

Számítógép-hálózatok

A felsőbb rétegek

2025/2026. tanév, I. félév

Dr. Kovács Szilveszter

E-mail: szilveszter.kovacs@uni-miskolc.hu

www.iit.uni-miskolc.hu/~szkovacs

Informatika Intézet 107/a.

Tel: (46) 565-111 / 21-07

A szállítási réteg



- **TSAP: Transport Service Access Point**
- **TPDU: Transport Protocol Data Unit**
- **A 4. réteg, a hálózati- és a viszonyréteg között**

A szállítási réteg

- **Feladata**

- Interfész alulra, felülre
- Megbízható, gazdaságos adatszállítást forrástól célhosztig, függetlenül a hálózatoktól (a céltól, forrástól, a közbenső alhálózatoktól), ÖK vagy ÖK mentes alapon
- Tudjuk, hogy valódi end-to-end szolgáltató entitások vannak

- **Miért kell?**

- Az interfész - ha nem lenne, nem lehetne hozzáférni
- Megbízhatóság - ezt az adatkapcsolati és a fizikai réteg is biztosíthatná.
- ÖK alapú és ÖK mentes szolgálat - ezt is biztosíthatják az alsóbb rétegek (ált. csak az egyiket).
- Vég-vég - ezt a hálózati réteg is biztosíthatná (lásd IPX datagram kapcsolat – nincs is)

A szállítási réteg

- **Miért kell?**
 - A szállítási réteg a hálózati rétegre épül
 - A hálózati réteg lehet ÖK alapú, vagy ÖK mentes,
 - Nem szükségszerűen megbízható!
(Az IP ÖK mentes és megbízhatatlan)
 - Még megbízható hálózati réteg mellett is lehetnek hibák .
(A teljes hibamentesség a hálózati rétegben nem megoldható, nem az a „dolga”.)

A szállítási réteg

- Legyen a hálózati réteg fölött a szállítási, ami valóban megbízható end-to-end szolgáltatokat biztosít,
- az alkalmazások így
- szabványos interfészeken keresztül különböző hálózatokon (megbízható és megbízhatatlan is) is jól működhetnek.
- Ezért a hálózat megbízhatósága szempontjából a szállítási réteg lényeges funkciókat lát el.

A szállítási réteg

- Fontos cél a szállítási rétegben
 - hibamentes átvitel akár hibákkal terhelt hálózati réteg fölött is!
 - Ebből következik: a a fölöttes rétegeknek tényleg nem kell emiatt nyugtázással stb. foglalkozni!
 - (PL. ha egy hálózati összeköttetés megszakad, akkor a szállítási réteg nyit egy újat és ott folytatja, ahol a régivel abbahagyta. A fölöttes réteg észre sem veszi ezt.)
- Fontos célja még
 - elrejtteni a konkrét hálózatot (annak minden problematikáját, sajátságát) a felettes rétegek előtt.

Üzenet szegmentálás-összerakás; nyalábolás-szétbontás

- **Darabolás - összerakás**
 - van, hogy egy üzenet (ami a felsőbb rétegtől jön) túl nagy a hálózati (esetleg az adatkapcsolati) rétegnek
 - A szállítási réteg ilyenkor darabol - összerak.
- **Multiplexálás - demultiplexálás**
 - Előfordul, hogy sok kis üzenet van ugyanahhoz a célhoz.
 - A szállítási réteg nyalábolhatja ezeket egy csomagba (illetve demultiplexálja ezt a másik oldalon).
Teljesítménynövelés.

Kapcsolati szolgálatok

- **Csomagszámozás (szegmensszámozás)**
 - A helyes sorrend visszaállítás szolgálathoz kellhet (ÖK mentes kapcsolatnál feltétlenül)
- **Hibavezérlés**
 - Lehetnek hibás, elveszett, vagy késő csomagok,
 - a várt csomagszám térből kilógó csomagok.
 - **Megoldások:**
 - Ellenőrző összeg a csomagokban,
 - időzítések, hogy a késő csomagokat eldobjuk,
 - a csomagszámozás egyedi legyen.
- **Az end-to-end kapcsolathoz kell flow-control**
 - A kérdés itt: vajon mind a forrás, mind a cél foglalkozzon az elveszett - késő csomag problémából való kilábalással

A szállítási réteg szolgálat primitívjei

- Lehet ÖK mentes és ÖK alapú szolgálat. Utóbbi lehet megerősítéses.

- A primitívek

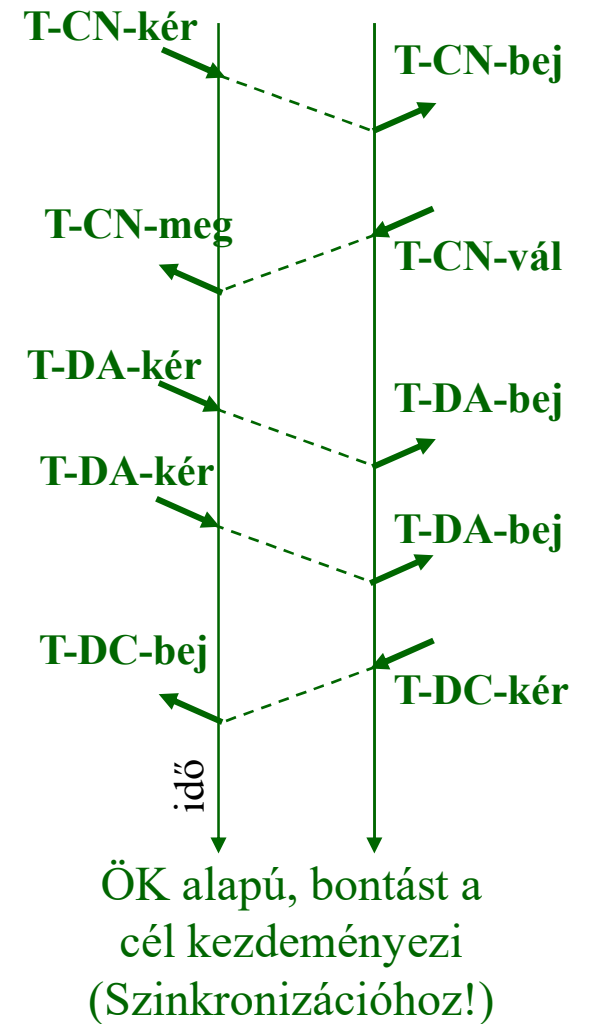
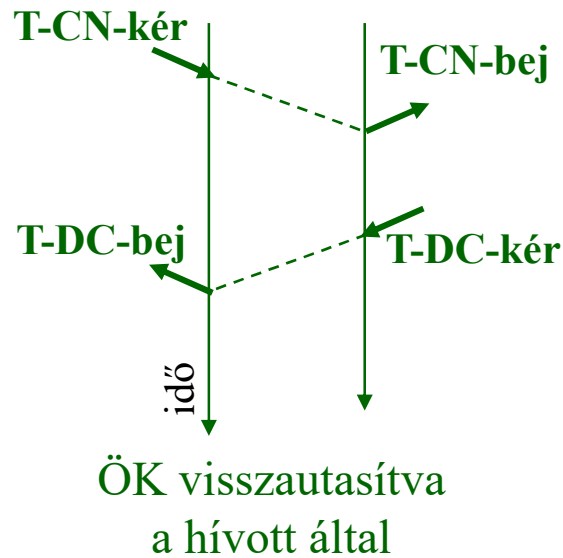
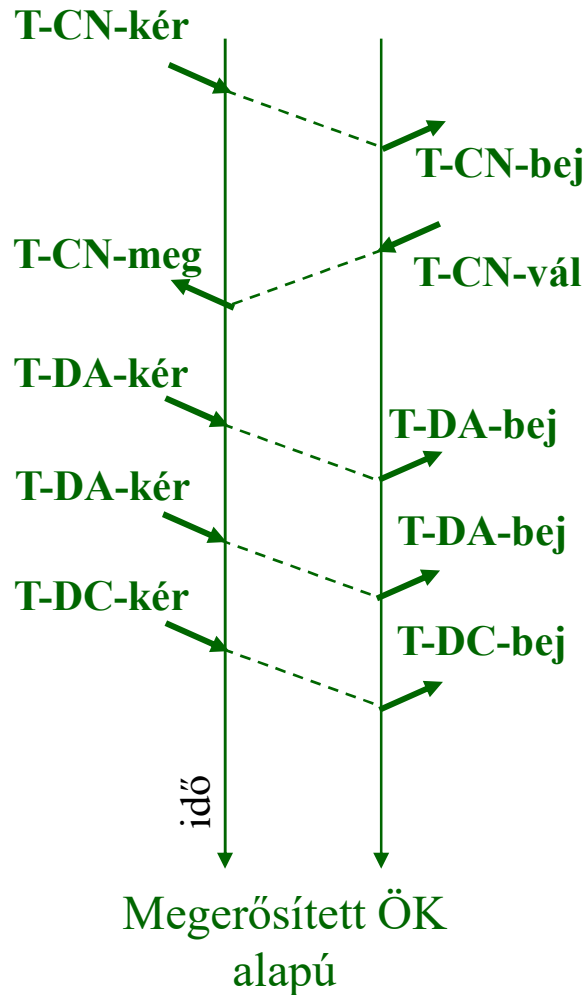
ÖK alapúhoz

- T-CN-kérés (connect)
- T-CN-bejelentés
- T-CN-válasz (megerősítéses szolgálathoz)
- T-CN-megerősítés (megerősítéses szolgálathoz)
- T-DC-kérés (disconnect)
- T-DC-bejelentés
- T-DA-kérés (data)
- T-DA-bejelentés

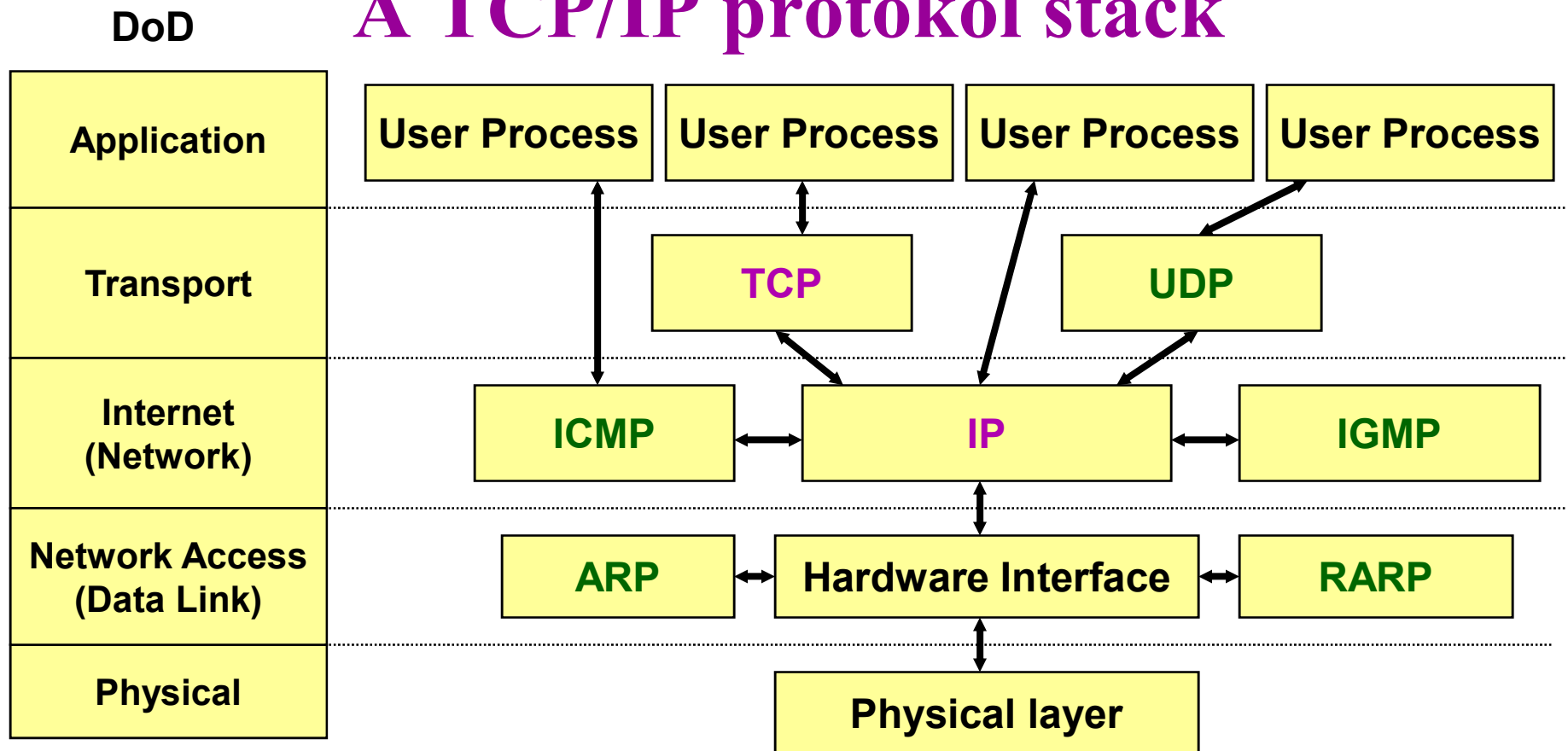
ÖK menteshez

Forgatókönyv példák

(A viszonyréteg szemszögéből)



A TCP/IP protokol stack



Szállítási réteg:

TCP: Transmission Control Protocol (Telnet, Rlogin, FTP, SMTP, DNS)

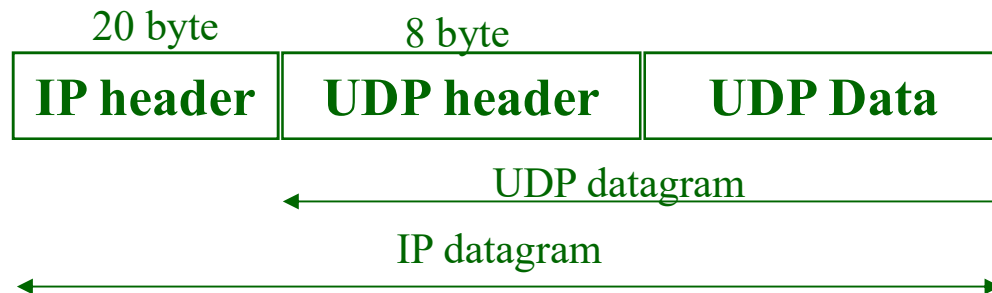
→ megbízható adattovábbítás (összeköttetés alapú szolgálat)

UDP: User Datagram Protocol (TFTP, SNMP, DNS)

→ összeköttetés-mentes datagramm szolgálat

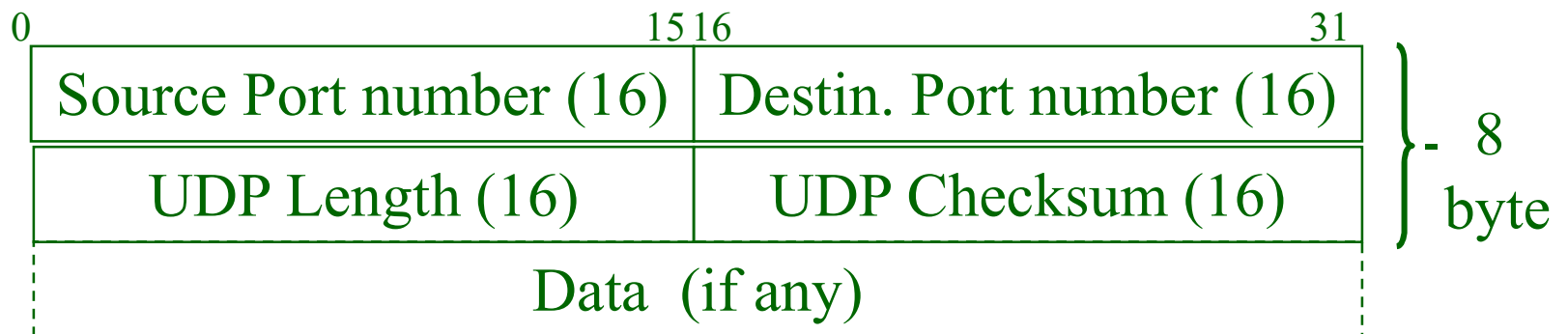
UDP: User Datagram Protocol

- Egyszerű,
- ÖK mentes (datagram),
- nem megbízható szolgálat.
- Minden továbbítandó üzenet 1 UDP datagram (amit egy IP datagram-ként, csomagként továbbítanak)
 - Az IP enkapszuláció:

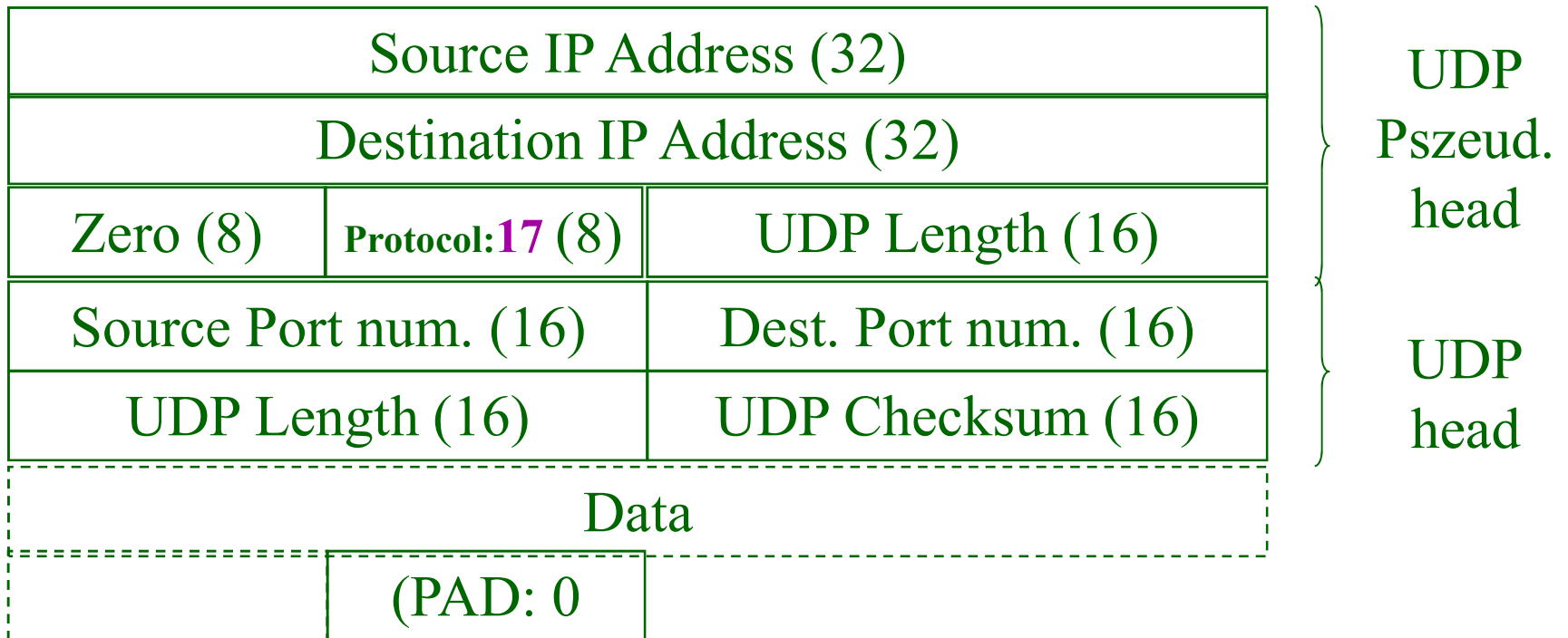


UDP Header

- 16 bit a forrás és cél szolgálat elérési port szám
- **16 bit UDP length**: a teljes UDP csomag hossza byte-ban
 - 16 bit → min: 8 (UDP header)
max: $2^{16} - 8\text{Byte (UDP header)} - 20\text{Byte (IPv4 header)}$ (65,535–8–20)
 - **IPv6** esetén lehet hosszabb is (jumbo payload) , ilyenkor UDP Length = 0
- **16 bit UDP checksum** az UDP header+UDP data-n (biztonság növelésére az IP header egy részére is kiterjed, hasonlóan a TCP checksum-hoz),
a feladó generálja (opcionális), a vevő ellenőrzi:
1 komplement 16 bit összeg
(ha a vett CS=0 → az adó nem használja)

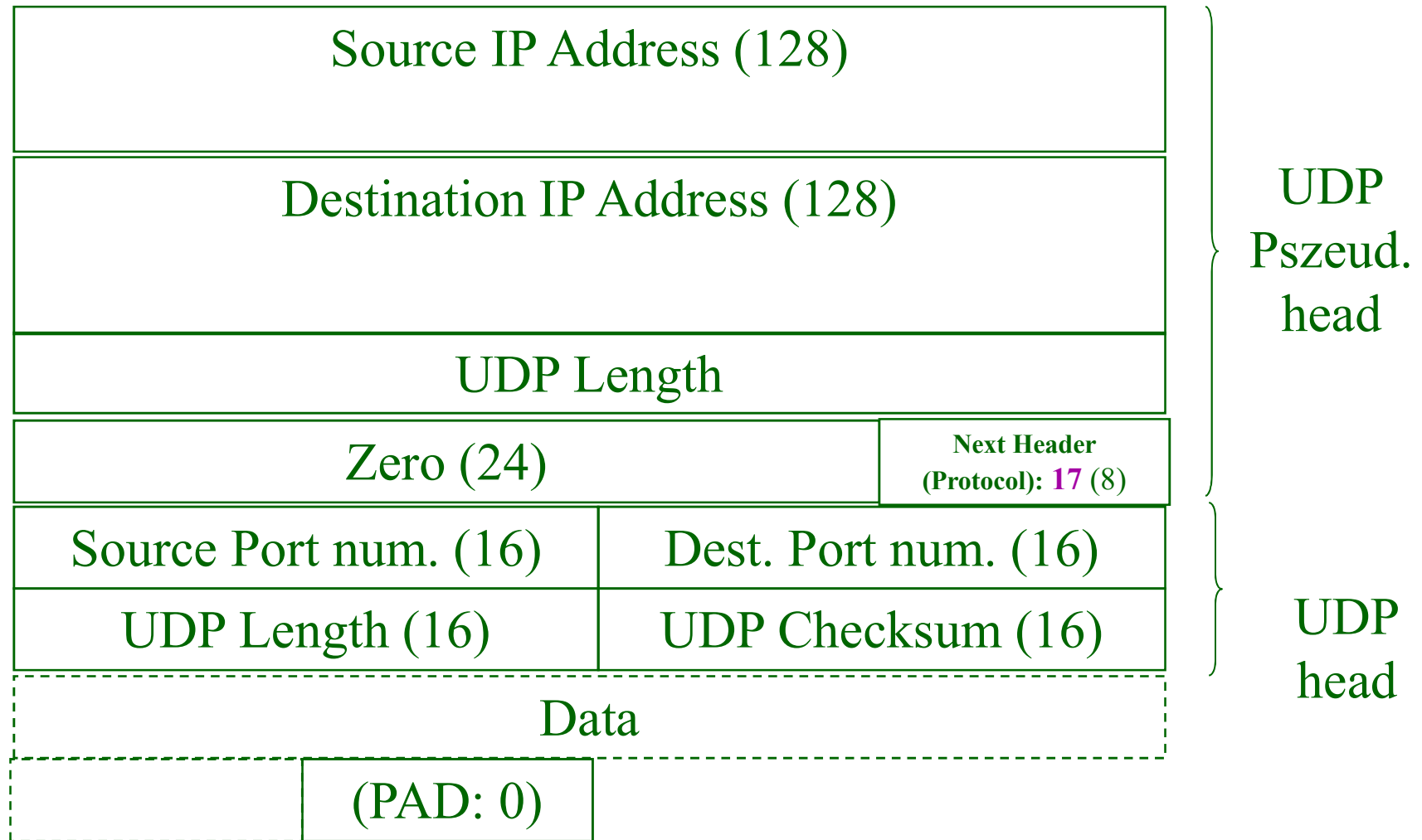


UDP pseudo fej a checksum számításához IPv4



- Ha a Checksum = 0 lenne → **65535** (-0, 1-komplemenst) továbbít
- A Checksum = 0 a checksum hiányát jelzi (az adó nem használja)

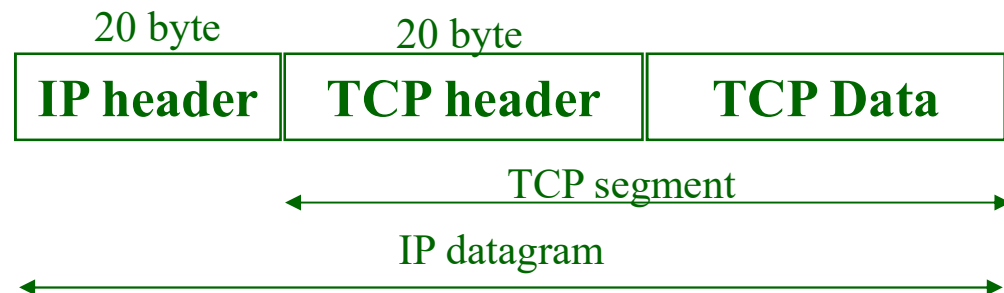
UDP pseudo fej a checksum számításához IPv6



- IPv6 esetén kötelező a számítása

TCP Transmission Control Protocol

- Bonyolultabb,
- ÖK alapú (sorrendhelyes),
- megbízható (hibamentes),
- duplex (kétirányú) szolgáltatot biztosít.
- Meghatározza az IP felé az optimális csomagméretet
- „TCP szegmens”: az IP felé továbbított adataegység
- „Byte stream service”: ha a kapcsolat felépült, a forrás byte-okat küld, a cél byte-okat fogad folyamatosan (virtuális áramkör byte-okra)
- IP enkapszuláció:



TCP Transmission Control Protocol

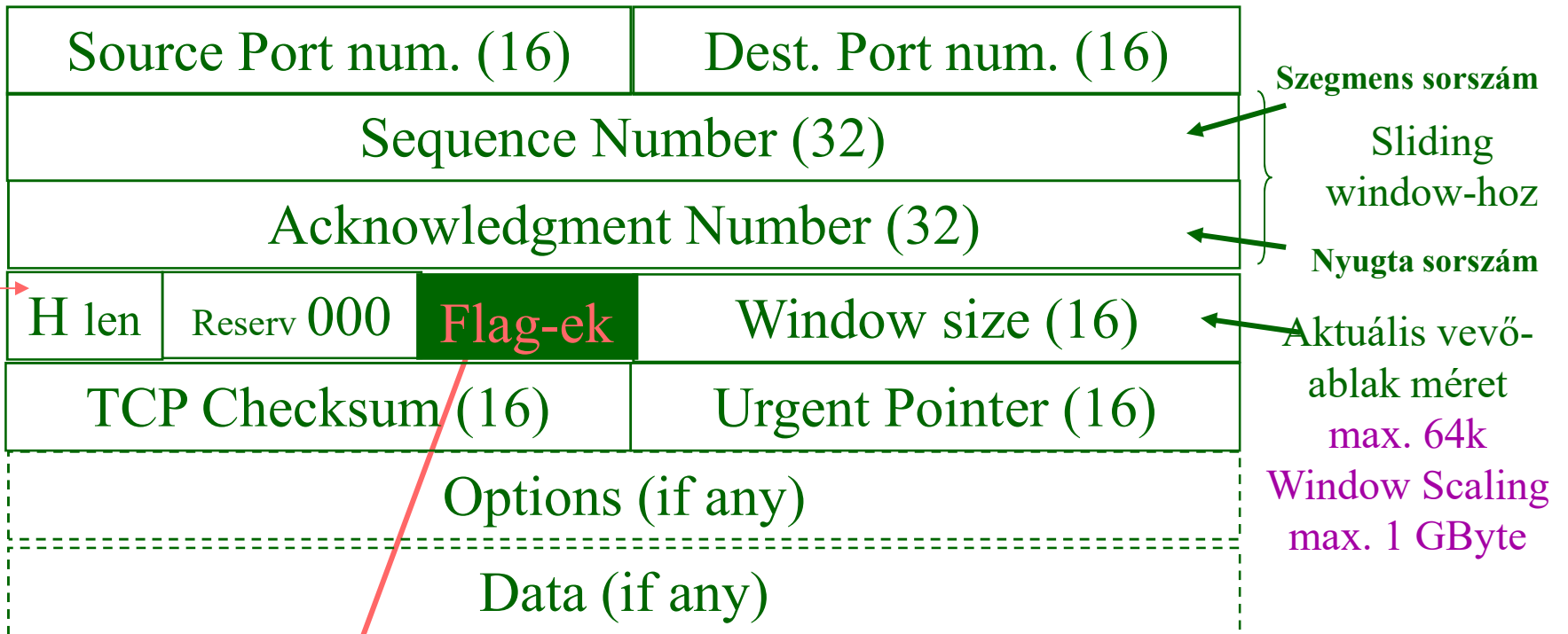
- **TCP = Transmission Control Protocol**
- **Connection-oriented OP Transport**
- **RFCs**
 - RFC 793 defines TCP
 - RFC 1122 – bug fixes and clarification
 - RFC 1323 – extensions
- **TCP segment**
 - One IP datagram
- **MTU = Maximum transfer unit**

TCP Service Model

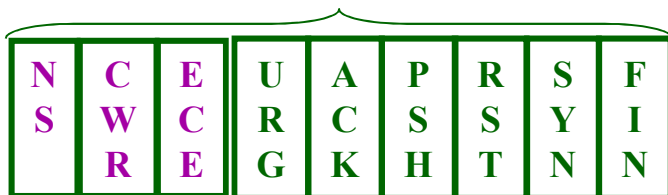
- **Well-known ports = 0-1023**
- **Inetd = super server can handle requests for multiple services**

Port	Protocol	Use
21	FTP	File transfer
23	Telnet	Remote login
25	SMTP	E-mail
69	TFTP	Trivial File Transfer Protocol
79	Finger	Lookup info about a user
80	HTTP	World Wide Web
110	POP-3	Remote e-mail access
119	NNTP	USENET news

TCP Header



TCP Header hossz (4)
 32 bites szavakban
 ⇒ max. 60 byte



- **Explicit Congestion Notification:**
- ECN-nonce sum (**NS**),
- Congestion window reduced (**CWR**),
- ECN-Echo (**ECE**): SYN:1 TCP peer ECN capable, SYN:0 IP Traffic Class: Congestion Experienced (utolsó 2 bit: 11) **E. IX. / 19.**

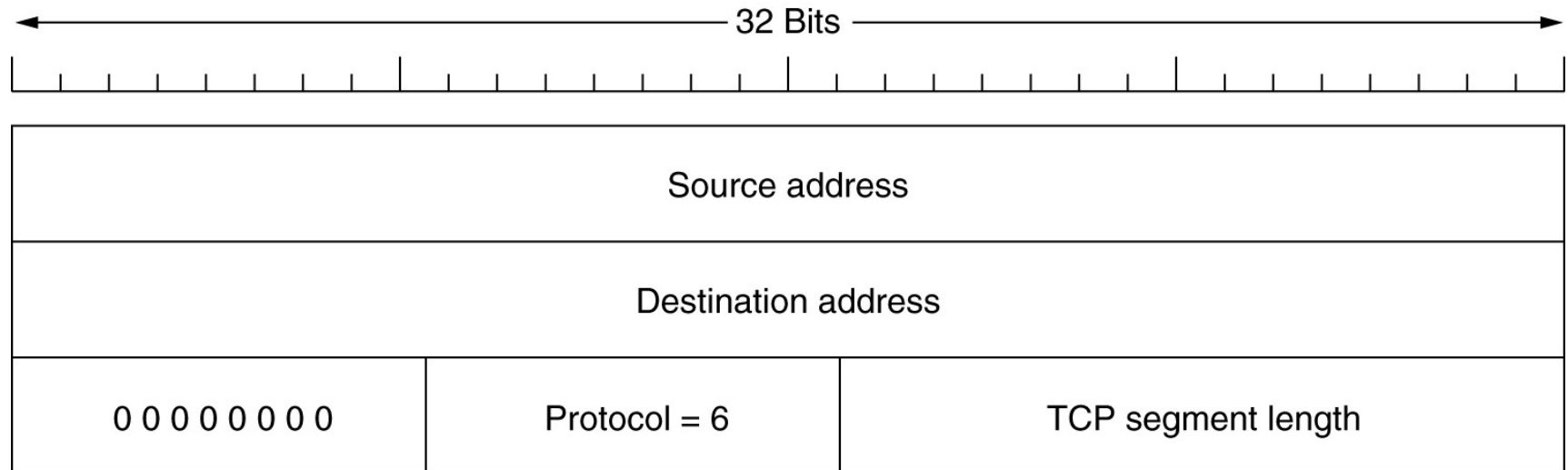
TCP Header

- **Socketpair:**
 - **Client IP, Client Port, Server IP, Server Port** négyes azonosítja a kapcsolatot.
- **Flag-ek**
 - **SYN:** új kapcsolat megnyitásakor (ezt jelzi) → „szinkronizáció”, → a sequence number ilyenkor: ISN (Initial SN) – kezdeti érték
 - **ACK:** a nyugta sorszáma érvényes (nyugta)
 - **URG: Urgent pointer** érvényes: a sürgős üzenet vége utáni pozícióra mutat, innét folytatódik a normál folyam (folyamon belüli (inbound) szolgálati üzenet, a feldolgozása előzze meg a normál folymot)
 - **PSH:** a vevő a lehető leggyorsabban továbbítsa az adatokat az alkalmazás felé
 - **RST: Reset Connection**
(azonnali kapcsolatbontás, bármely fél kezdheti, RST a válasz rá)
 - **FIN:** a küldő befejezte az adatok küldését

TCP Header

- **Opciók (Pl.)**
 - **MSS: Maximum Segment Size announcement**
 - A kapcsolat felépítésekor (SYN) mindkét oldal meghatározhatja a számára maximális szegmensméretet
 - A helyi MTU-t (Maximum Transmit Unit) küldi el (vagy MTU Discovery)
 - Azért, hogy minimalizálja az IP szinten a tördelést
 - Az MSS-t csak hirdetik (mekkorát fogad), a két oldal MSS értéke különböző lehet.
 - **Window Scale option**
 - A kapcsolat felépítésekor (SYN) mindkét oldal meghatározhatja a saját Window Scale értékét
 - 0 (no shift) to 14 for each direction independently
 - Ennyi bittel kell balra eltolni a Window Size értéket (0: 65,535 Byte, 14: 1GByte)

Pseudo header – used for checksum IPv4



- **TCP Checksum (16):**

- Header + Data + Pseudo header
- Az egész 1 komplement összegének (számításkor 0-nak veszi az ellenőrző összeg helyét)
- Negatív (1 komplement) előjellel vett értéke az ellenőrző összeg
- Ellenőrzéskor az egész összege így 0 (1 komplement)

**IPv6: Ugyanúgy mint az UDP esetében,
csak a Next header (Protocol) mező értéke 6.**

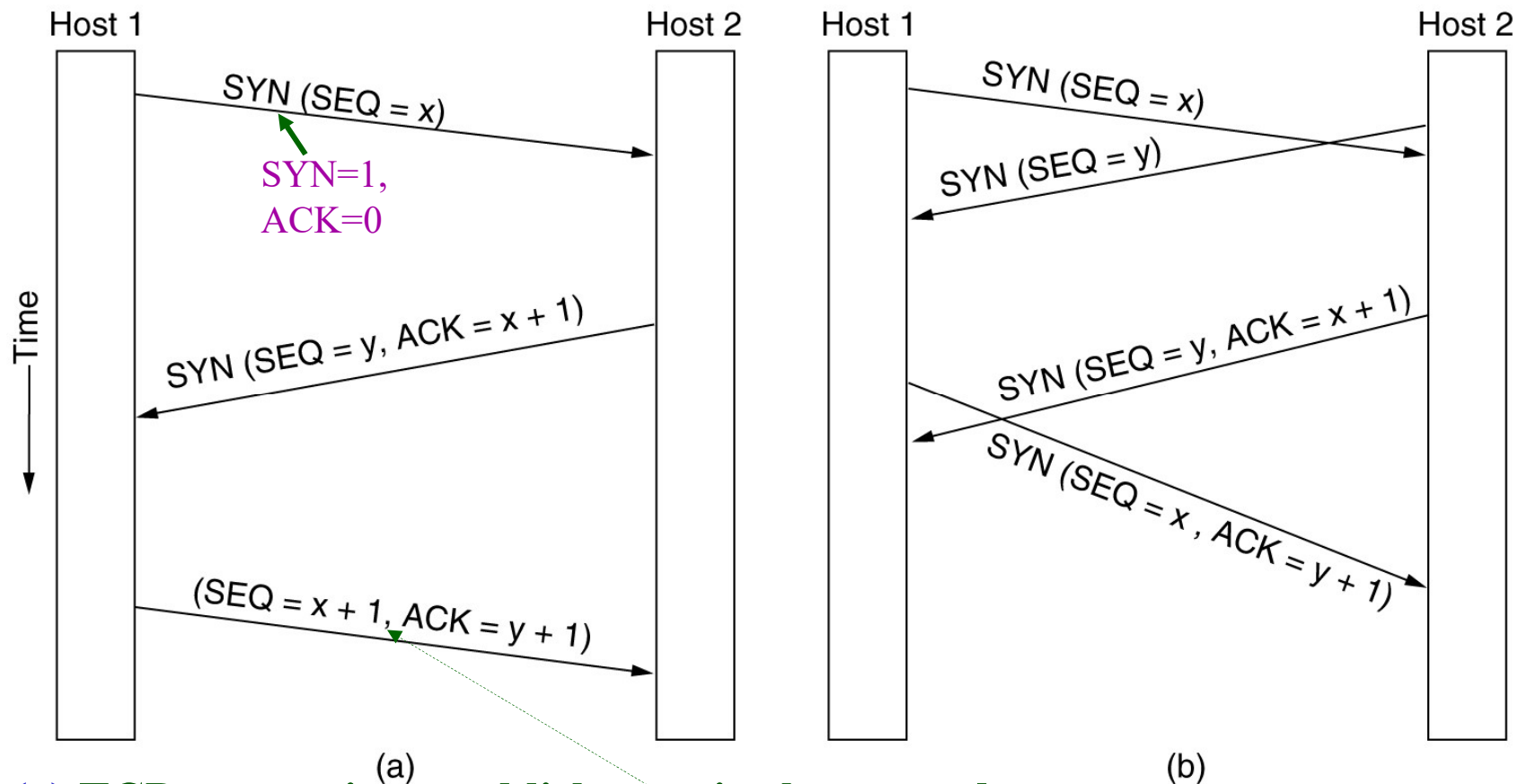
TCP connections

- Full duplex
- Byte stream
- Urgent data

TCP Protocol

- **Sliding window**
- **Timer**
- **Seq and ack are byte count**
- **Ack has next seq number expected**

TCP Connection Establishment

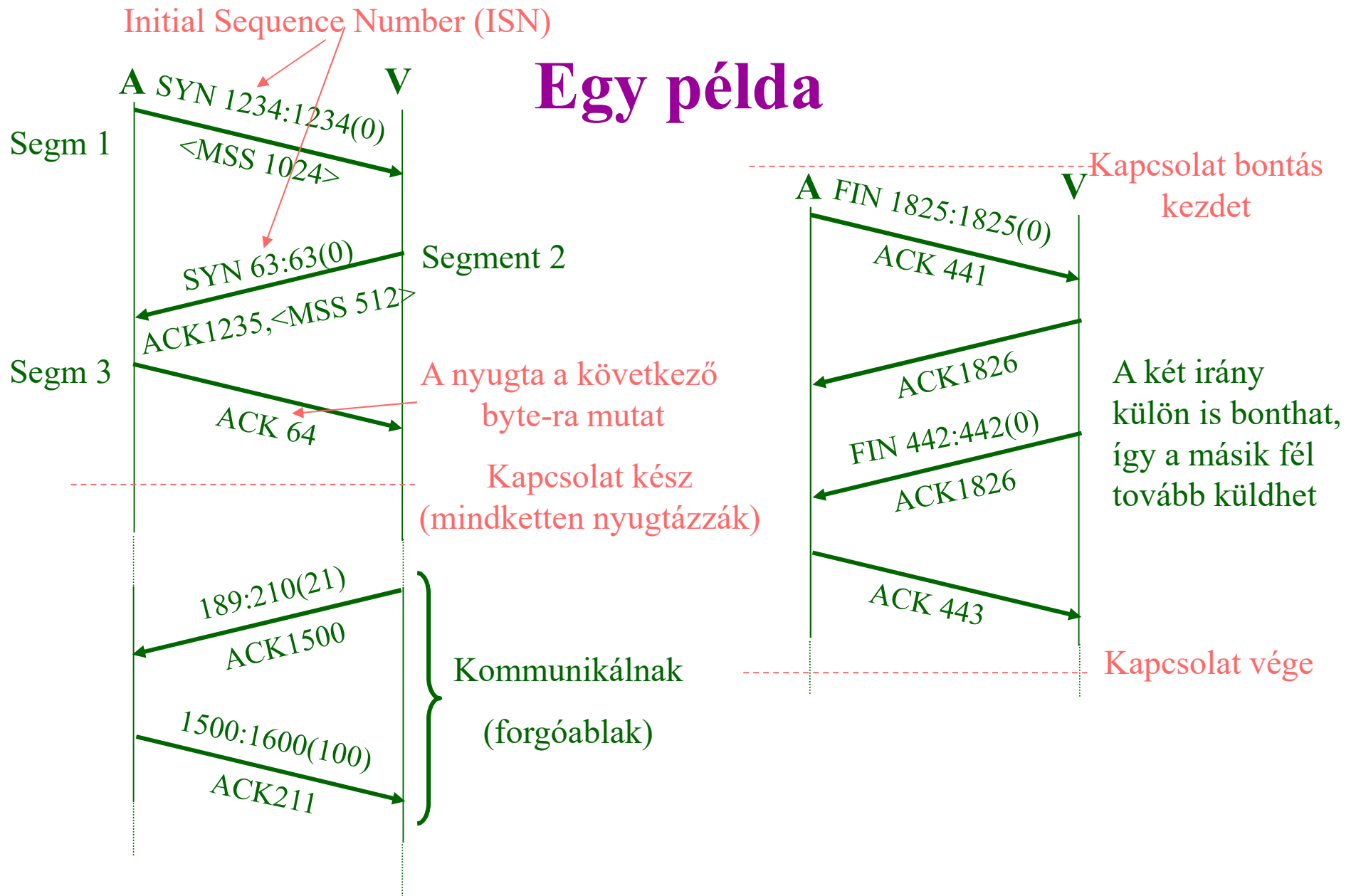


(a) TCP connection establishment in the normal case.

(b) Call collision – két kapcsolat indul egyszerre ugyanazon socketek között, de csak egy jön létre

A SYN szegmens (még ha az adat üres is) egy byte hosszú, ezért egyértelműen nyugtázható

Egy példa



TCP Finite State Machine

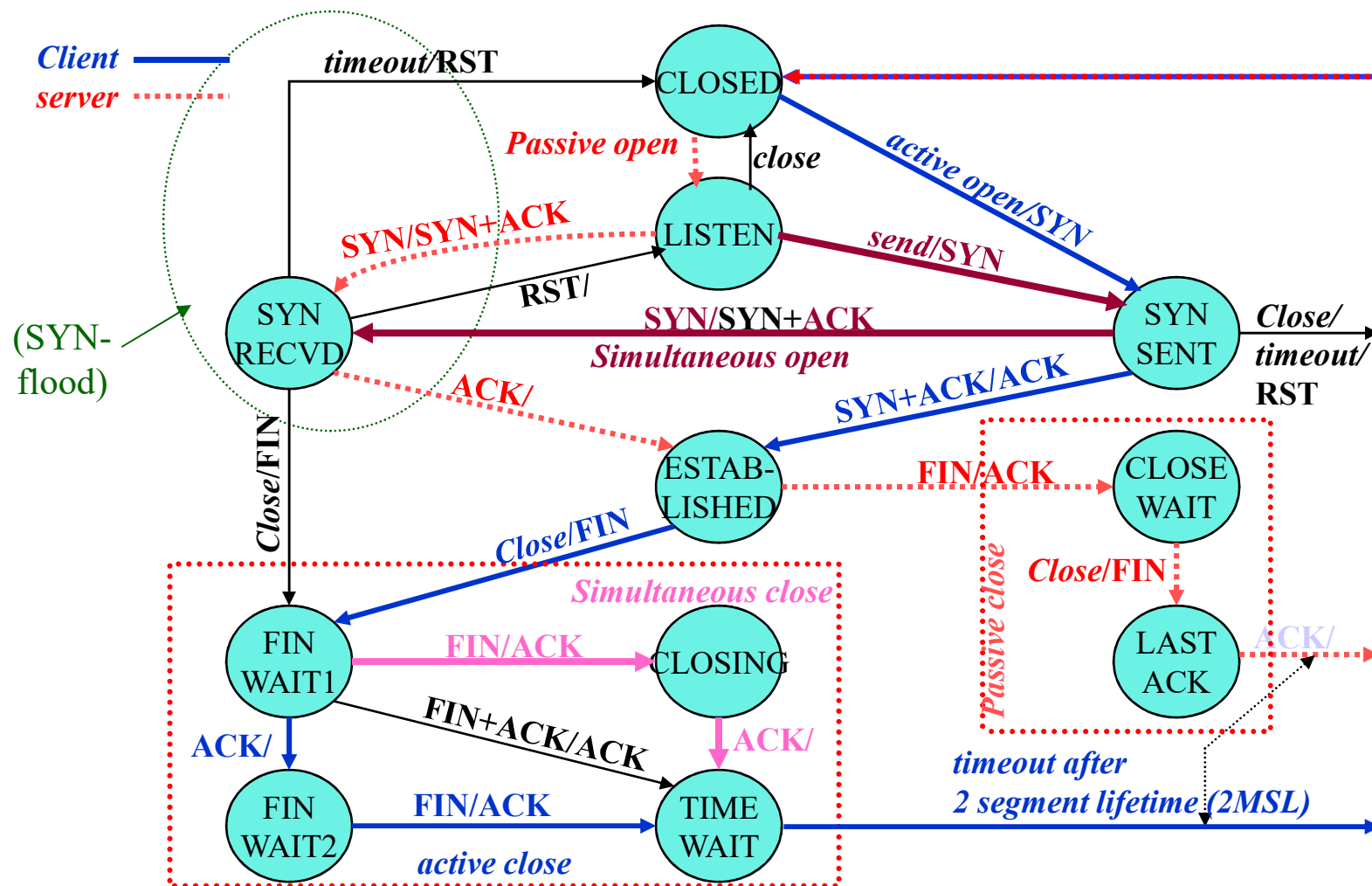
- TCP can best be explained with a theoretical model called a *finite state machine*.
- Various TCP states and their descriptions are:

– CLOSED	closed	} States involved in establishing a connection
– LISTEN	listening for connection	
– SYN SENT	active, have sent SYN	
– SYN RECEIVED	have sent and received SYN	
– ESTABLISHED	established connection	} States involved when remote end initiates shutdown
– CLOSED WAIT	have received FIN, waiting for close	
– LAST ACK	have received FIN and close, awaiting final ACK	
– CLOSED	closed	} States involved when local end initiates shutdown
– FIN WAIT 1	have closed, sent FIN	
– CLOSING	closed, exchanged FIN, awaiting final ACK	
– FIN WAIT 2	have closed, FIN is acknowledged, awaiting FIN	
– TIME WAIT	in 2MSL (MSL=30secs-2mins) wait after close	
– CLOSED	closed	

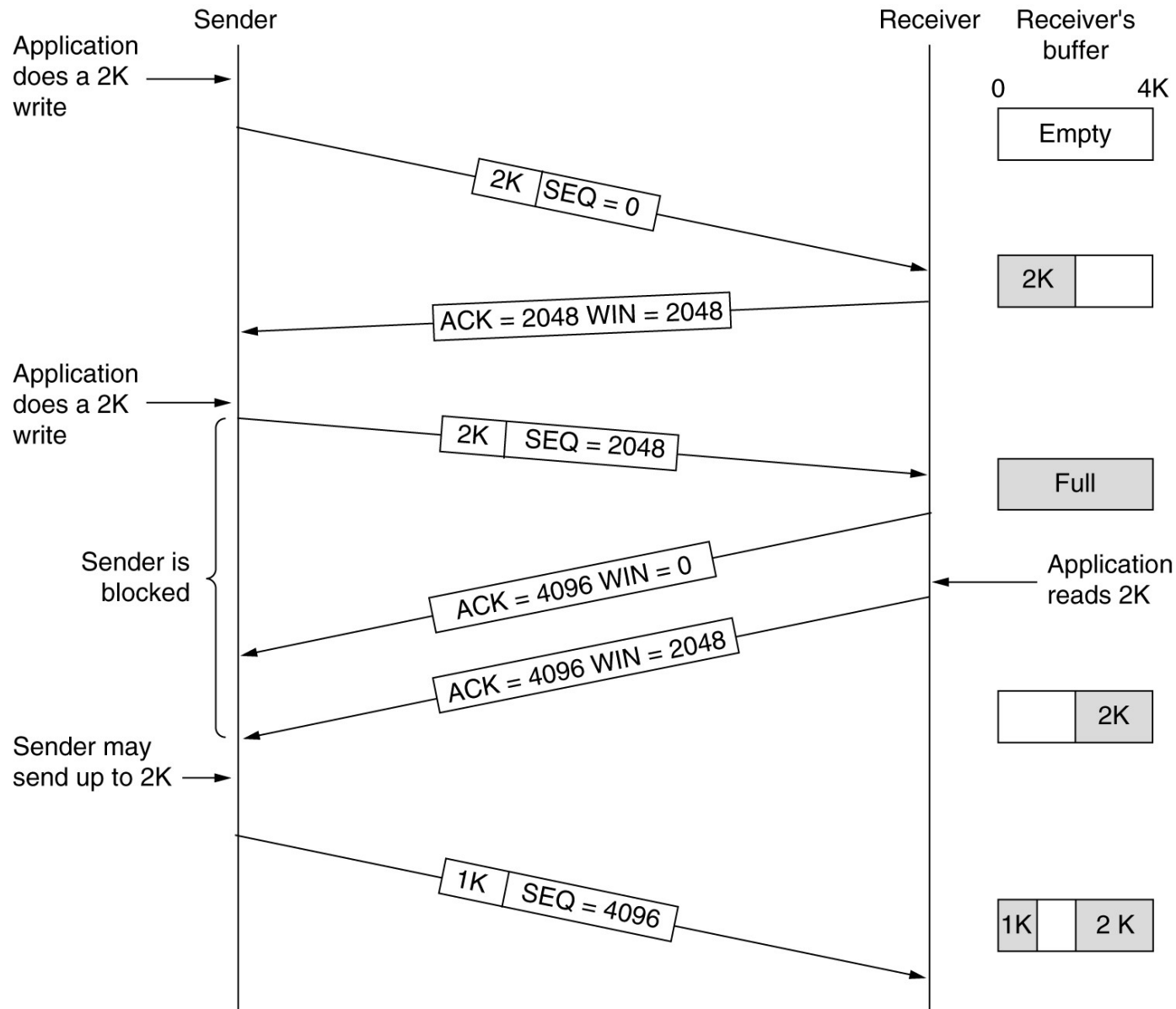
TCP Control segments

- **SYN = connection**
- **ACK = acknowledge**
- **FIN = end**
- **RST = error**

TCP Finite State Machine



TCP Transmission Policy: Window management



TCP timers

- **Retransmission** – set this timer when sending segment. When timer goes off retransmit segment.
- **Persistence** – set this timer when sender receives zero window size. When timer goes off sender sends probe segment.
- **Keep alive** – set this timer when sender sends segment or receiver receives segment. When timer goes off send probe.
- **TIMED WAIT** – set this timer when closing connection. When timer goes off remove connection record.

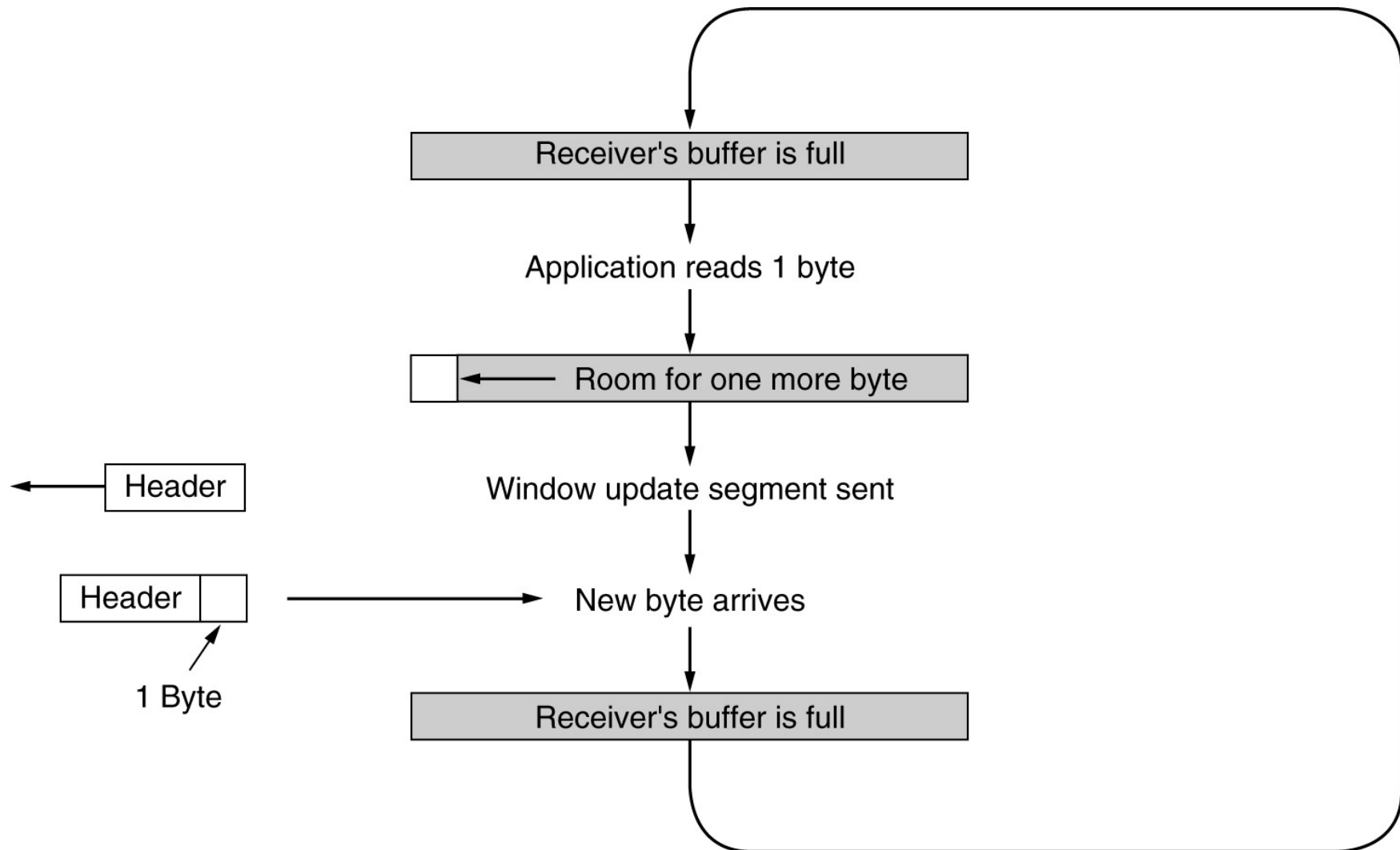
TCP Transmission Policy: Nagle algoritmus

- Ha a küldő egy byte-onként kapja a küldendő adatokat, elküldi az első byte-ot és küldés nélkül gyűjti a többi byte-ot, míg az elsőnek a nyugtája vissza nem érkezik és akkor küldi el az egészet egyben.
- Majd mindig megvárja az összes nyugtát mielőtt az újabb egységet küldené
- Csökkenti a sok kis (egy byte) csomag küldéséből adódó veszteséget (pl. telnet)

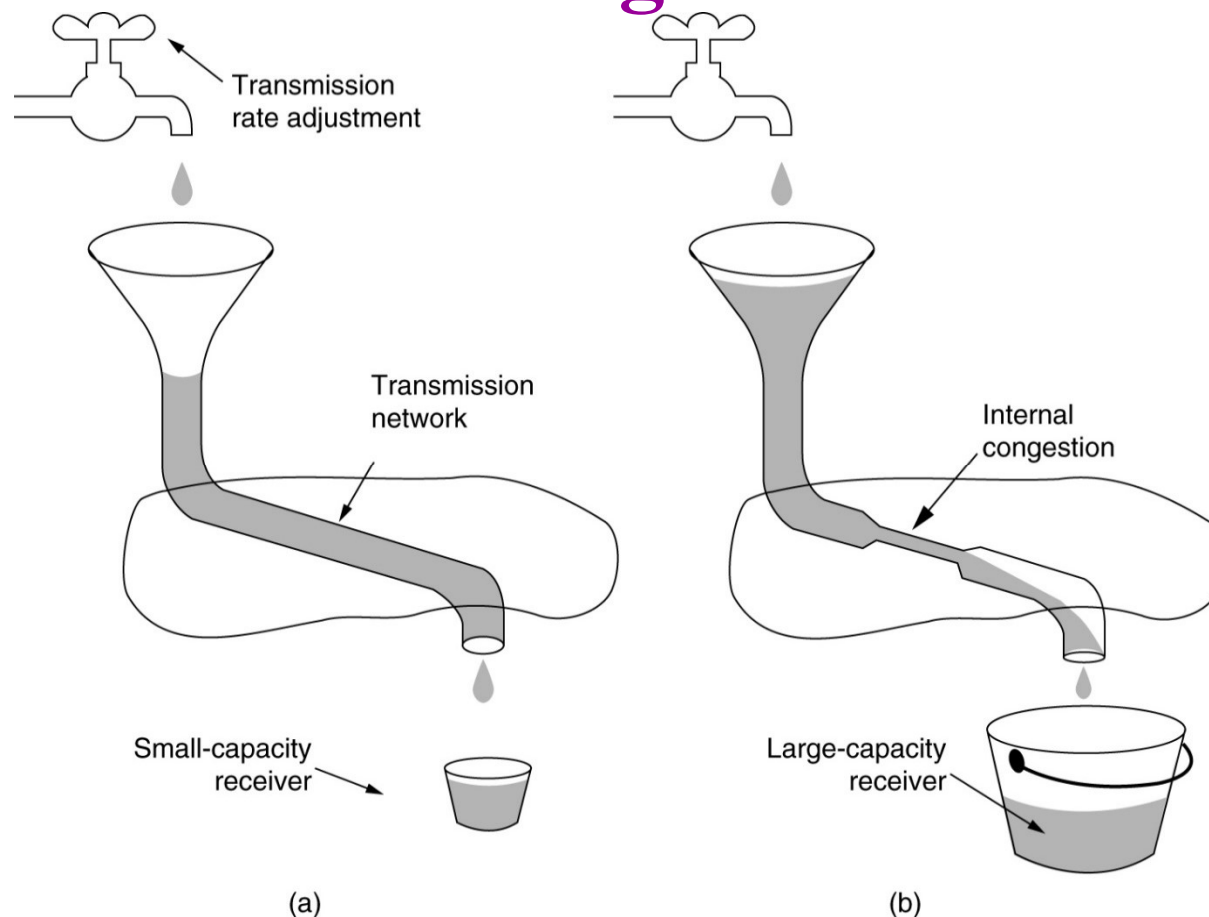
TCP Transmission Policy: Silly Window syndrome

- Akkor történik, ha az adatok nagy blokkokban érkeznek, de a interaktív alkalmazás csak **egy byte-onként olvassa** azokat.
- A vevőnek csak akkor kell **új vevőablak méretet** küldenie, ha már van elég helye (MTU or half buffer), nem pedig byte-onként.

TCP Transmission Policy: Silly Window syndrome



TCP Congestion Control



(a) A gyors hálózat alacsony kapacitású fogyasztót táplál.

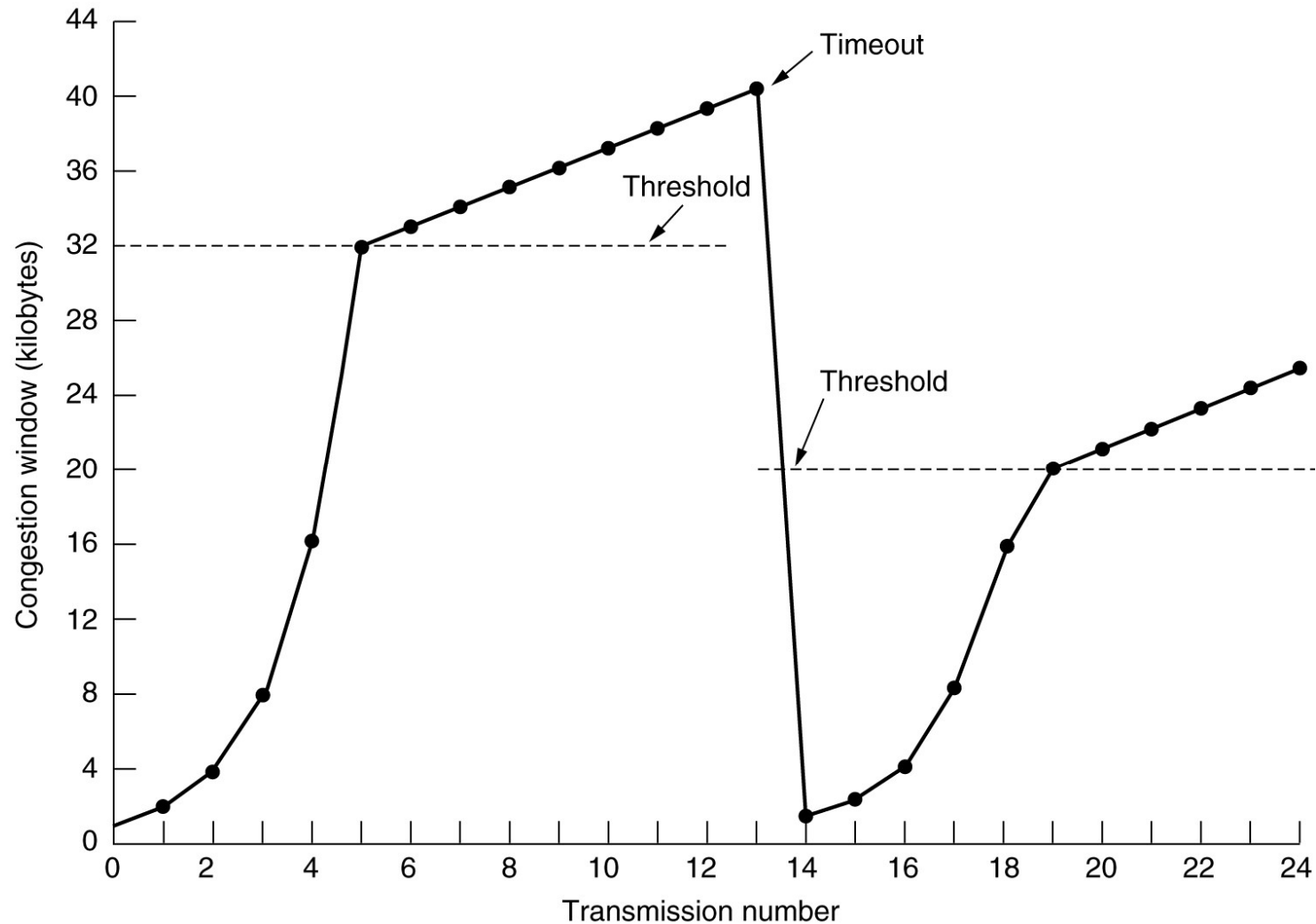
(b) A lassú hálózat nagy kapacitású fogyasztót táplál.

Két ablak adat az adóban: min (vevő ablak, torlódási ablak)

Congestion Control - Congestion Window

- **Slow start - Jacobson**
 - torlódási ablak = 1 MTU – val indul
 - Ha nyugtázzák, megduplázza a méretét
 - Folytatja
 - Exponenciálisan nő a mérete – a torlódási küszöbig
- **Torlódási küszöb (threshold) – initially 64 KB**
 - Időtúllépés esetén a torlódási küszöböt az aktuális torlódási ablak felére állítja, majd
 - újból „Slow start”,
 - de úgy, hogy csak a torlódási küszöbig exponenciális,
 - azt elérve sikeresség esetén is csak lineárisan nő
 - maximuma a vevő ablakméret (csak addig nőhet)
 - Az ICMP forrás folytatást = időtúllépésként értelmezi

TCP Congestion Control

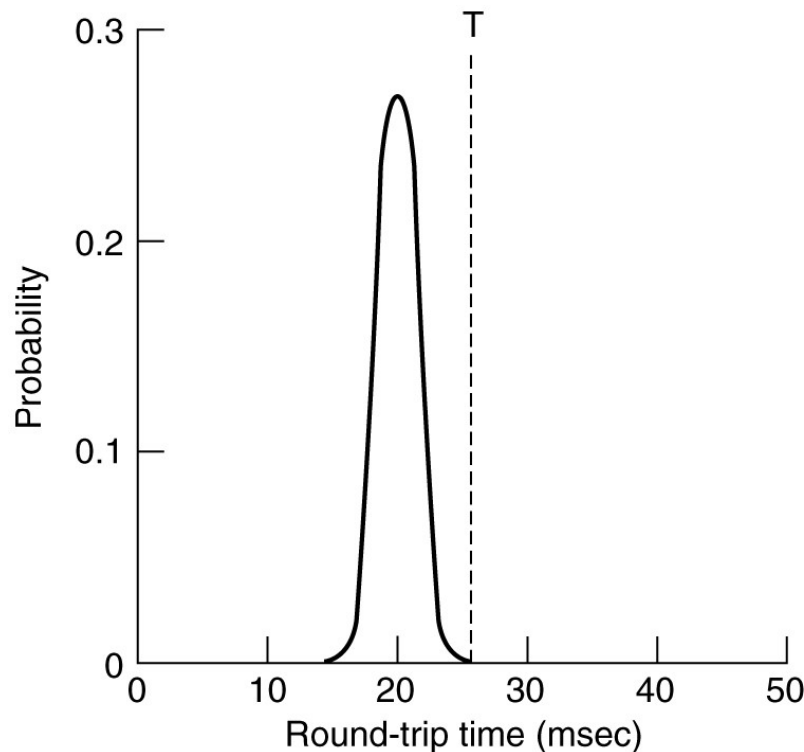


An example of the Internet congestion algorithm.

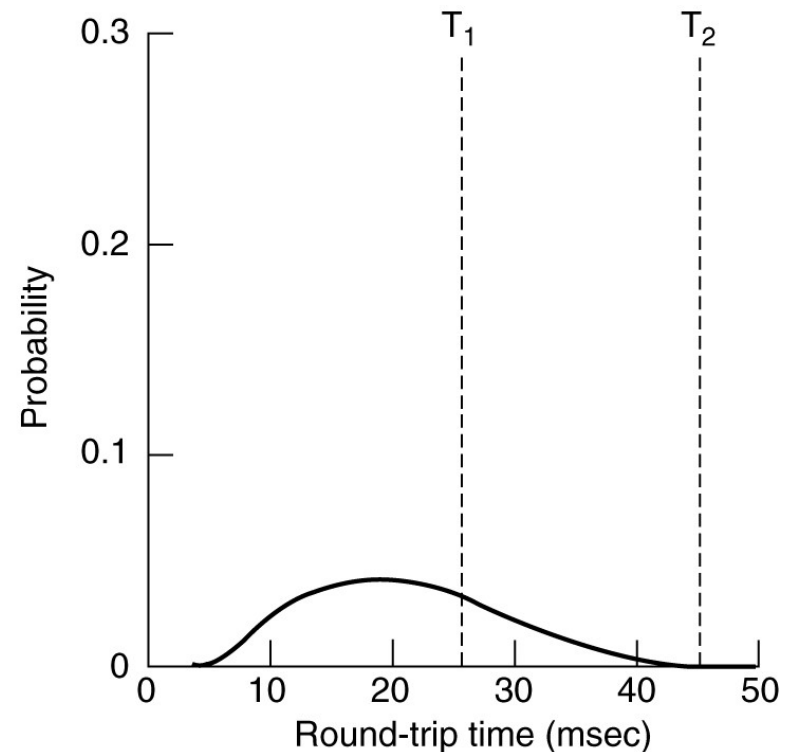
TCP Header - Explicit Congestion Notification

- **Flag-ek: Explicit Congestion Notification,**
köztes routerek torlódás jelzése adatvesztés nélkül
- **ECN-nonce sum (NS)** ellenőrző bit, hogy a vevő ne változtathassa meg a torlódás jelzését
- **ECN-Echo (ECE):**
 - SYN:1 TCP peer ECN capable,
 - SYN:0 IP Traffic Class: Congestion Experienced CE (utolsó 2 bit: 11) echo back the congestion indication, köztes routerek torlódás jelzése
- **Congestion window reduced (CWR):**
 - to acknowledge that the congestion-indication echoing was received.
- Ha a küldő TCP szegmenst kap az ECE bittel, akkor csökkenti a torlódási ablak méretét (congestion window) és visszajelez a fogadónak a szegmensekben a CWR bittel.
- A node keeps transmitting TCP segments with the ECE bit set until it receives a segment with the CWR bit set.

TCP Timer Management



(a)



(b)

- (a) Probability density of ACK arrival times in the data link layer.**
- (b) Probability density of ACK arrival times for TCP.**

Jacobson

- $RTT = \text{round trip time}$
- $RTT = \alpha RTT + (1 - \alpha)M$, ahol M a legutóbbi ack time
- Tipikusan $\alpha = 7/8$
- $\text{Time out} = \beta RTT$
ahol eleinte $\beta=2$, majd β a nyugta beérkezés
sűrűségfüggvényének szórásával arányos
- A szórás becslése csúszóátlagolással:
 $D = \alpha D + (1 - \alpha)|RTT - M|$,
- $\text{Timeout} = RTT + 4 D$
- Karn: IP over radio
 - Ne frissítsők az RTT-t az újraküldött szegmensekkel
 - Duplázzuk a timeout-ot minden hiba esetén