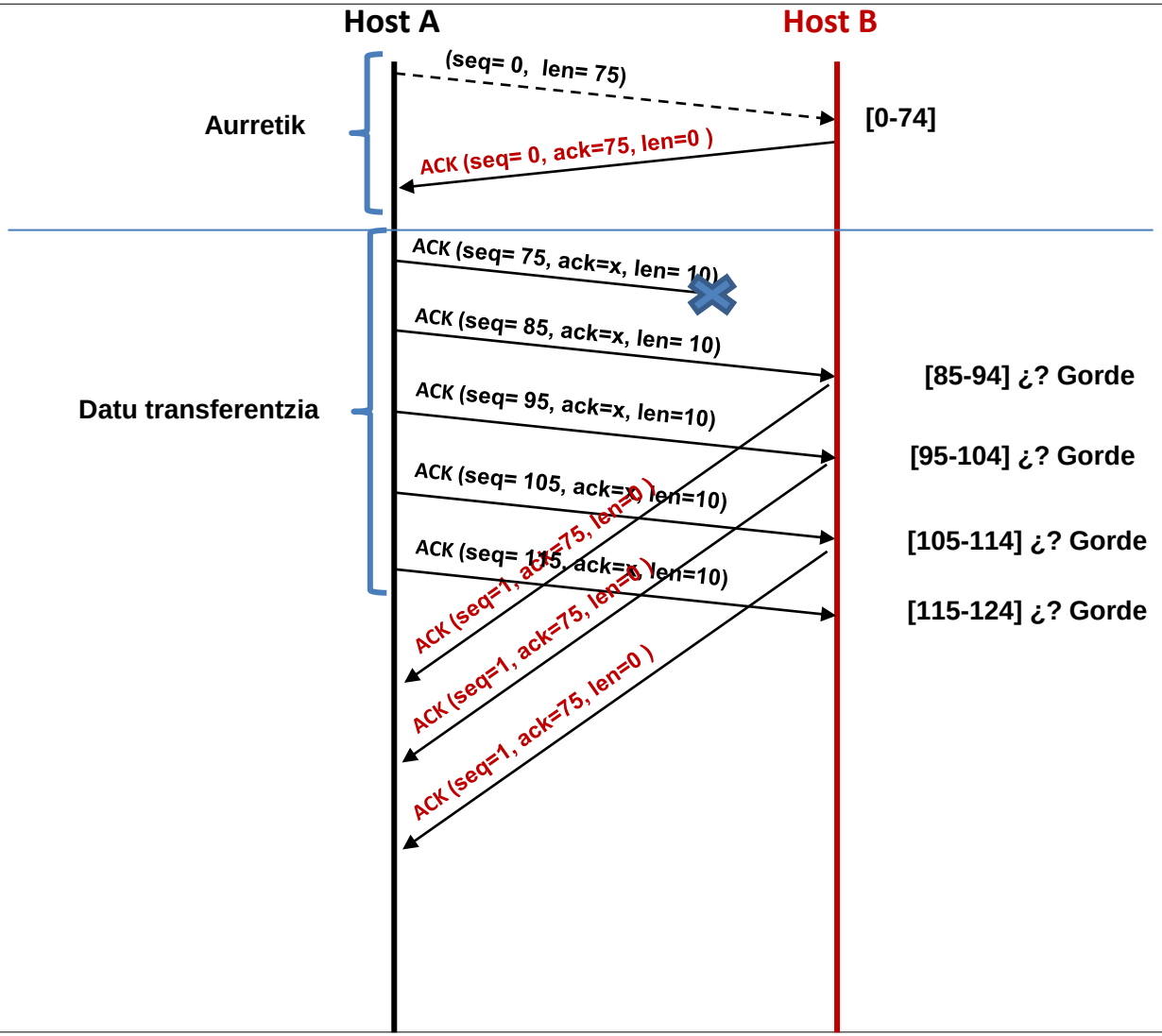
Gaizka Saldaña
Dpto. Ingeniería de Sistemas y Automática

X ARIKETA



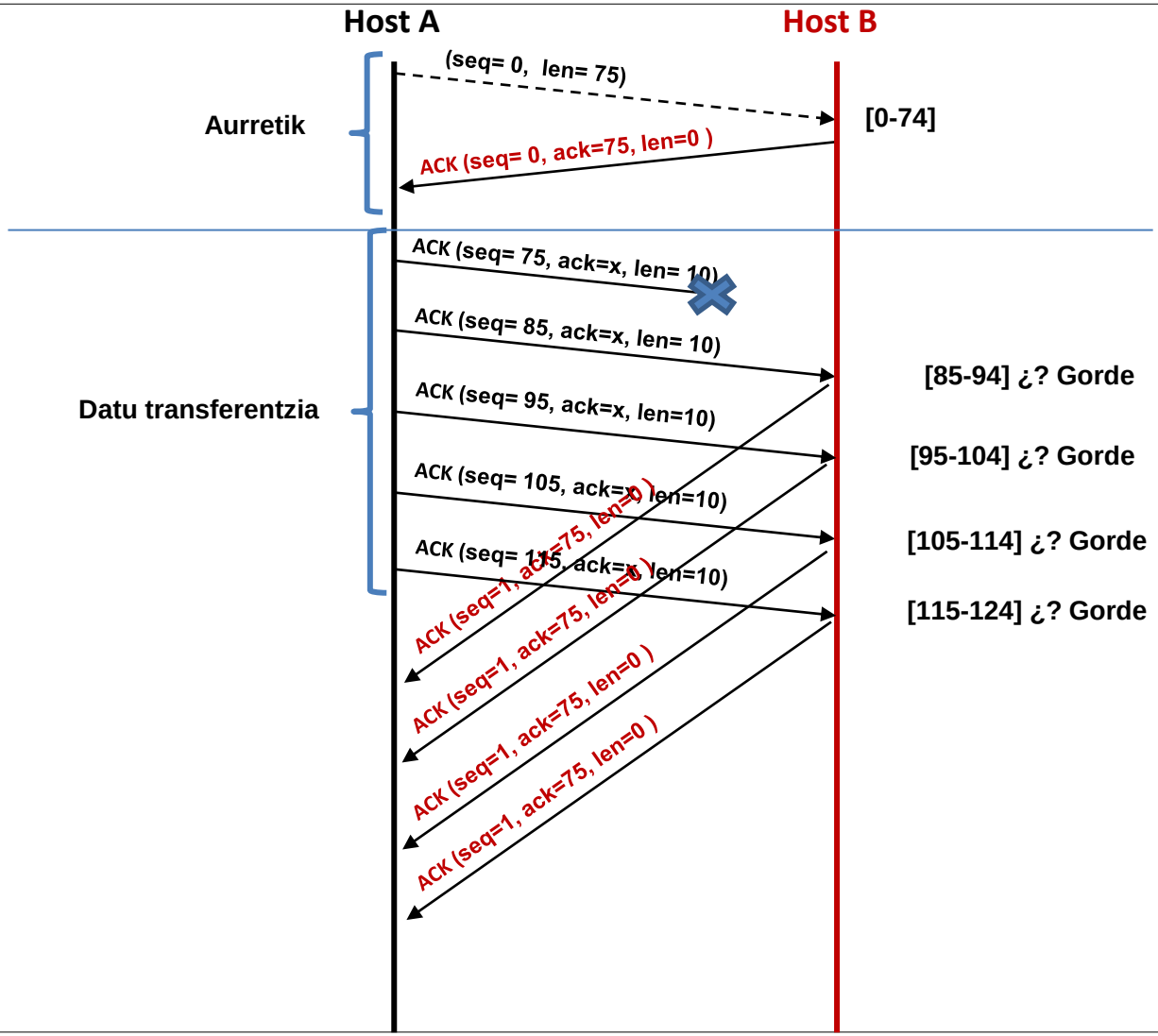
Gaizka Saldaña
Dpto. Ingeniería de Sistemas y Automática

X ARIKETA



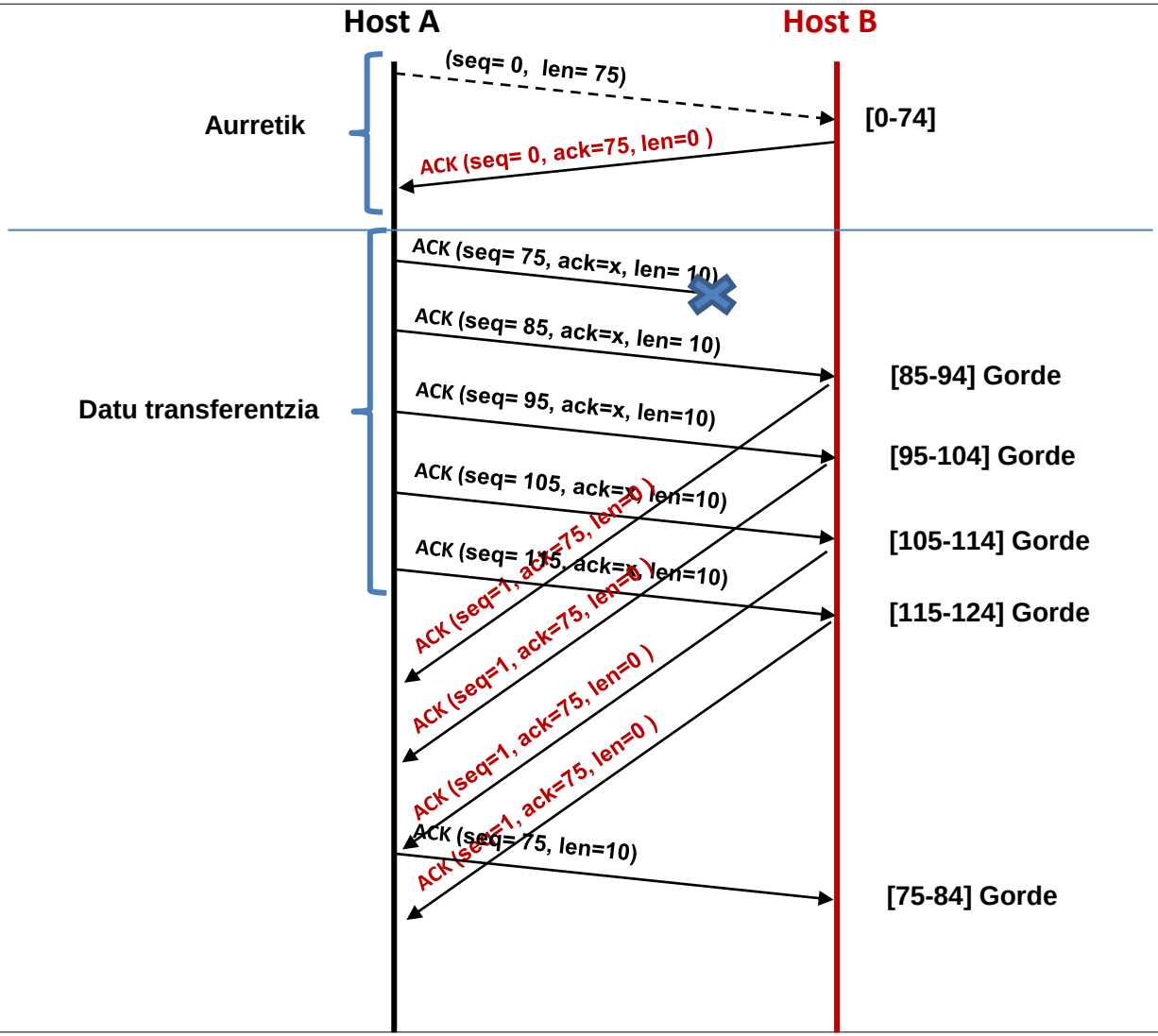
Gaizka Saldaña
Dpto. Ingeniería de Sistemas y Automática

X ARIKETA

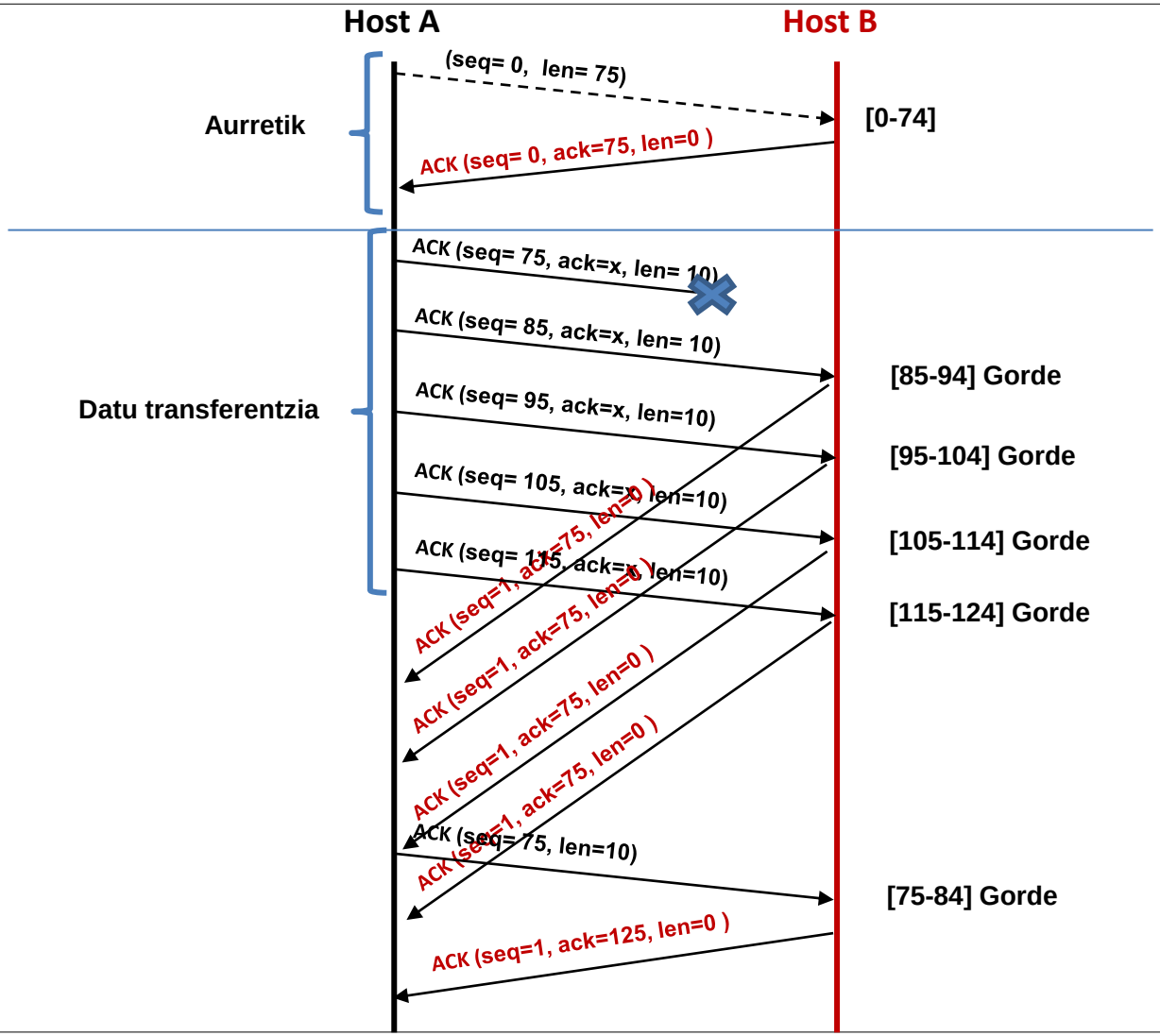


Gaizka Saldaña
Dpto. Ingeniería de Sistemas y Automática

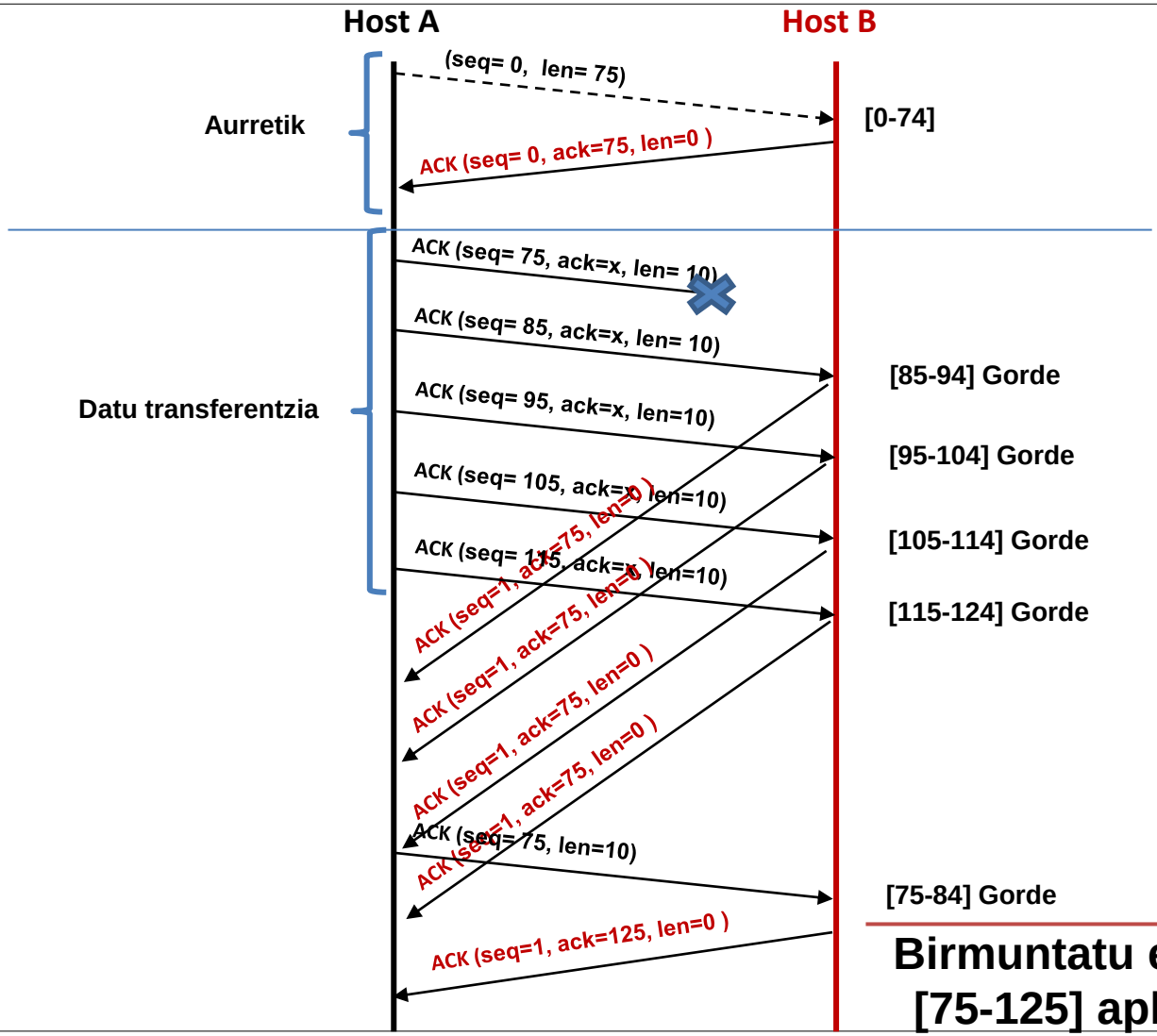
X ARIKETA


Host A
Host B
Aurretik
(seq= 0, len= 75)
[0-74]
ACK (seq= 0, ack=75, len=0)
Datu transferentzia
ACK (seq= 75, ack=x, len= 10)
ACK (seq= 85, ack=x, len= 10)
[85-94] Gorde
ACK (seq= 95, ack=x, len=10)
[95-104] Gorde
ACK (seq= 105, ack=x, len=10)
[105-114] Gorde
ACK (seq= 115, ack=x, len=10)
[115-124] Gorde
ACK (seq=1, ack=75, len=0)
ACK (seq=1, ack=75, len=0)
ACK (seq=1, ack=75, len=0)
ACK (seq=1, ack=75, len=0)
ACK (seq= 75, len=10)
[75-84] Gorde

X ARIKETA


Host A
Host B
Aurretik
(seq= 0, len= 75)
[0-74]
ACK (seq= 0, ack=75, len=0)
Datu transferentzia
ACK (seq= 75, ack=x, len= 10)
ACK (seq= 85, ack=x, len= 10)
[85-94] Gorde
ACK (seq= 95, ack=x, len=10)
[95-104] Gorde
ACK (seq= 105, ack=x, len=10)
[105-114] Gorde
ACK (seq= 115, ack=x, len=10)
[115-124] Gorde
ACK (seq=1, ack=75, len=0)
ACK (seq=1, ack=75, len=0)
ACK (seq=1, ack=75, len=0)
ACK (seq=1, ack=75, len=0)
ACK (seq= 75, len=10)
[75-84] Gorde
ACK (seq=1, ack=125, len=0)

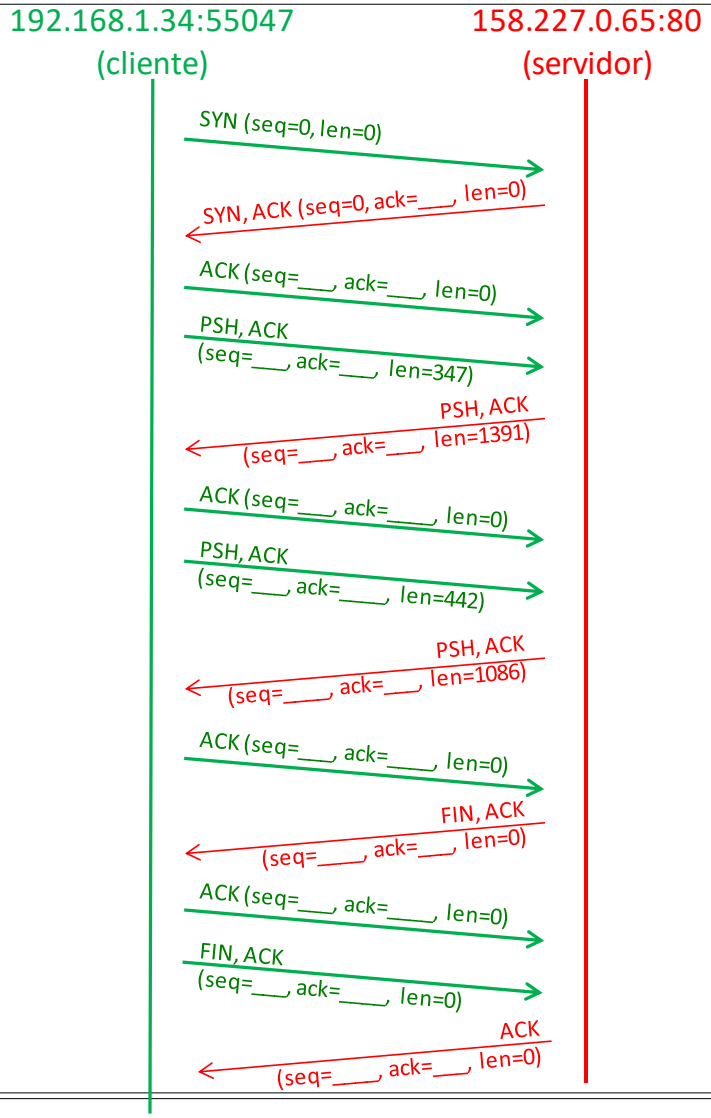
X ARIKETA



Birmuntatu eta datu guztiak [75-125] aplikaziora pasatu

PROPOSATURIKO ARIKETA 2019 EKAINA (PA)

- **Honako TCP fluxua bezero baten eta zerbitzariaren artean gertatzen da. Osatu sekuentzia eta aitorpen zenbakiak. Diagrama erabiliz, azaldu TCP saioak.**



Gaizka Saldaña
Dpto. Ingeniería de Sistemas y Automática

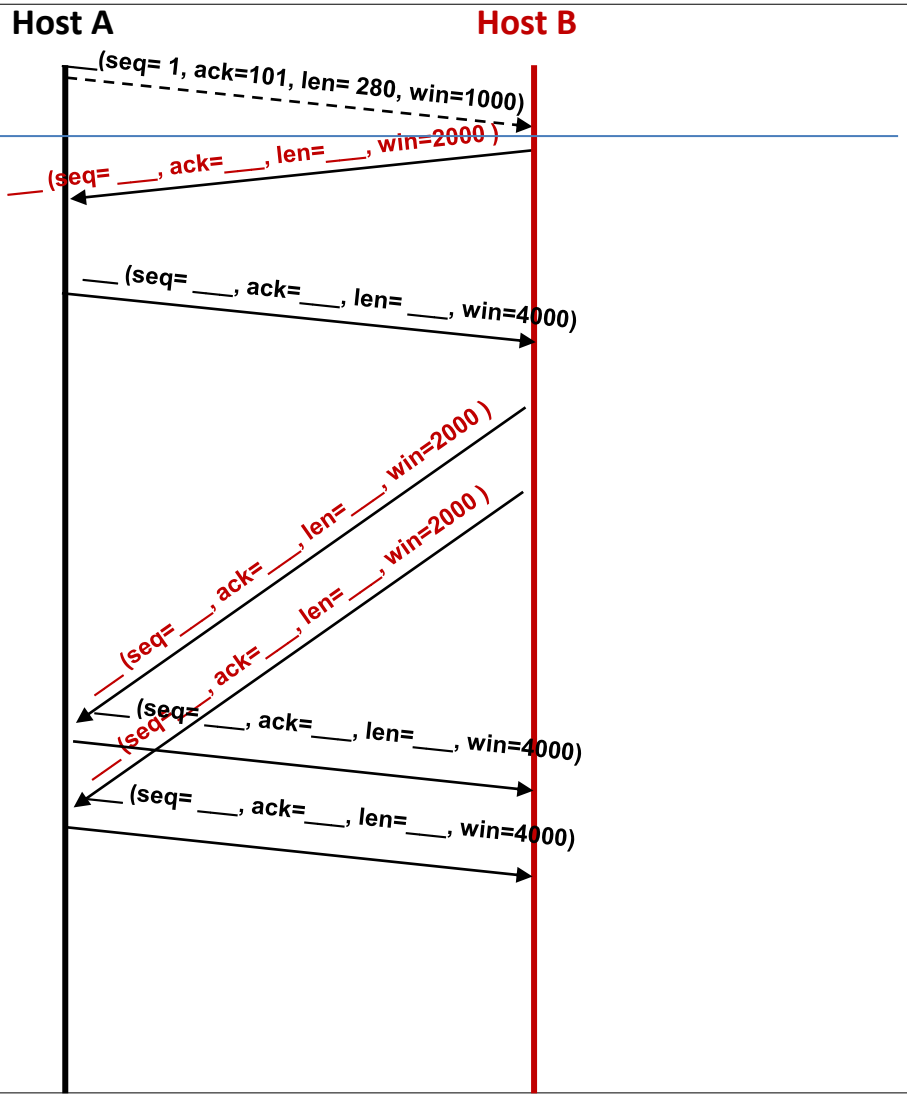
2. PROPOSATURIKO ARIKETA (PA)

- Ondorengo irudian, ordenagailutik TCP konexioa ezarri ondoren, 3600 byteko fitxategia deskargatzeko eskaera zerbitzari batetik abiatzeko unea erakusten da.
- Osatu TCP segmentu bakoitzean adierazten duen denbora-diagrama, beharrezkoa denean:
  - zein “flag" aktibatuko diren; "Sekuentzia zenbakia" (Seq) eta "Aitorpen zenbakia" (Ack) eremuen balioak; eta segmentuak duen datu kopurua (len).
- Bai PC1entzat eta bai zerbitzariarentzat, gehieneko segmentuaren tamaina (TMS) 1300 byte da.
- Errespetatu igorritako segmentu bakoitzean deklaratzen den bidalketa-leihoa (WIN)

2. PROPOSATURIKO ARIKETA (PA)

Eskaera

Deskarga



3. PROPOSATURIKO BESTE ARIKETA (PA)

HTTP bezero batek (C) 1200 byteko HTML orri bat deskargatzea eskatzen du web zerbitzari batetik (S). Hasieran, bi TCPen leihoen neurriak 600 bytekoak dira, gehieneko segmentu tamaina (MSS) 300 bytekoa da (ez du TCP goiburua barne). Zerbitzariak (S) hainbat segmentu jarraian bidaltzen dituenean, bezeroak (C) jasotako datuen aitorpen globala bidaltzen du, behin bi segmentu bakoitzeko. Bezeroaren eta zerbitzariaren arteko segmentuen trukaketa deskribatu, ondoren azaldutako egoera bakoitzean.

a) Bezeroak zerbitzariarekin konexioa ezartzen du. Aukeratutako hasierako sekuentzia zenbakiak hauek dira: Bezeroa 2000; zerbitzaria 4000.

b) Bezeroak eskaera zerbitzarira bidaltzen du (300 byte) eta zerbitzariak eskatutako HTML orriarekin erantzuten du

c) Azkenik, zerbitzariak konexioa ixten du.

2018 UZTAILAREN ETA 2019 UZTAILAREN AZTERKETEN ARIKETA

TCP konexio erraz bat argi adierazi

2017 MAIATZAREN AZTERKETAREN ARIKETA

6. **(1 pto)** Python bezero batek HTTP eskaera hauek bidali eta jasotzen ditu:

```
POST https://egela1516.ehu.eus/login/index.php/ HTTP/1.1
Host: egela1516.ehu.eus
Content-Type: application/x-www-form-urlencoded
Content-Length: 40

username=emartinez001&password=123456789
```

```
HTTP/1.1 200 OK
Date: Thu, 20 Nov 2015 20:25:52 GMT
Content-Length: 16
Content-Type: text/plain

Kaixo Eduardo!!!
```

Adierazi bezeroaren eta zerbitzariaren arteko TCP fluxua konexioa amaitu arte.

Notak:

- Eskaeraren lehen lerroak eta goiburuek 144 byte luzera dute, CR eta LF karaktereak kontatu gabe.
- Erantzunaren lehen lerroak eta goiburuek 92 byte luzera dute, CR eta LF karaktereak kontatu gabe.