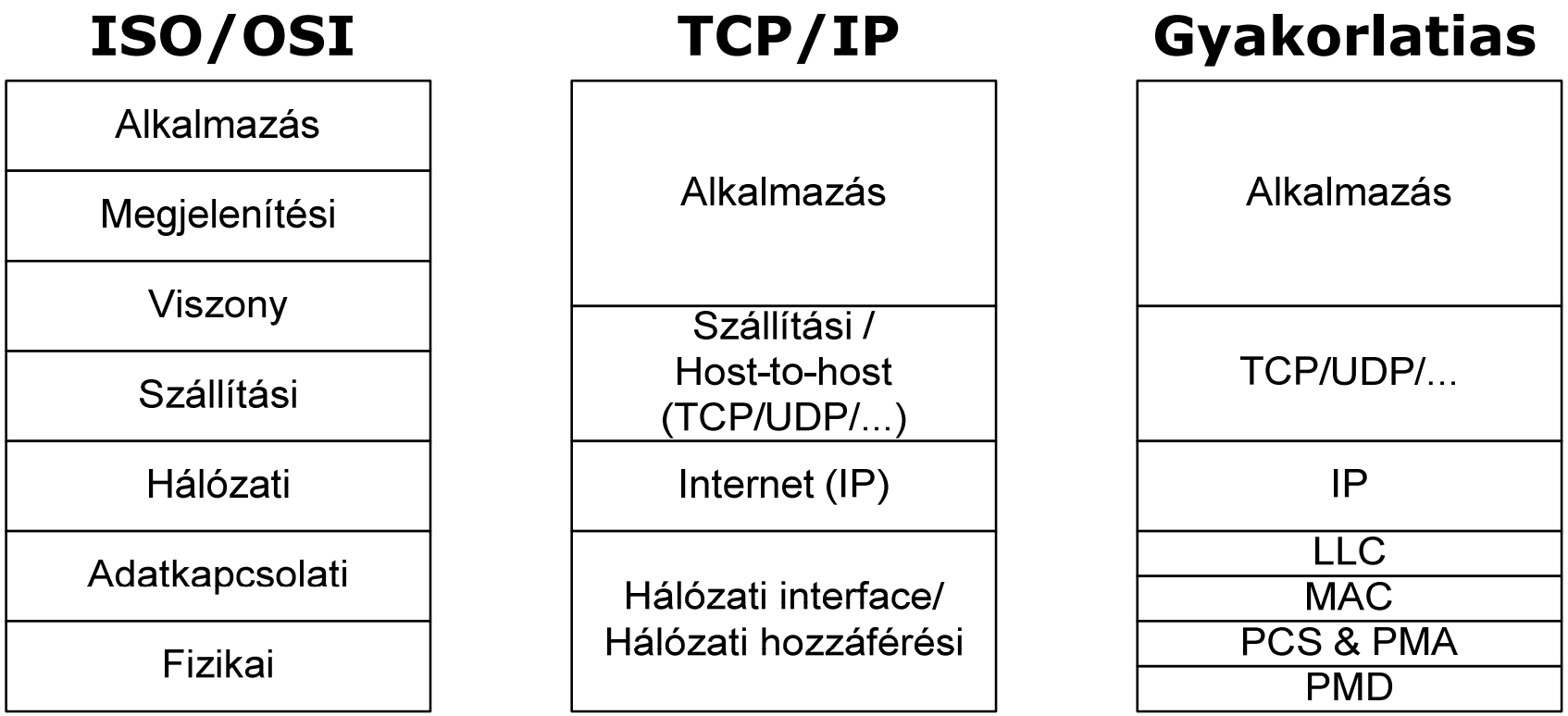
Szállítási (transport, host-to-host) protokollok

UDP és TCP

A TCP/IP architektúra és az ISO/OSI rétegmodell



IP: Internet Protocol

TCP: Transmission Control Protocol

UDP: User Datagram Protocol

LLC: Logical Link Control

MAC: Medium Access Control

PCS: Physical Coding Sublayer

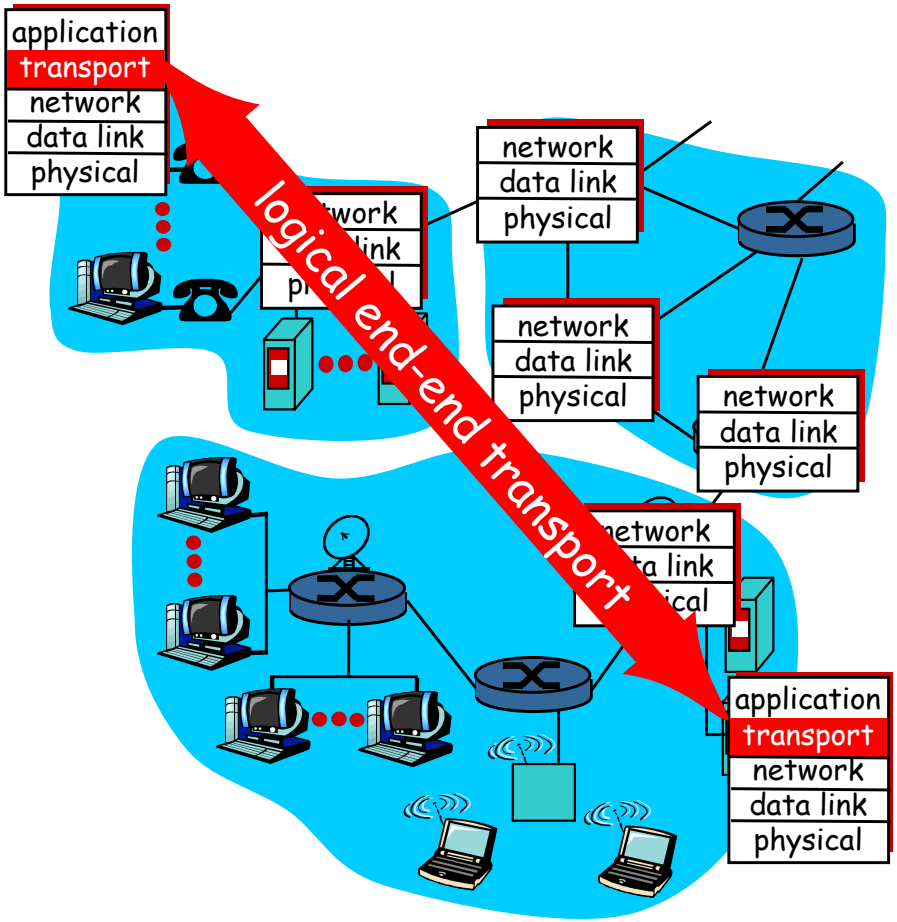
PMA: Physical Medium Attachment

PMD: Physical Medium Dependent

A hálózati és a szállítási réteg

- *hálózati réteg:* végpontok („host”-ok) közötti logikai kapcsolatok
- *szállítási réteg:* alkalmazások (process) közötti logikai kapcsolatok
  - a hálózati réteg szolgáltatásainak igénybevételére alapozva

A szállítási réteg



Logikai kapcsolatok a végpontokban futó alkalmazások között

A szállítási protokollok a végpontokban futnak, a csomópontokban nem

Az alkalmazások adategységeit szállítási protokoll-adategységekbe tördeljük, a kapottakból pedig összerakjuk

Szállítási protokollok: UDP és TCP

- UDP – User Datagram Protocol
- TCP – Transmission Control Protocol
- Az UDP és TCP közös képességei:
  - Portok kezelése
  - Multiplexelési képesség
- Alapvető különbség az UDP és a TCP között:
  - UDP összeköttetésmentes (connectionless),
  - TCP összeköttetés-alapút (connection–oriented)

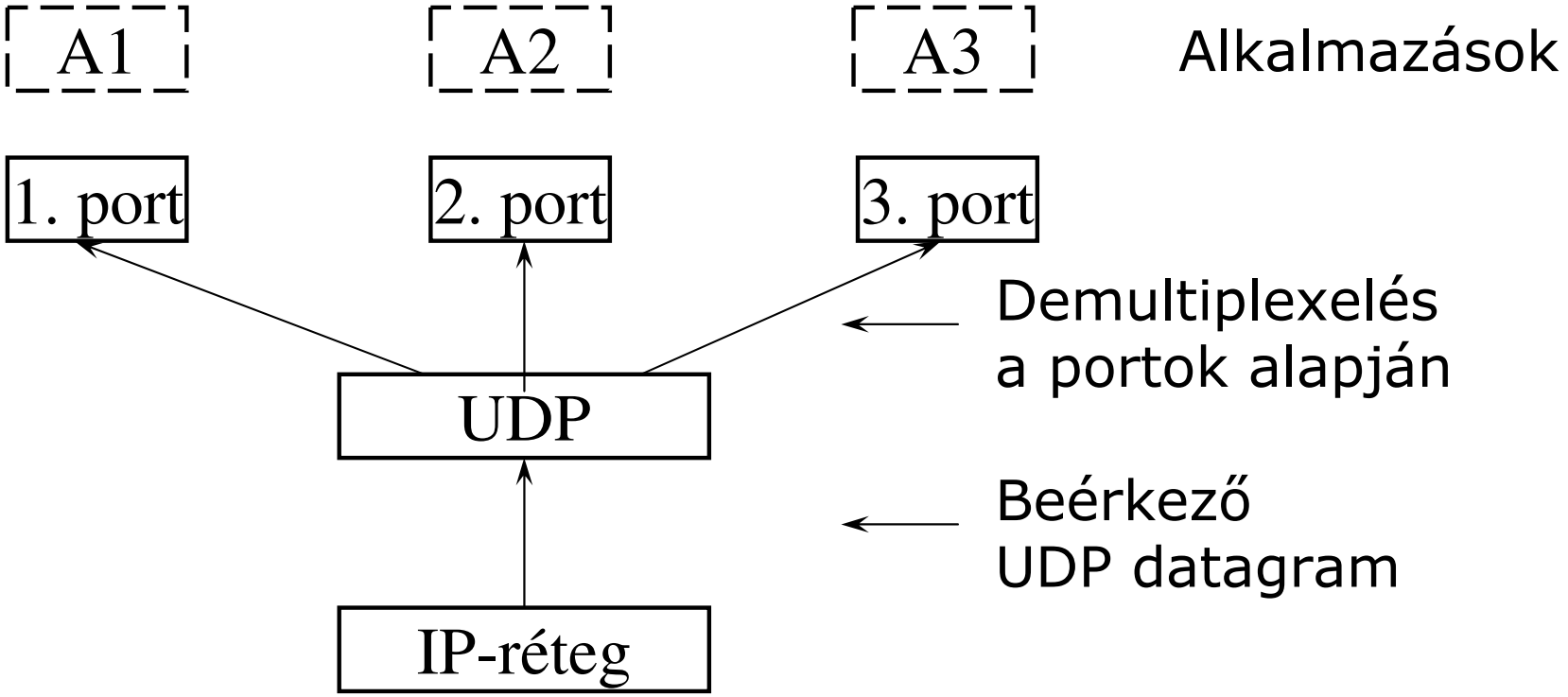
transzport-szolgáltatást nyújt

Az UDP és TCP közös képességei (1)

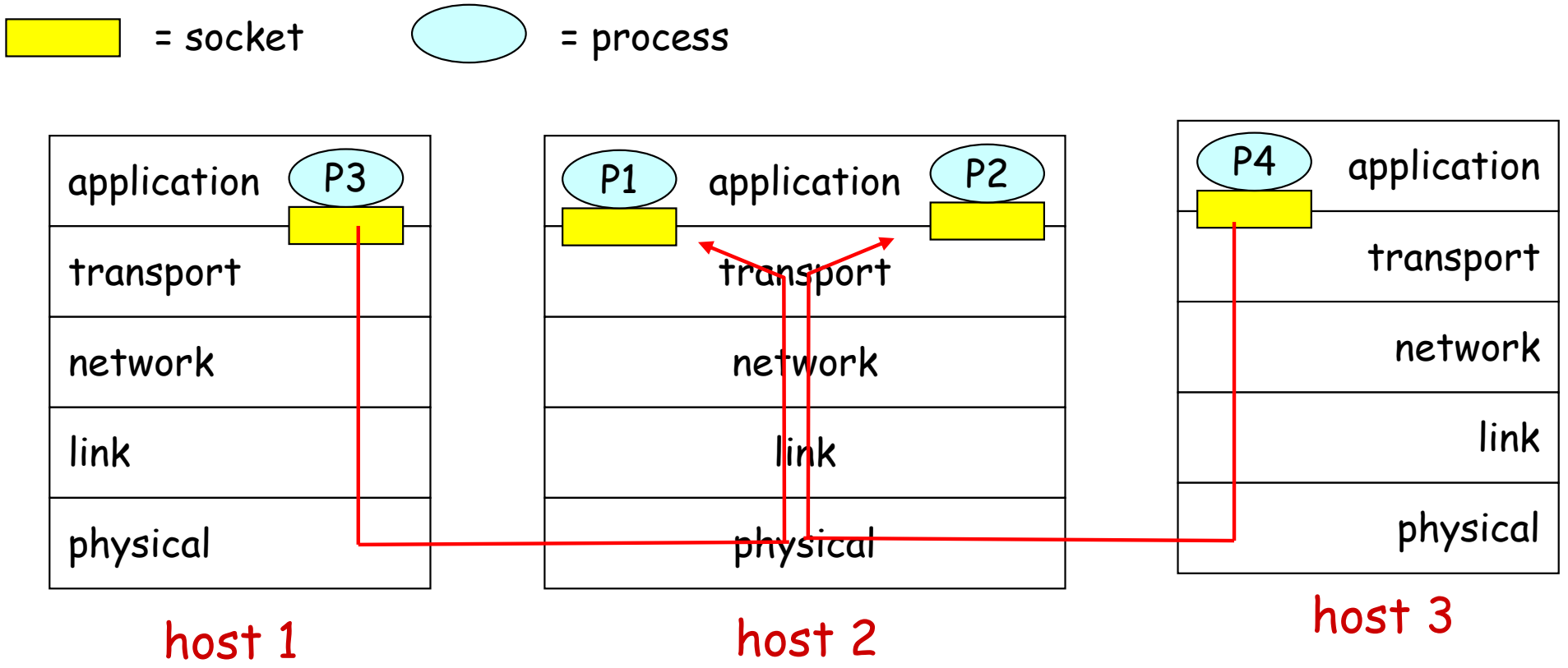
- Portok kezelése:
  - Az IP-rétegben a csomagok végpontnak, „host”-nak vannak címezve
  - A végpontokon belül: több alkalmazás, folyamat
  - Megkülönböztetésük: portok használatával
  - Foglalt (reserved, „well-known”) és rendelkezésre álló (available) portszámok
  - Foglalt portok: ide mindig lehet küldeni datagrammokat
    - 0...1023 közötti portszámok
    - pl. 80: HTTP, 21: FTP, 69: TFTP (ezek főként TCP-re)
  - Az UDP-en belül megállapításra kerülnek az alkalmazandó portszámok
- Multiplexelés /demultiplexelés
  - A portmechanizmus segítségével

Az UDP és TCP közös képességei (2)

Multiplexelés és demultiplexelés
Példa:



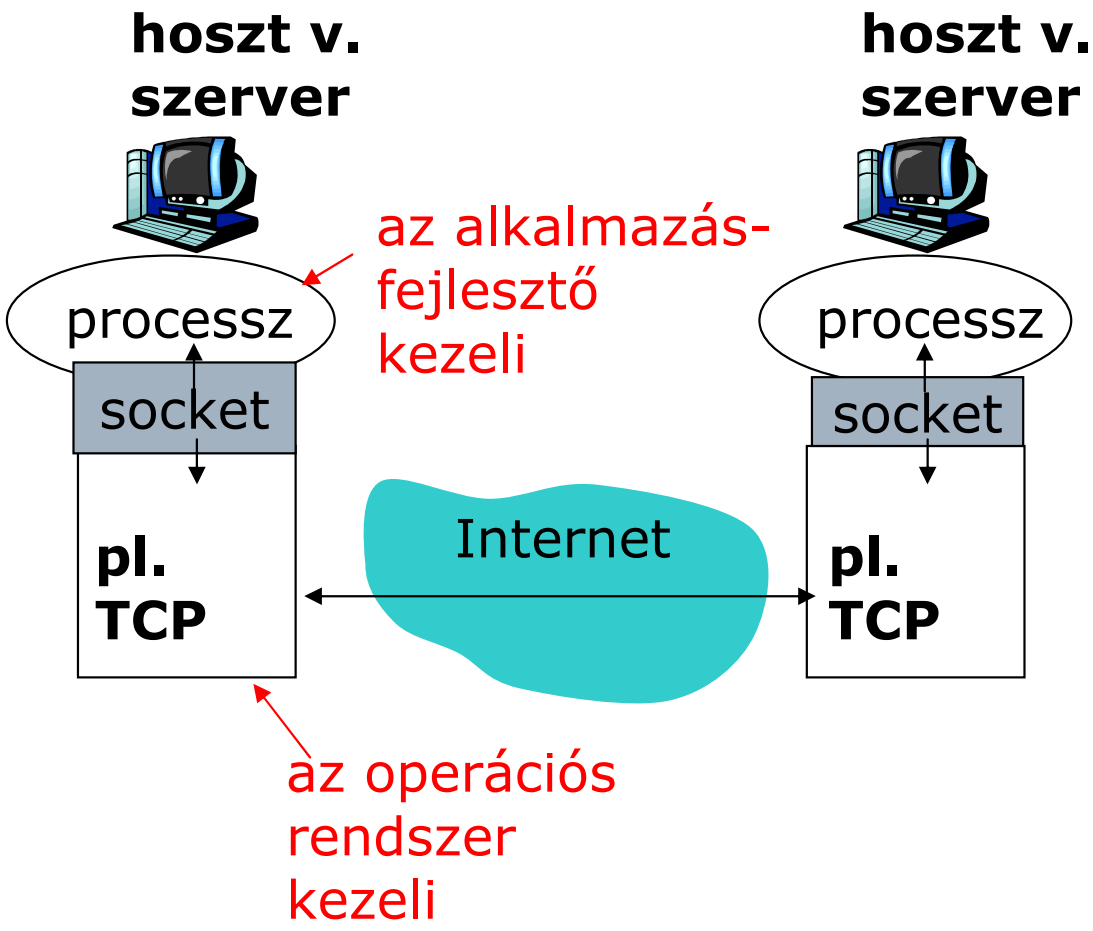
Multiplexelés-demultiplexelés



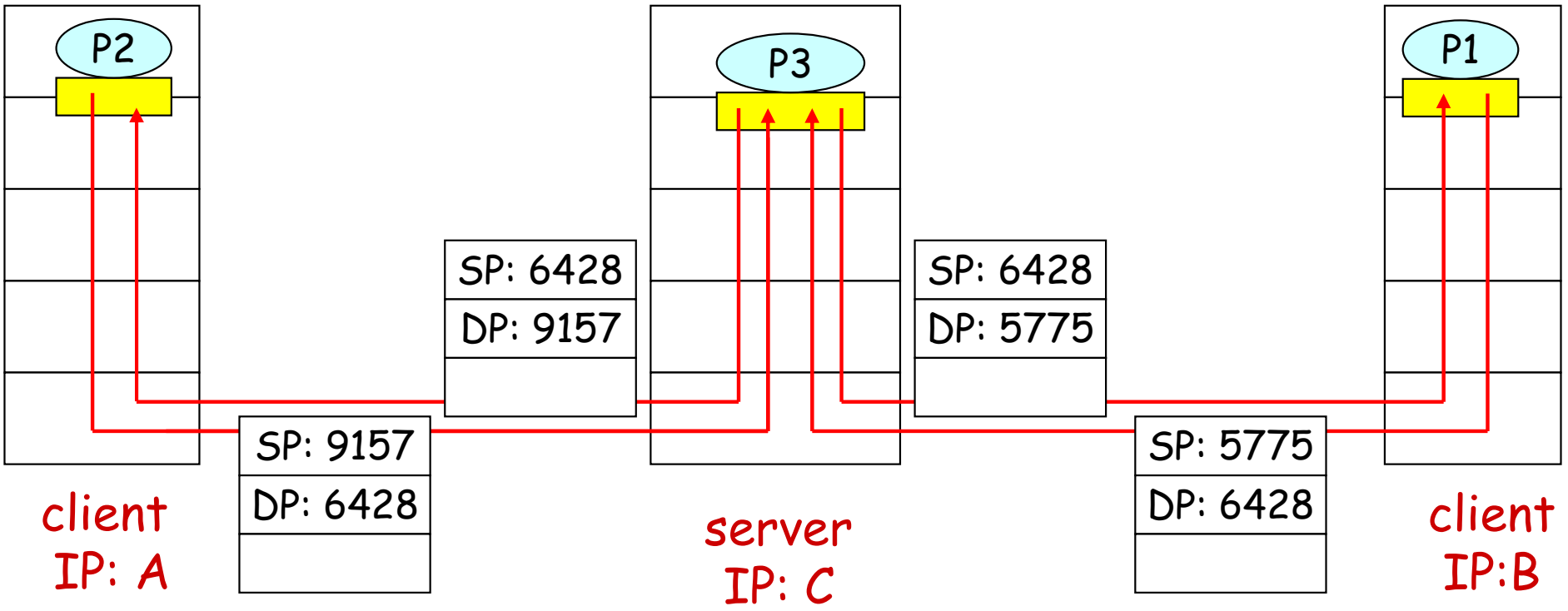
Socket: magyarázat

- Socket: interfész, „ajtó” az alkalmazás és a hálózat között
- A socket-et az alábbiak jellemzik:
  - transzportprotokoll (UDP v. TCP)
  - saját IP cím
  - saját portszám
  - *(opcionális) távoli host IP címe*
  - *(opcionális) távoli alkalmazás portszáma*
- Leegyszerűsítve: socket = IP cím + portszám
- Az operációs rendszer a bejövő IP csomagokat a fentiek alapján továbbítja az alkalmazásnak, kiszedve ezeket a megfelelő PDU-kból
- Helyi és távoli socket együtt: socket-pár

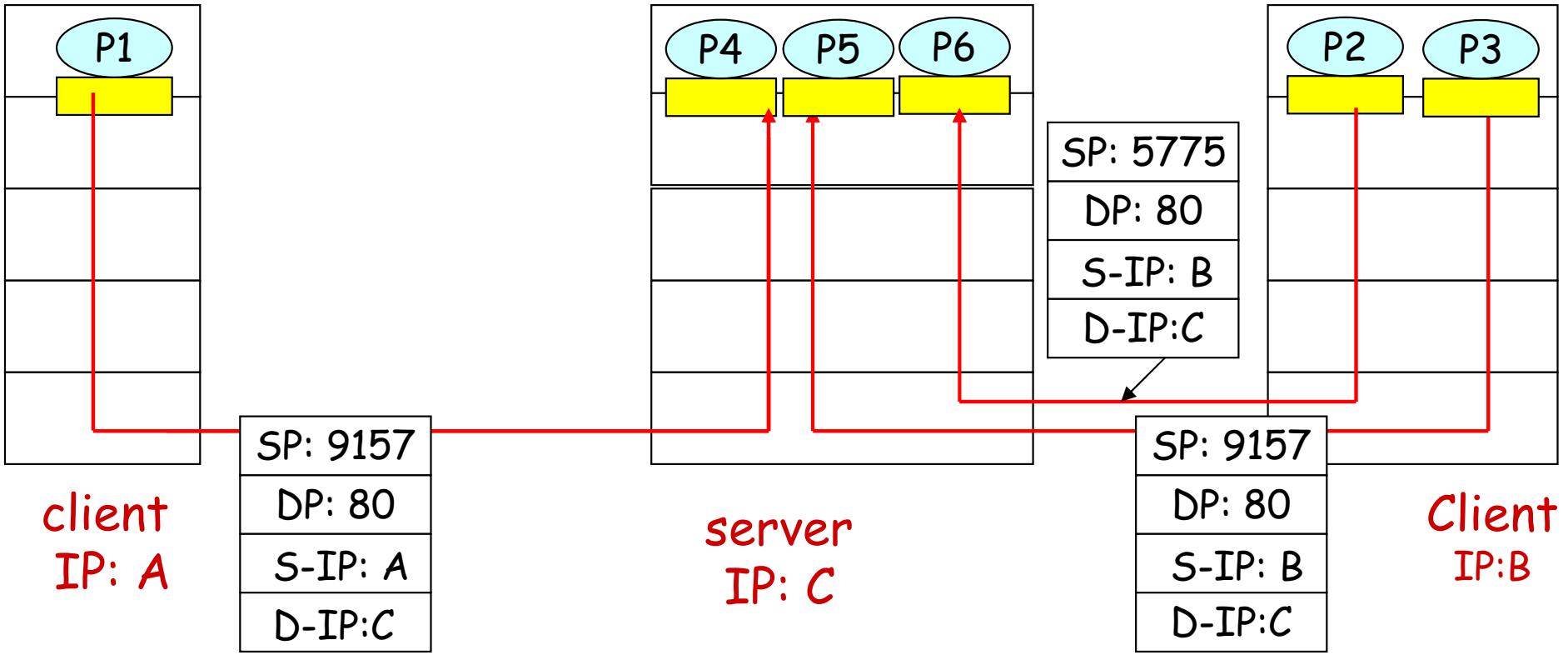
Socket: illusztráció



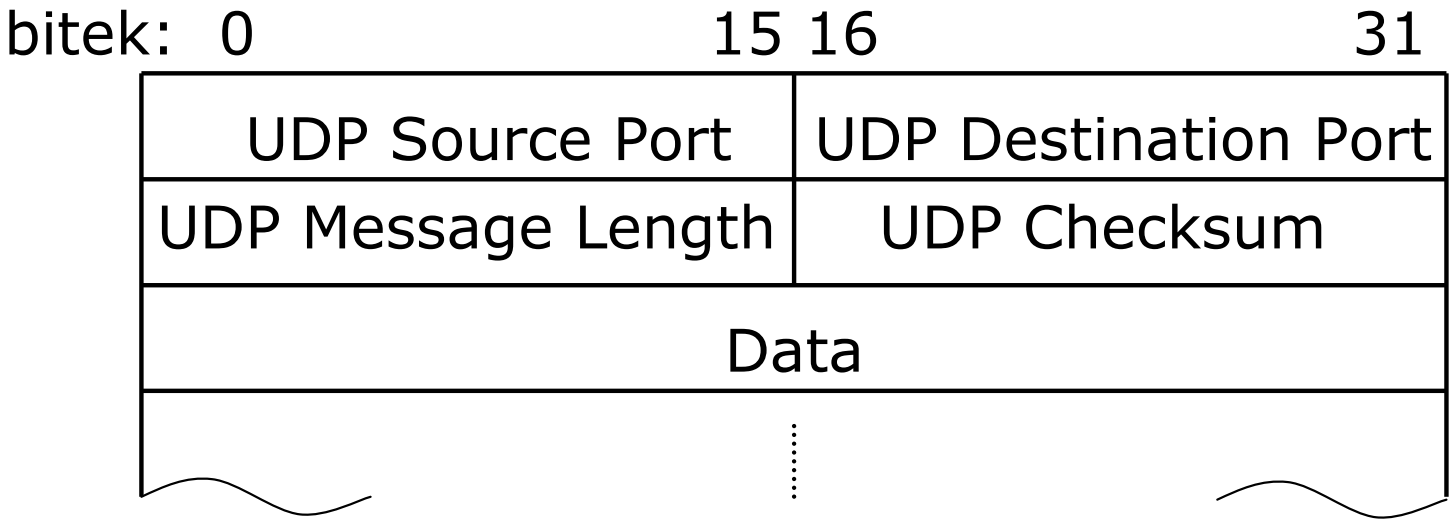
Multiplexelés-demultiplexelés, összeköttetésmentes eset (UDP)



Multiplexelés-demultiplexelés, összeköttetés-alapú eset (TCP)



UDP – User Datagram Protocol (3)

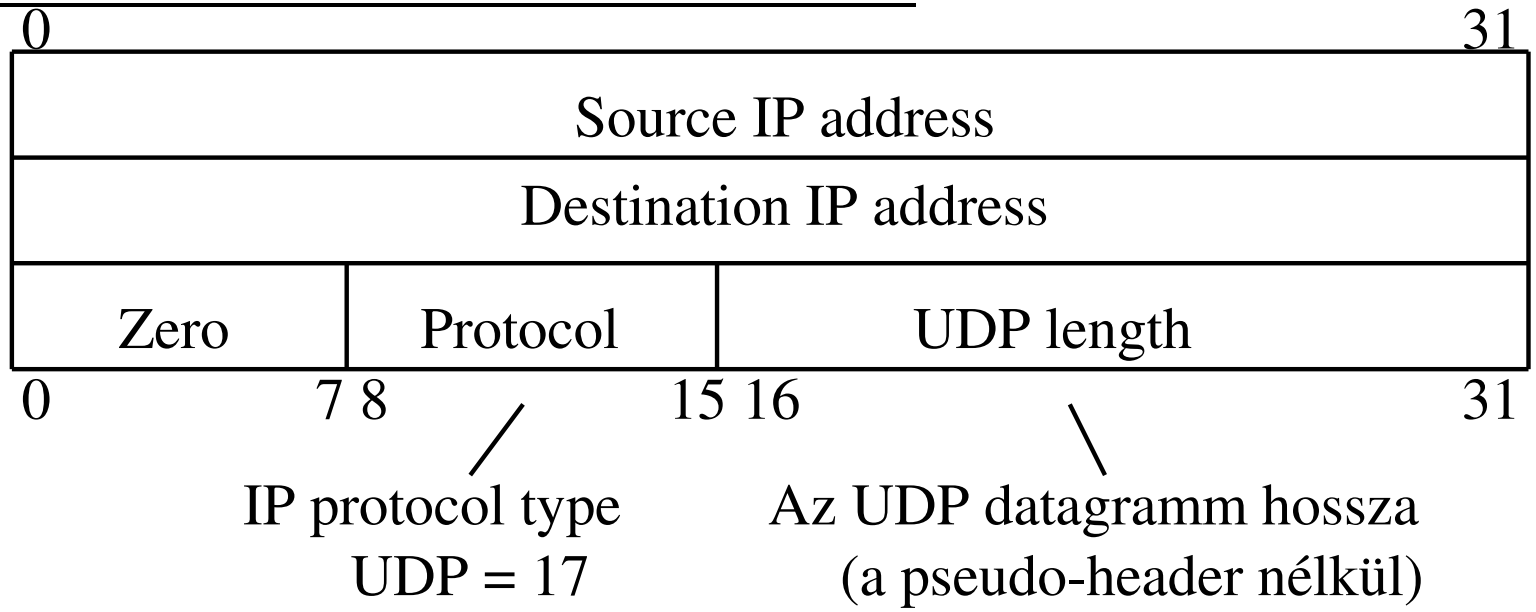


- Source port opcionális (nem használt: 0)
- Length: oktettben, min: 8
- Checksum: opcionális (nincs: 0)

- Az UDP checksum az egyetlen lehetőség annak ellenőrzésére, hogy a datagram helyesen érkezett-e meg (a IP-csomag ellenőrzése csak a fejrészt fogja át)

UDP – User Datagram Protocol (4)

- A checksum számítási elve (1-es komplemens 16 bites szavakra)
- Tartalmazza az IP-címeket is
  (annak ellenőrzésére, hogy a datagramm elérte a helyes címzettet, nemcsak a helyes portot)
- „Pseudo-header” hozzáadásával:



UDP – User Datagram Protocol
Összefoglalás: funkcionalitás és a költsége

- Portkezelés, ezáltal a különböző alkalmazások/folyamatok megkülönböztethetők
- Több alkalmazás egyidejű kezelése port-hozzárendeléssel és multiplexeléssel/demultiplexeléssel
- Hibajelzés az UDP datagram tartalmára és az IP csomag további részeire
- A fentiek költsége:
  minimum 8 oktettnyi overhead

TCP –Transmission Control Protocol
Fő jellemzői

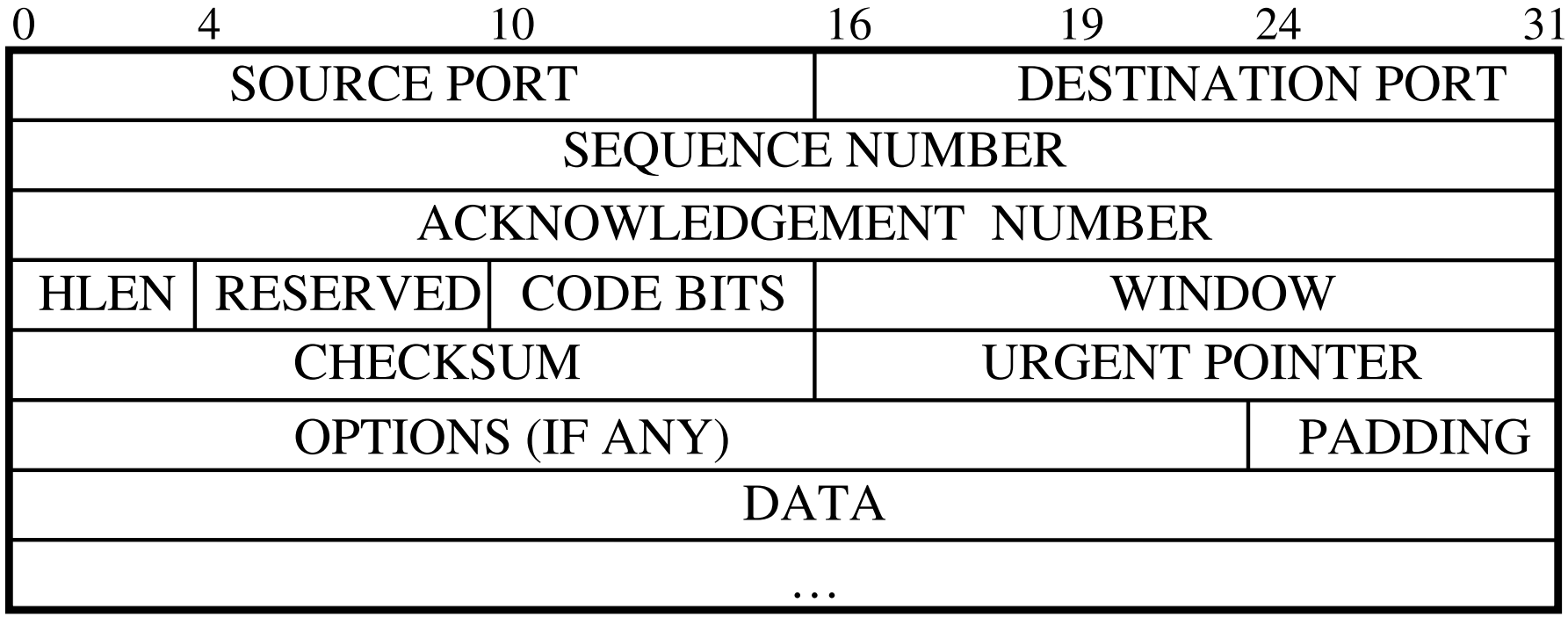
- **Célkitűzés:** megbízható szállítási szolgáltatás nyújtása az IP nem megbízható datagram-szolgáltatásán
- **Jellemzői:**
  - ***Virtuális összeköttetések:*** összeköttetés épül fel és marad fenn a kommunikáció tartamára
  - ***Stream–típusú szolgáltatás:*** bit- (oktett-) streamek sorrendhelyes átvitele
  - ***Strukturálatlan stream:*** nincsenek határolók a streamen belül
  - ***Pufferelt átvitel:*** a streamből a datagramm megtöltéséhez szükséges mennyiséget várja össze
  - ***Duplex kapcsolatok:*** két független stream
  - ***Vezérlő információk küldése:*** az ellenkező irányban folyó streambe ágyazva (piggybacking)

TCP – Transmission Control Protocol
Miről lesz szó?

- Adategység: szegmens; szegmensstruktúra
- Megbízható átvitel sorszámozás és pozitív nyugtázás segítségével
- Összeköttetés-alapú kommunikáció: kapcsolatfelépítés és -lebontás
- Forgalomszabályozás (flow control) ablakmechanizmus segítségével
- Torlódásvezérlés (congestion control)

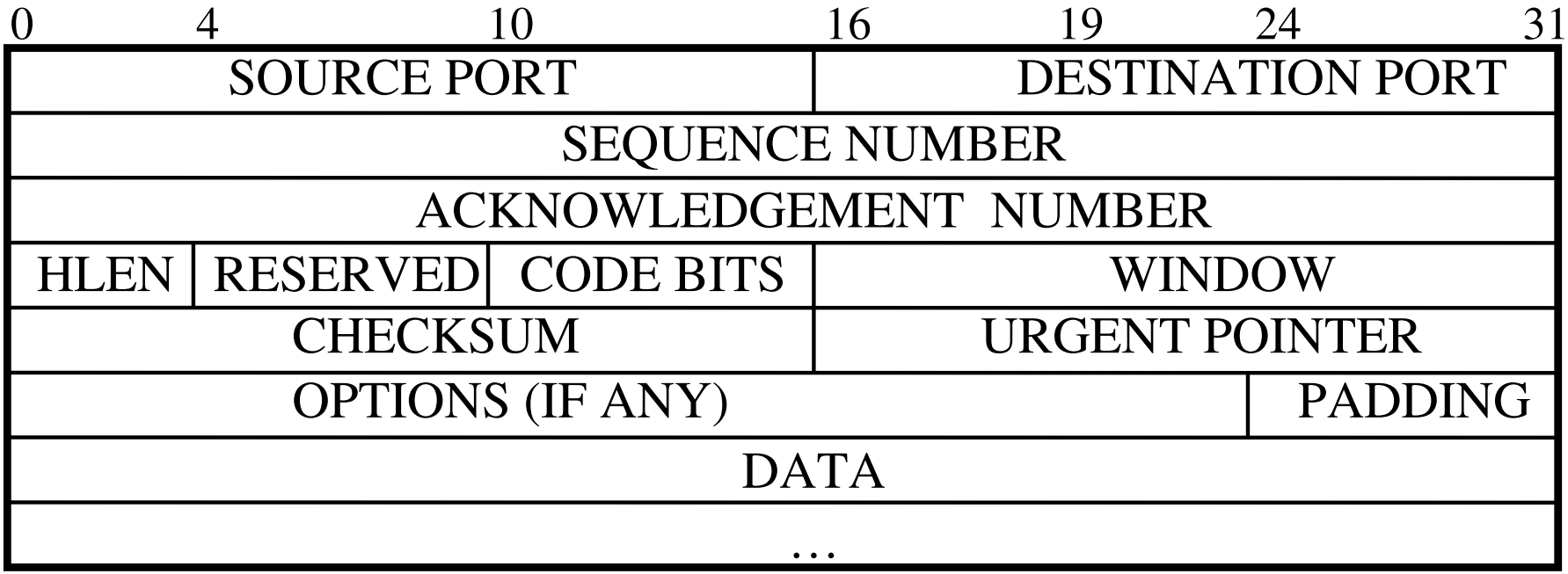
TCP – Transmission Control Protocol

A PDU – Protocol Data Unit (a TCP-ben: „segment”) struktúrája



- Sequence no.: a szegmensben levő adat első byte-jának pozíciója a küldő byte stream-jében
- Ack no.: annak a byte-nak a sorszáma, amelyet a forrás legközelebb vár

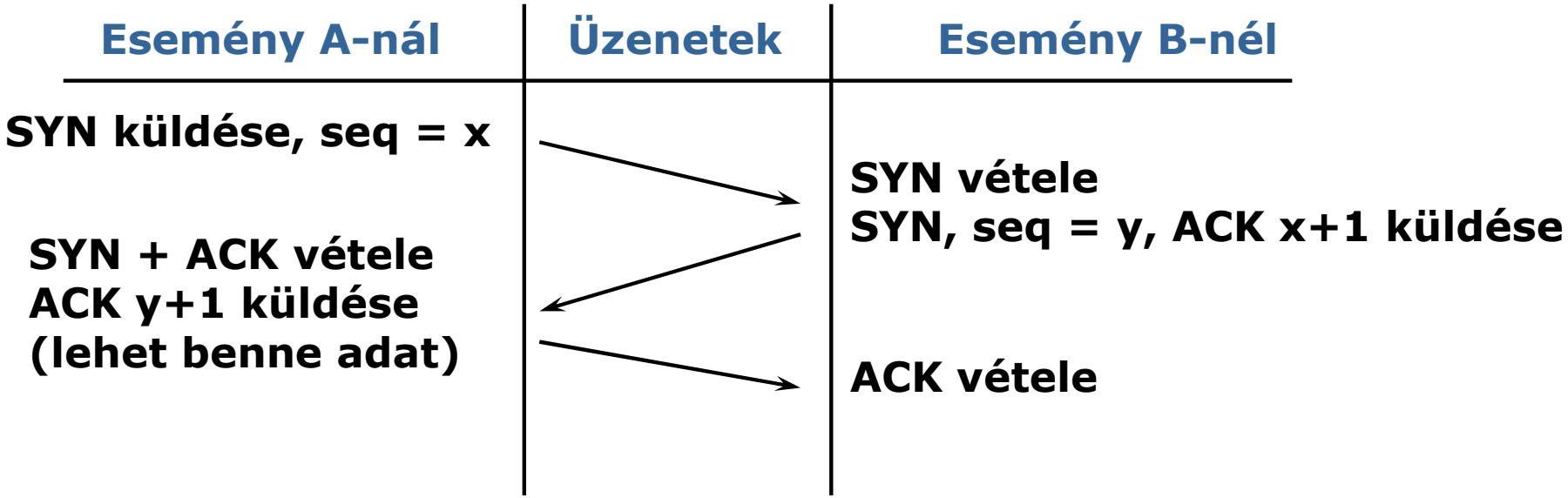
TCP – Transmission Control Protocol
Szegmensformátum (folyt.)



- Code bits (flags): URG, PSH, ACK, RST, SYN, FIN bitek a kapcsolat kezeléséhez használt jelzőbitek
- Window: a küldő ismertté teszi a vételi pufferének méretét
- Checksum: mint az UDP-ben
- Urgent pointer: ha URG, a szegmens „urgent” részt tartalmaz, ilyenkor a végére mutat

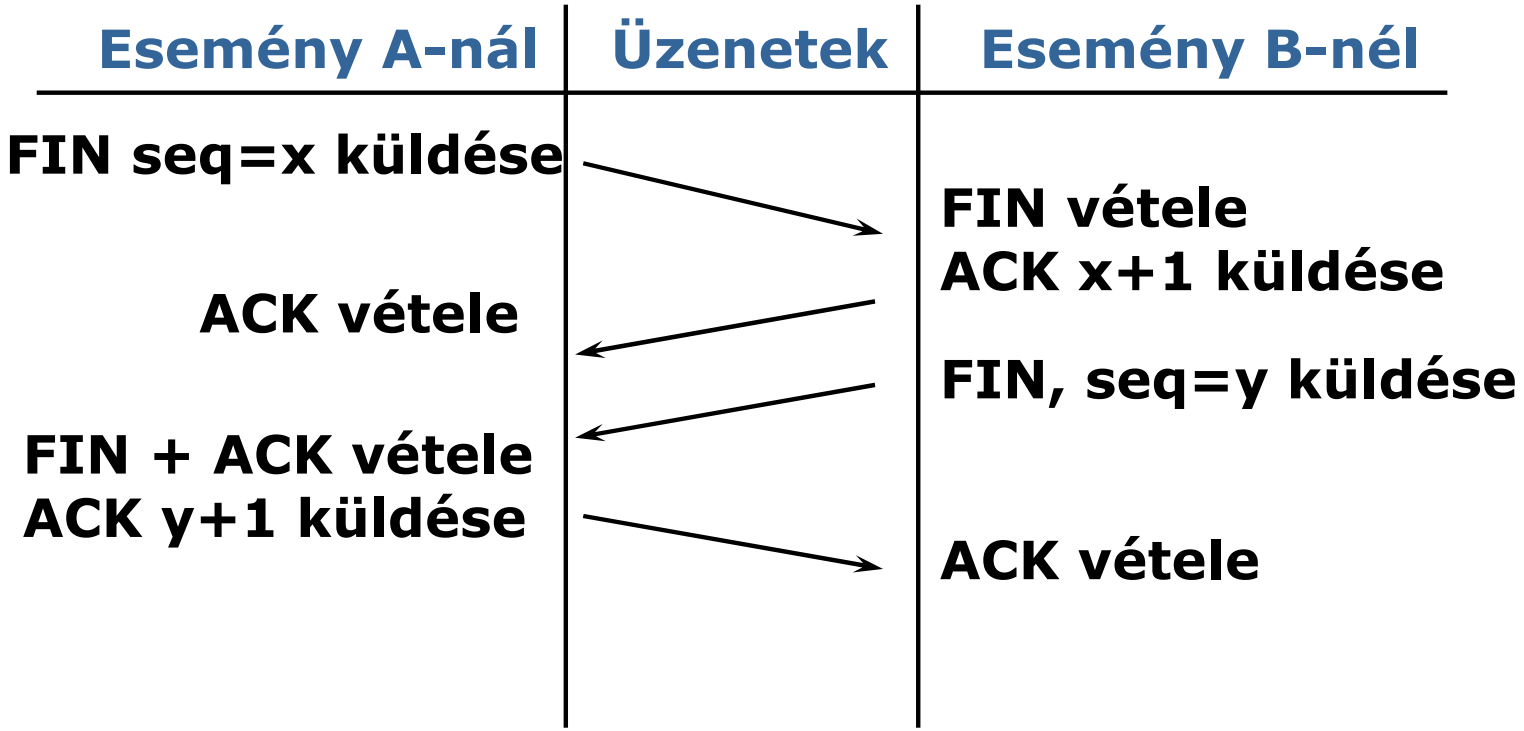
Hívásfelépítés a TCP-ben

„3-way handshake” eljárás

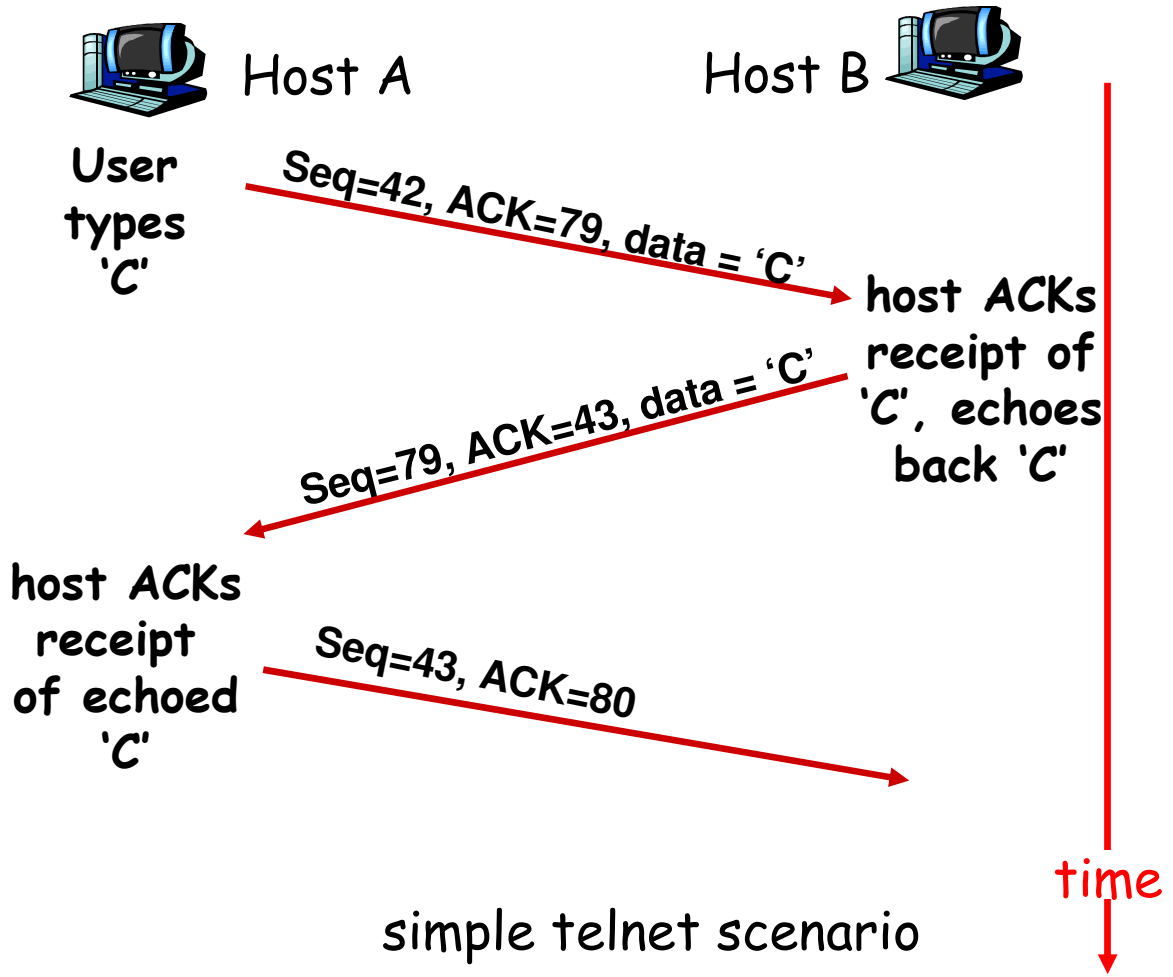


Híváslebontás a TCP-ben

„Modified 3-way handshake” eljárás



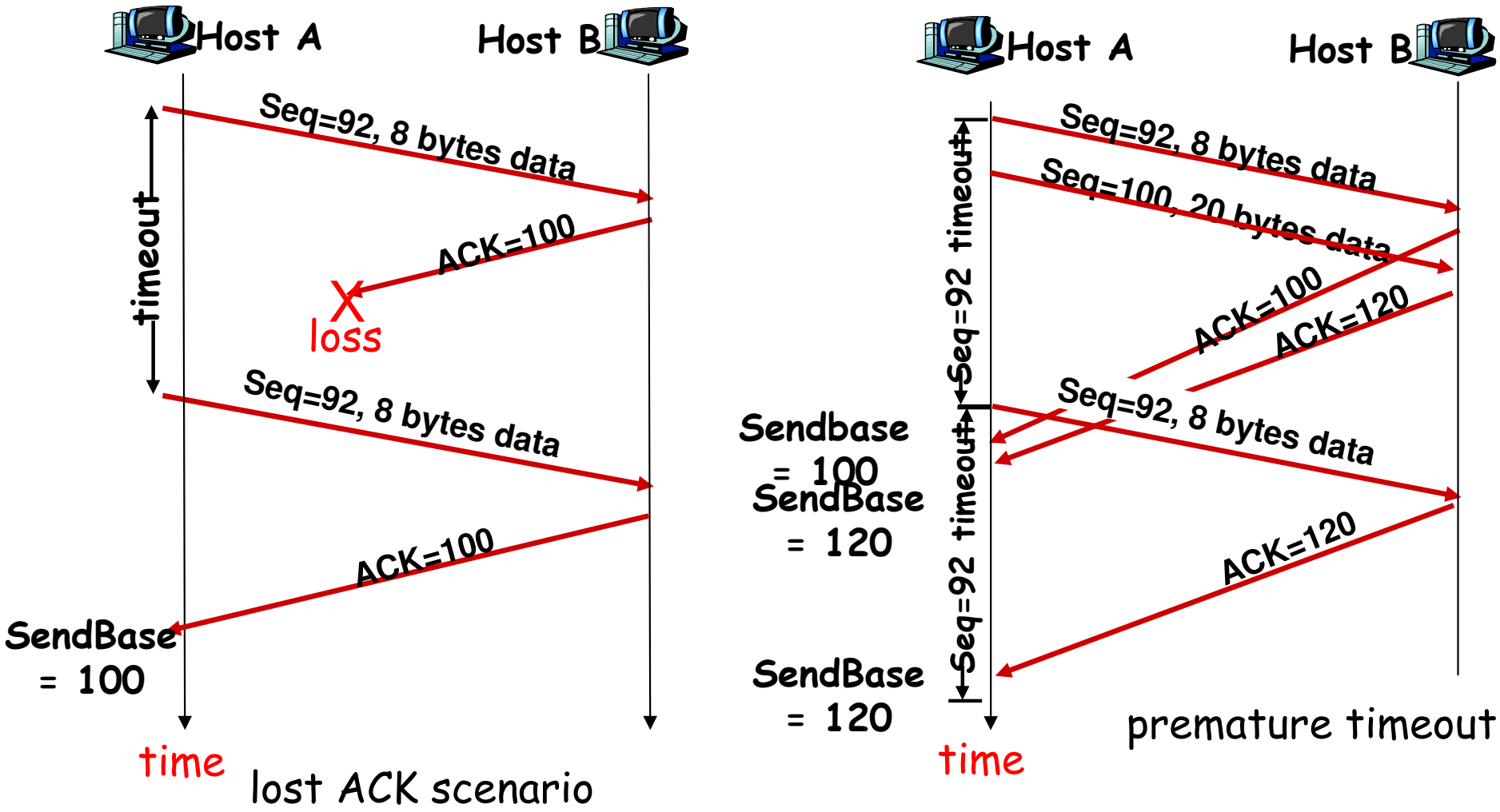
Sorszámok és nyugtaszámok használata a TCP-ben



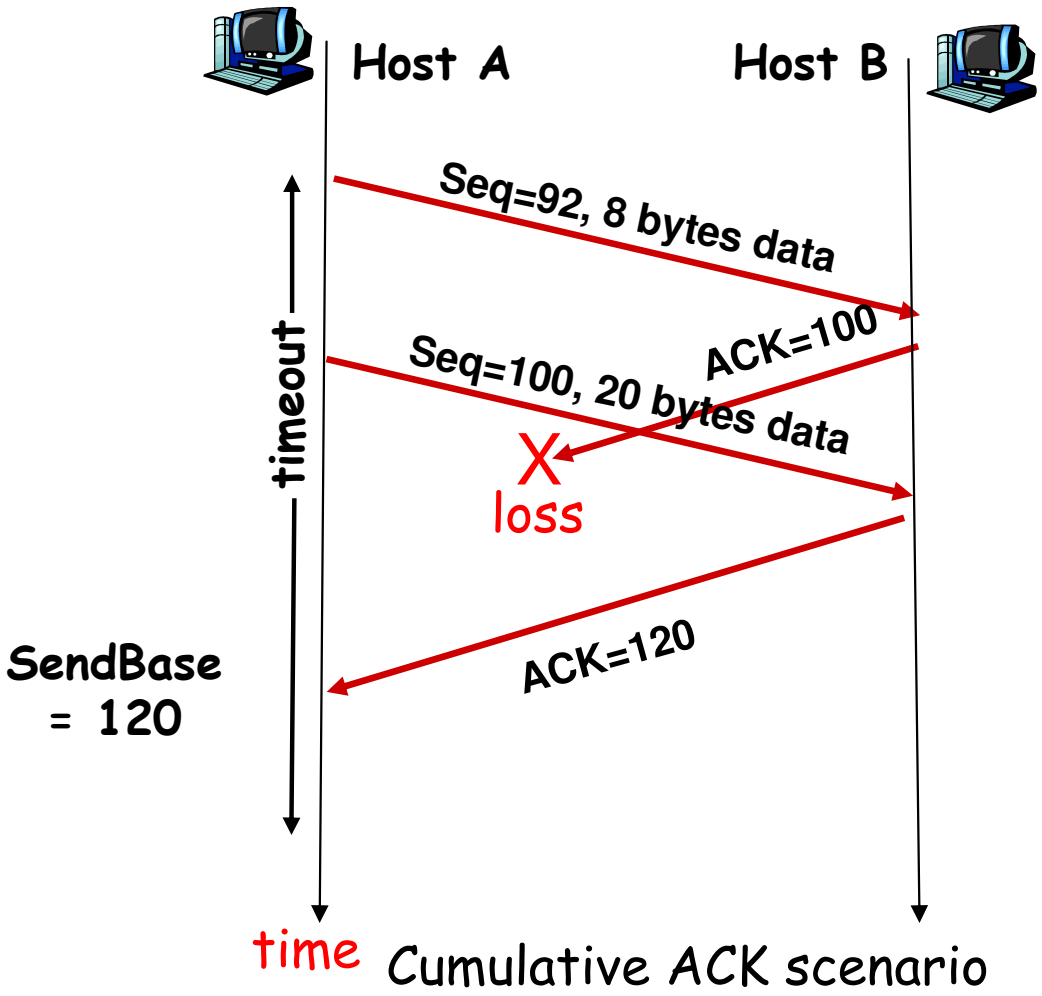
Várakozás nyugtázásra

- Várakozás ACK-ra: time–out
- Mekkorára válasszuk a time–out-ot?
- Probléma a túl kicsivel és a túl naggyal
- Megoldás:
  a teljes terjedési időhöz
  (RTT - round-trip time) igazítani, adaptívvá tenni
- Szabályok arra, hogy mit tegyünk, ha nem jön ACK a time-out alatt

Újraküldési esetek a TCP-nél: elveszett nyugta és korai timeout



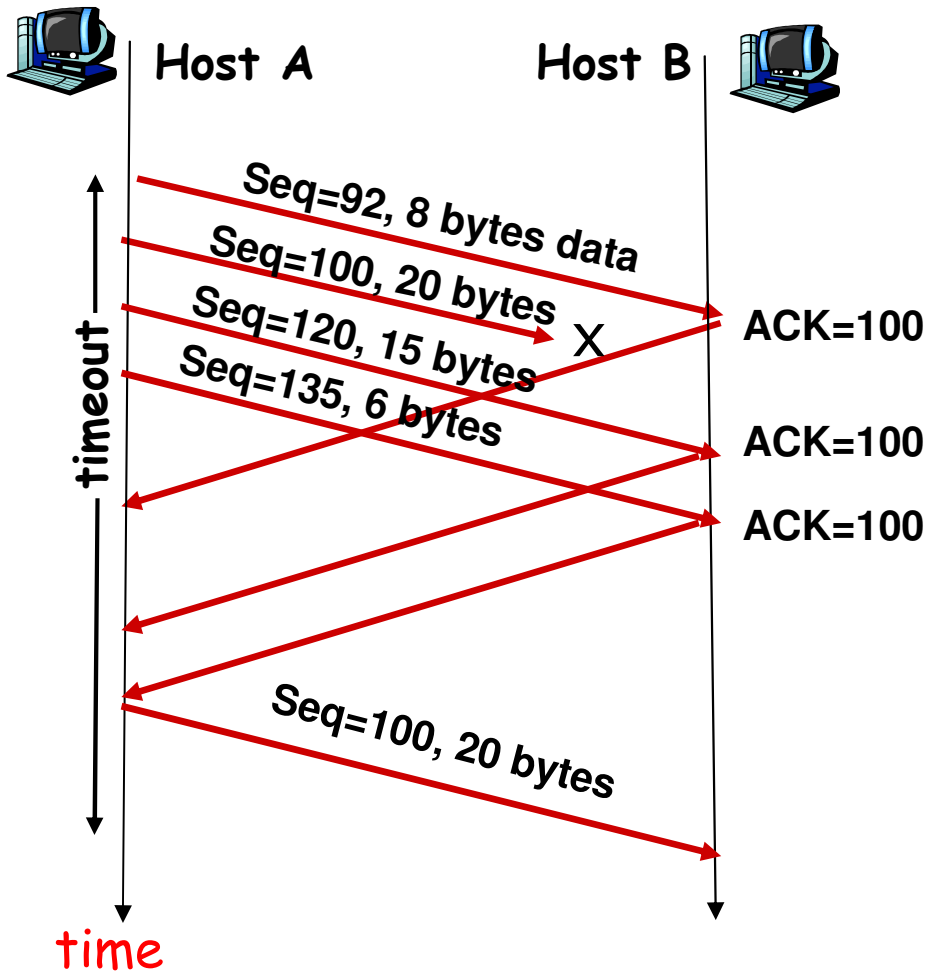
Újraküldési esetek a TCP-nél: összevont nyugta



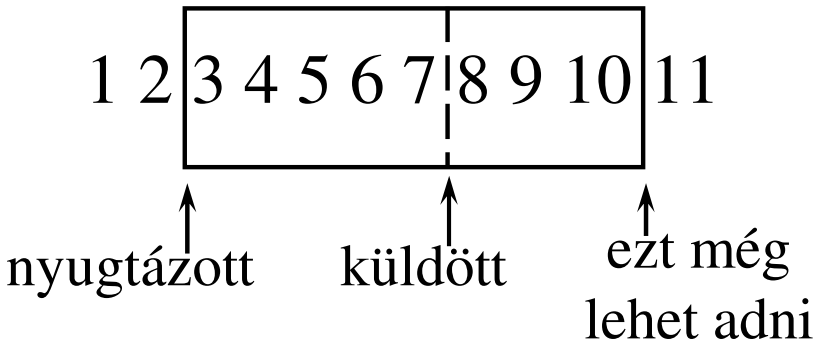
Fast retransmit (1)

- A time-out idő gyakran túl hosszú:
  - nagy késleltetés mielőtt az elveszett csomagot az adó újra tudná küldeni.
- De hogy értesülhetne az elveszett csomagról előbb, mintsem hogy letelt volna a timeout?
  - Az elveszett szegmensekre utalhatnak a duplikált ACK-ok.
  - Ha a vevő hézagot vesz észre a vett szegmensek sorozatában (elveszett csomag) akkor újból lenyugtázza a megelőző helyesen vett szegmenst.
  - Több egymást követő duplikált ACK érkezhet.
- Fast retransmit szabály: ha az adó 3 egymást követő ACK-t kap ugyanarra a szegmensre, feltételezi, hogy az azt követő szegmens elveszett és
  - újraküldi azt még mielőtt lejárna a timeout.

Fast retransmit (2)

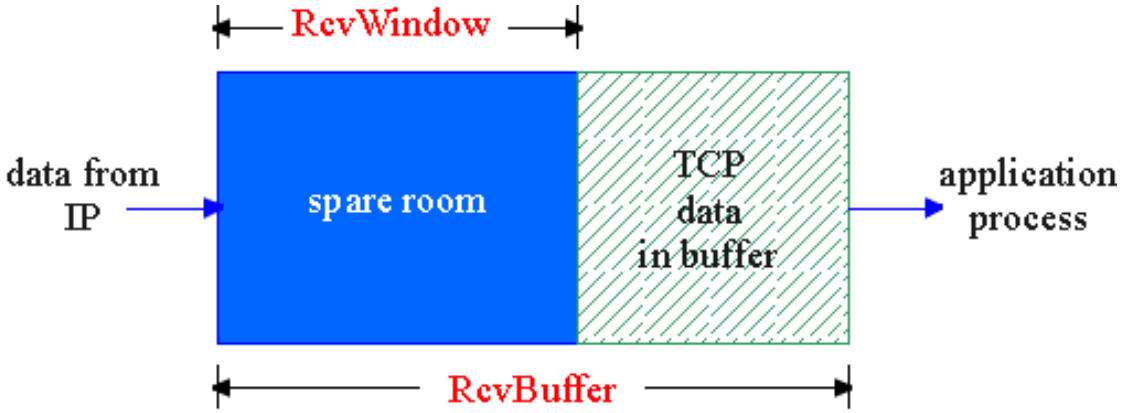


Flow control: a „sliding window” módszer elve



- Csúszóablakos („sliding window”) mechanizmus
  - az ablak mérete megadja a „kintlevő”, nyugtázatlan csomagok max. számát. (Pl.: w=8
- A TCP-ben: az ablak-mechanizmus oktetteken működik

TCP Flow control: hogy működik?



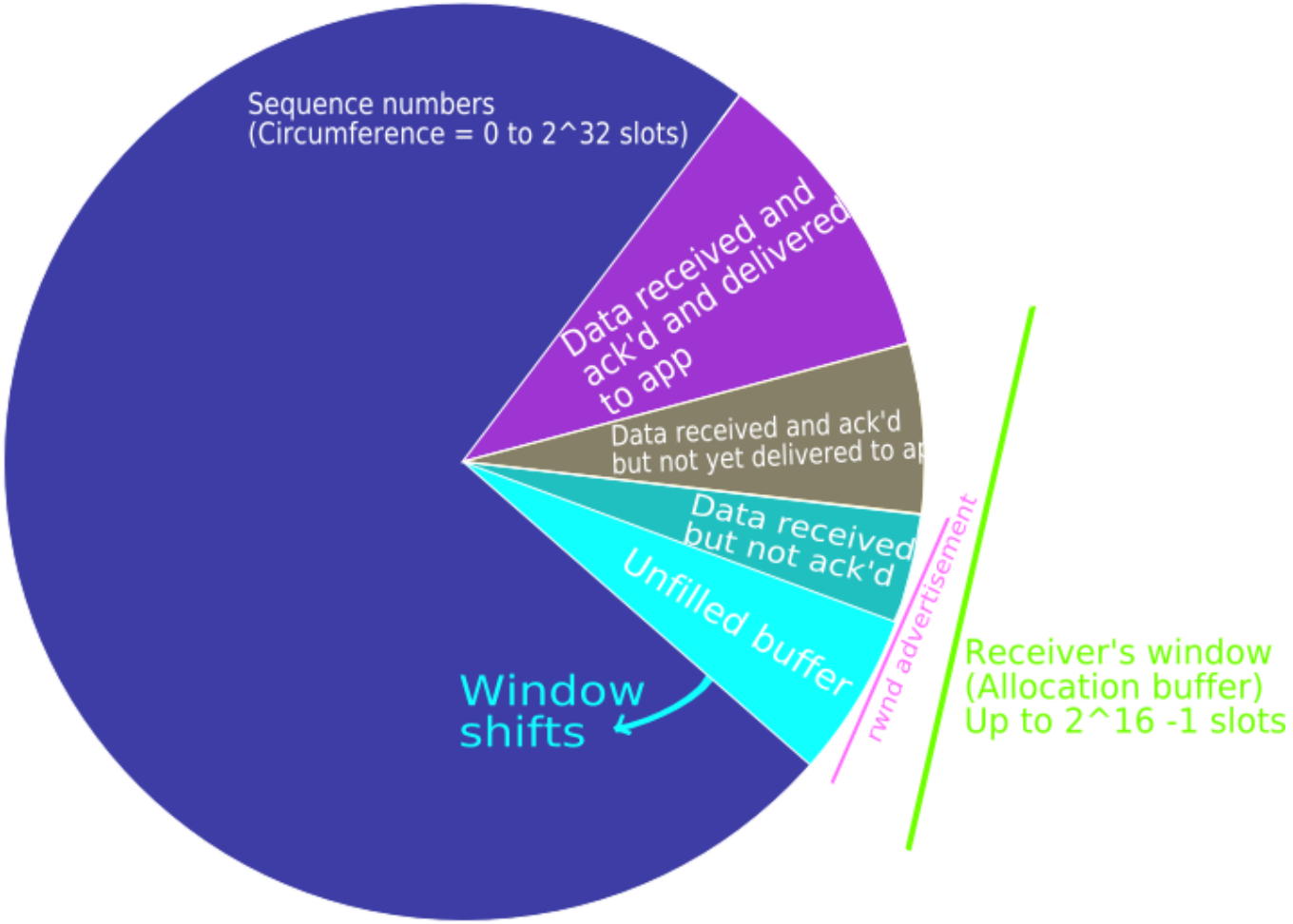
- spare room:

= RcvWindow

= RcvBuffer-[LastByteRcvd – LastByteRead]

- A vevő közli a szabad helyének a méretét (**RcvWindow**) a küldött szegmensben
- Az Adó legfeljebb **RcvWindow** menyiségű nyugtázatlan adatot küld

A TCP „óra”


Sequence numbers
(Circumference = 0 to 2^32 slots)
Data received and ack'd and delivered to app
Data received and ack'd but not yet delivered to ap
Data received but not ack'd
Unfilled buffer
Window shifts
rwnd advertisement
Receiver's window
(Allocation buffer)
Up to 2^16 -1 slots

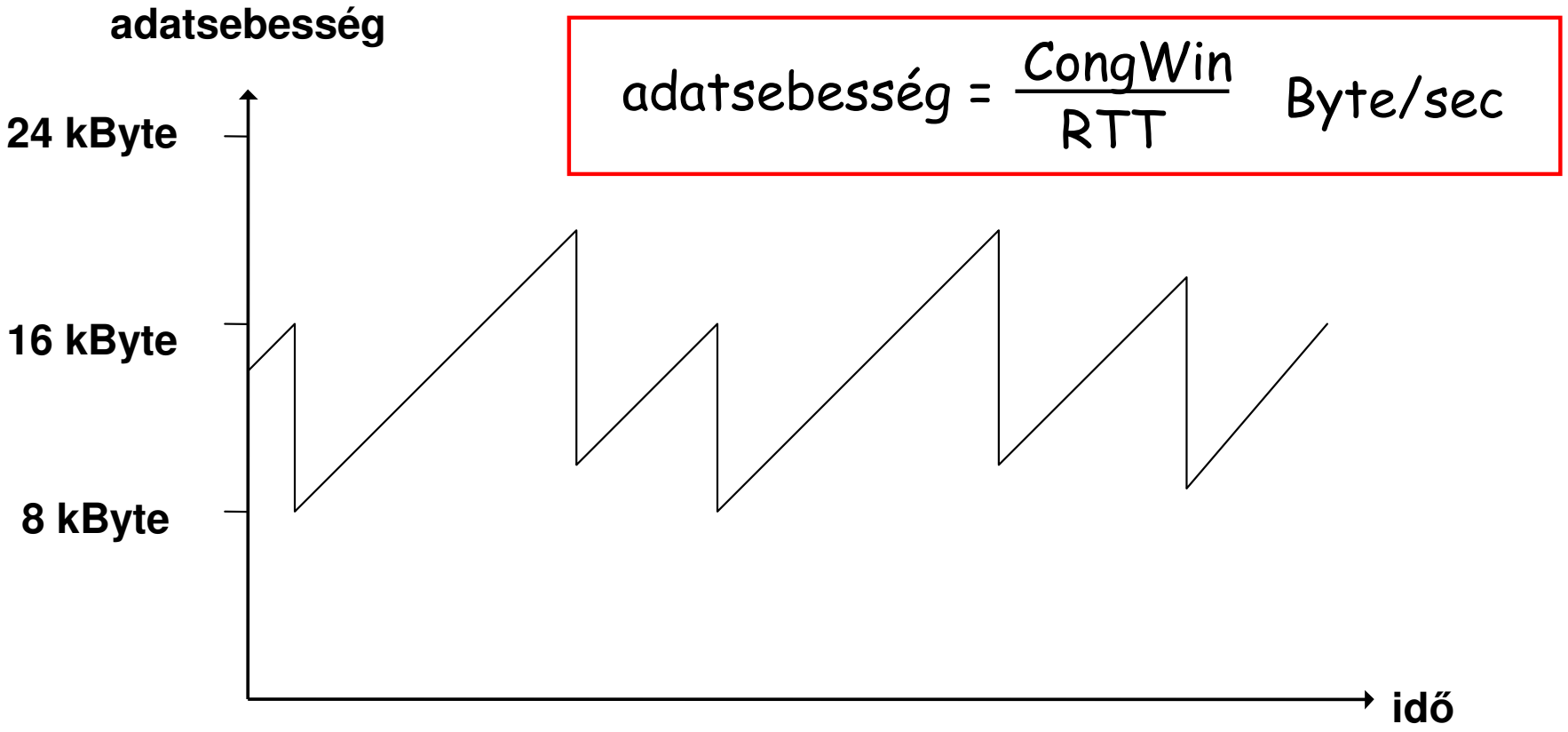
Torlódásvezérlés a TCP-ben

- A torlódásvezérlésről általában:
  - Azok a módszerek, amelyekkel linkek, csomópontok időszakos túlterheltségét megkíséreljük megszüntetni.
- Két fő módszer:
  - „Hálózat által segített” (network assisted) torlódásvezérlés
    - A hálózati elemek szolgáltatnak információt a túlterhelésről az adónak
  - Végpontok közötti (end-to-end) torlódásvezérlés
    - nincs visszacsatolás a hálózatból, a végpont következtet arra, hogy torlódás léphetett fel
- A TCP az utóbbit csinálja!

Torlódásvezérlés a TCP-ben

- Módszer: növeljük az adatsebességet, ha úgy érzékeljük, van elég átbocsátóképesség, amíg torlódásra utaló jeleket nem tapasztalunk, ha igen, csökkentsük.
- Hogyan:
  - congestion window, CongWin
  - növeljük a CongWin-t minden RTT alatt MSS-sel, amíg vesztést nem érzékelünk
  - csökkentsük a CongWin-t felére minden vesztéskor
  - = Additive increase – multiplicative decrease (AIMD) módszer
- Hogyan érzékeljük a torlódást (vesztést)?
  - timeout letelt, nem jött nyugta
  - többszörös nyugta érkezett ugyanarra a szegmensre (miért?)

Az adatsebesség alakulása AIMD esetén



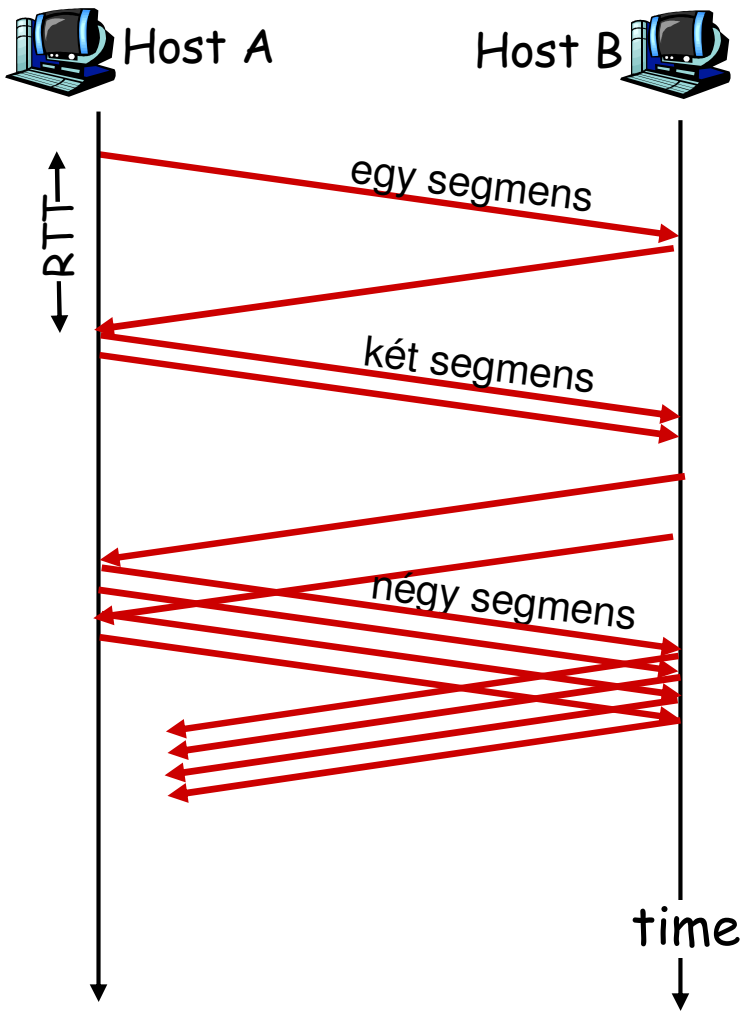
$$\text{adatsebesség} = \frac{\text{CongWin}}{\text{RTT}} \quad \text{Byte/sec}$$

TCP torlódásvezérlés: további részletek

- Az AIMD kiegészítései:
- „Slow Start”
  - az összeköttetés kezdetén a sebesség gyors (exponenciális) növelése az első vesztésig, utána AIMD
- Eltérő viselkedés timeout és többszörös (3-szoros) nyugták esetén

TCP Slow Start

- Az elején: **`CongWin`** = 1 MSS
  - Példa: MSS = 500 byte & RTT = 200 msec
  - kezdeti sebesség = 20 kbps
- Mivel a rendelkezésre álló átb. képesség >> MSS/RTT lehet,
  - célszerű gyorsan elérni, ezért
  - Induláskor exponenciálisan növelni az első vesztésig



Refinement: inferring loss *

- After 3 dup ACKs:
  - **CongWin** is cut in half
  - window then grows linearly
- But after timeout event:
  - **CongWin** instead set to 1 MSS;
  - window then grows exponentially
  - to a threshold, then grows linearly

Philosophy:

- 3 dup *ACKs* indicates network capable of delivering some segments
- timeout indicates a "more alarming" congestion scenario

Summary: TCP Congestion Control *

- When **`CongWin`** is below **`Threshold`**, sender in slow-start phase, window grows exponentially.
- When **`CongWin`** is above **`Threshold`**, sender is in congestion-avoidance phase, window grows linearly.
- When a triple duplicate ACK occurs, **`Threshold`** set to **`CongWin/2`** and **`CongWin`** set to **`Threshold`**.
- When timeout occurs, **`Threshold`** set to **`CongWin/2`** and **`CongWin`** is set to 1 MSS.

TCP sender congestion control *

State	Event	TCP Sender Action	Commentary
Slow Start (SS)	ACK receipt for previously unacked data	CongWin = CongWin + MSS, If (CongWin > Threshold) set state to "Congestion Avoidance"	Resulting in a doubling of CongWin every RTT
Congestion Avoidance (CA)	ACK receipt for previously unacked data	CongWin = CongWin+MSS * (MSS/CongWin)	Additive increase, resulting in increase of CongWin by 1 MSS every RTT
SS or CA	Loss event detected by triple duplicate ACK	Threshold = CongWin/2, CongWin = Threshold, Set state to "Congestion Avoidance"	Fast recovery, implementing multiplicative decrease. CongWin will not drop below 1 MSS.
SS or CA	Timeout	Threshold = CongWin/2, CongWin = 1 MSS, Set state to "Slow Start"	Enter slow start
SS or CA	Duplicate ACK	Increment duplicate ACK count for segment being acked	CongWin and Threshold not changed

TCP throughput *

- What's the average throughout of TCP as a function of window size and RTT?
  - Ignore slow start
- Let W be the window size when loss occurs.
- When window is W, throughput is W/RTT
- Just after loss, window drops to W/2, throughput to W/2RTT.
- Average throughout: .75 W/RTT

TCP Futures:
TCP over “long, fat pipes” *

- Example: 1500 byte segments, 100ms RTT, want 10 Gbps throughput
- Requires window size W = 83,333 in-flight segments
- Throughput in terms of loss rate:

$$\frac{1.22 \cdot MSS}{RTT\sqrt{L}}$$

- ➜ L = $2 \cdot 10^{-10}$ *Wow*
- New versions of TCP for high-speed needed!

TCP és UDP: összefoglalás

- Mindkettő host layer/transport layer protokoll
- Mindkettő portokat kezel
  - multiplexelés/demultiplexelés
  - ezáltal interface nyújtása az alkalmazói folyamatok felé
- Az UDP összeköttetés-mentes, best effort szolgáltatás
  - nem garantál célba juttatást, csak hibajelzést nyújt
  - gyorsan célba juttat
- A TCP összeköttetés-alapú, megbízható transzport szolgáltatás
  - sorrendhelyes, hibamentes szállítást nyújt
  - ára: késleltetés

Néhány gyakori alkalmazás és az használt transzportprotokollok

Alkalmazás	***Alkalmazási rétegbeli protokoll***	***Használt transzport-protokoll***
E-mail	SMTP	TCP
Távoli elérés	Telnet	TCP
Web-elérés	HTTP	TCP
File-átvitel	FTP	TCP
Routing	RIP	UDP
Hálózatmenedzsment	SNMP	UDP, TCP
VoIP, média-streaming	Többnyire nem szabványos	UDP