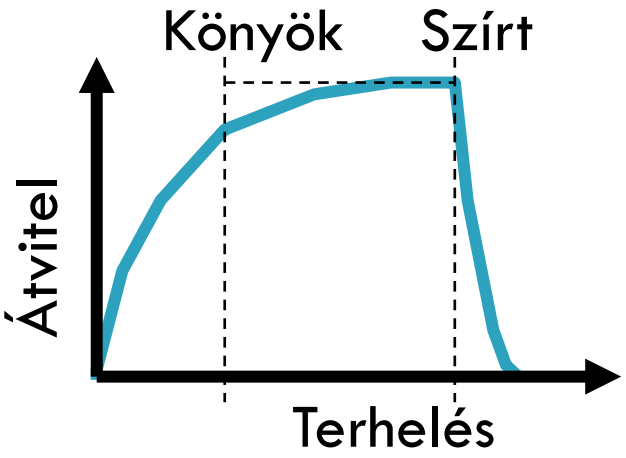
Számítógépes Hálózatok

12. Előadás: Szállítói réteg 2

Based on slides from **Zoltán Ács ELTE** and D. Choffnes Northeastern U., Philippa Gill from StonyBrook University , Revised Spring 2016 by S. Laki

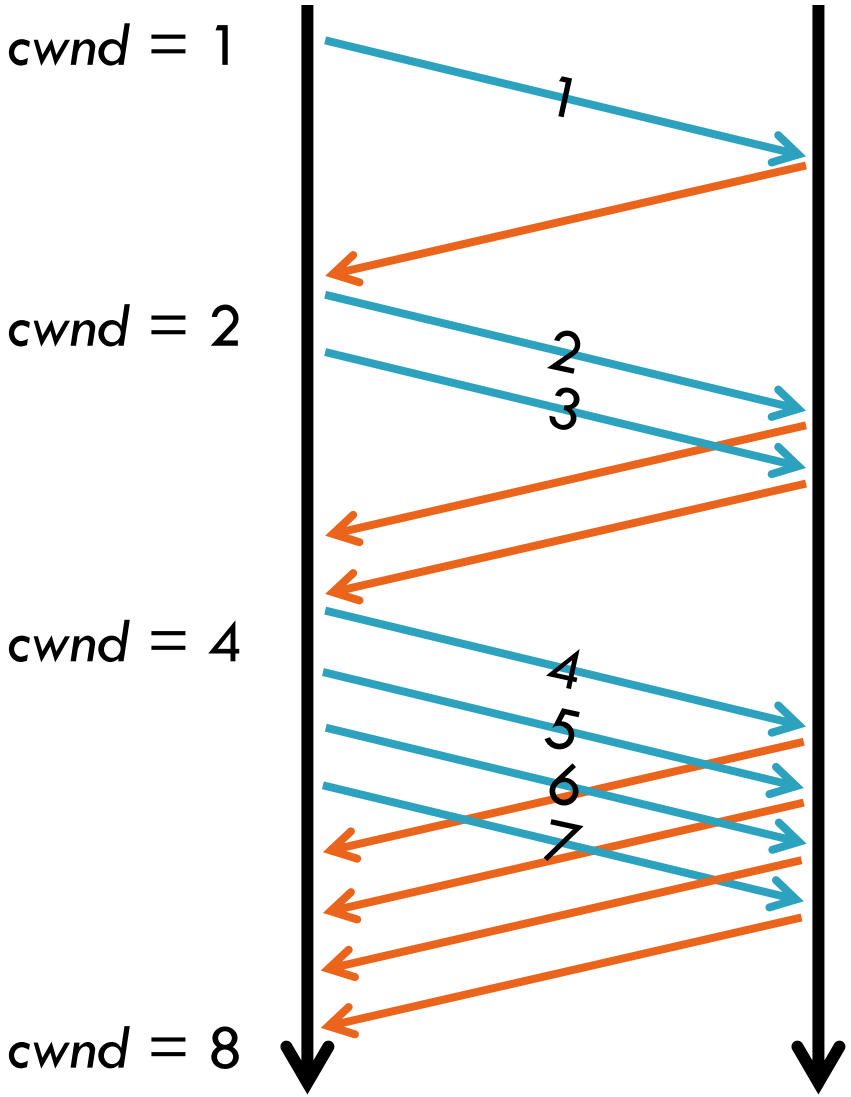
Lassú indulás - Slow Start

- Cél, hogy gyorsan elérjük a könyök pontot
- Egy kapcsolat kezdetén (vagy újraindításakor)
  - *cwnd* =1
  - *ssthresh* = *adv_wnd*
  - Minden nyugtázott szegmensre: *cwnd*++
- Egészen addig amíg
  - El nem érjük az *ssthresh* értéket
  - Vagy csomagvesztés nem történik
- A Slow Start valójában nem lassú
  - *cwnd* exponenciálisan nő



Slow Start példa

- *cwnd* gyorsan nő
- Lelassul, amikor…
  - *cwnd* >= *ssthresh*
  - Vagy csomagvesztés történik

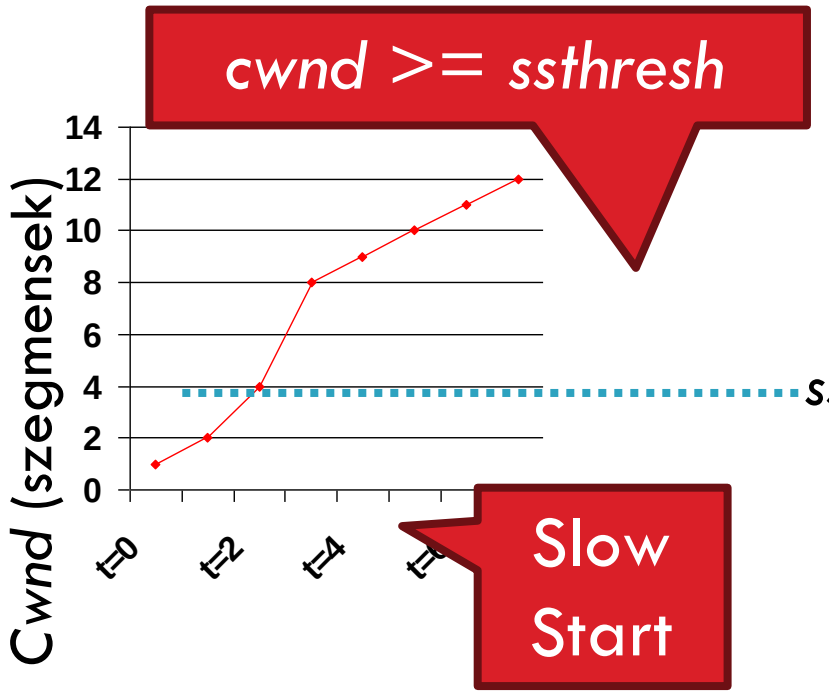


Torlódás elkerülés

- Additive Increase Multiplicative Decrease (AIMD) mód
- *ssthresh* valójában egy alsóbecslés a könyök pontra
- **Ha** *cwnd* >= *ssthresh* **akkor**

  Minden nyugtázott szegmens alkalmával növeljük a *cwnd értékét (1/cwnd)-vel (azaz cwnd += 1/cwnd)*.
- Azaz a *cwnd* eggyel nő, ha minden csomag nyugtázva lett.

Torlódás elkerülés példa



ssthresh = 8

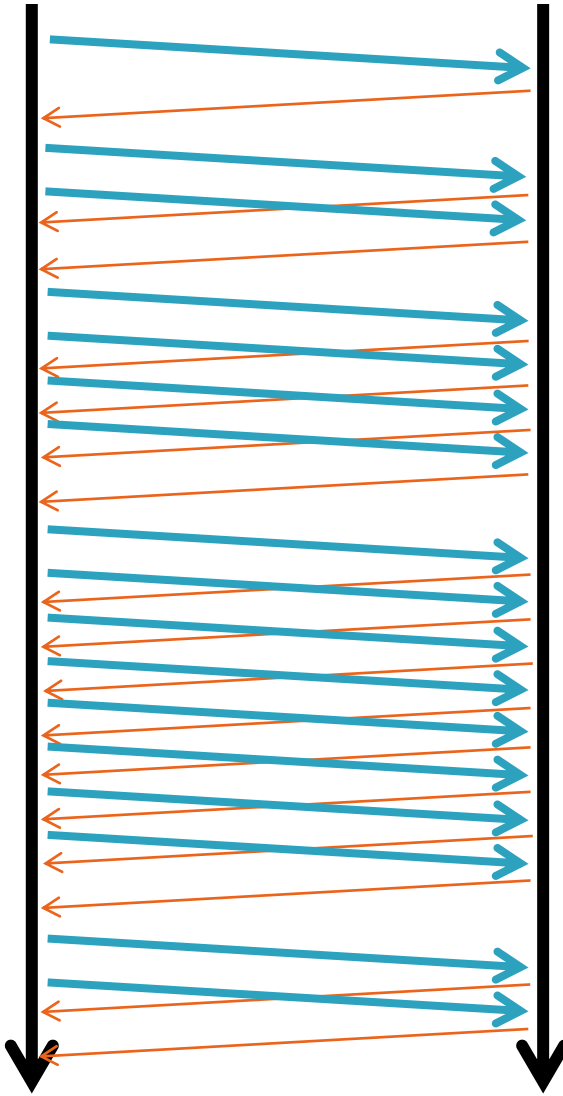
cwnd = 1

cwnd = 2

cwnd = 4

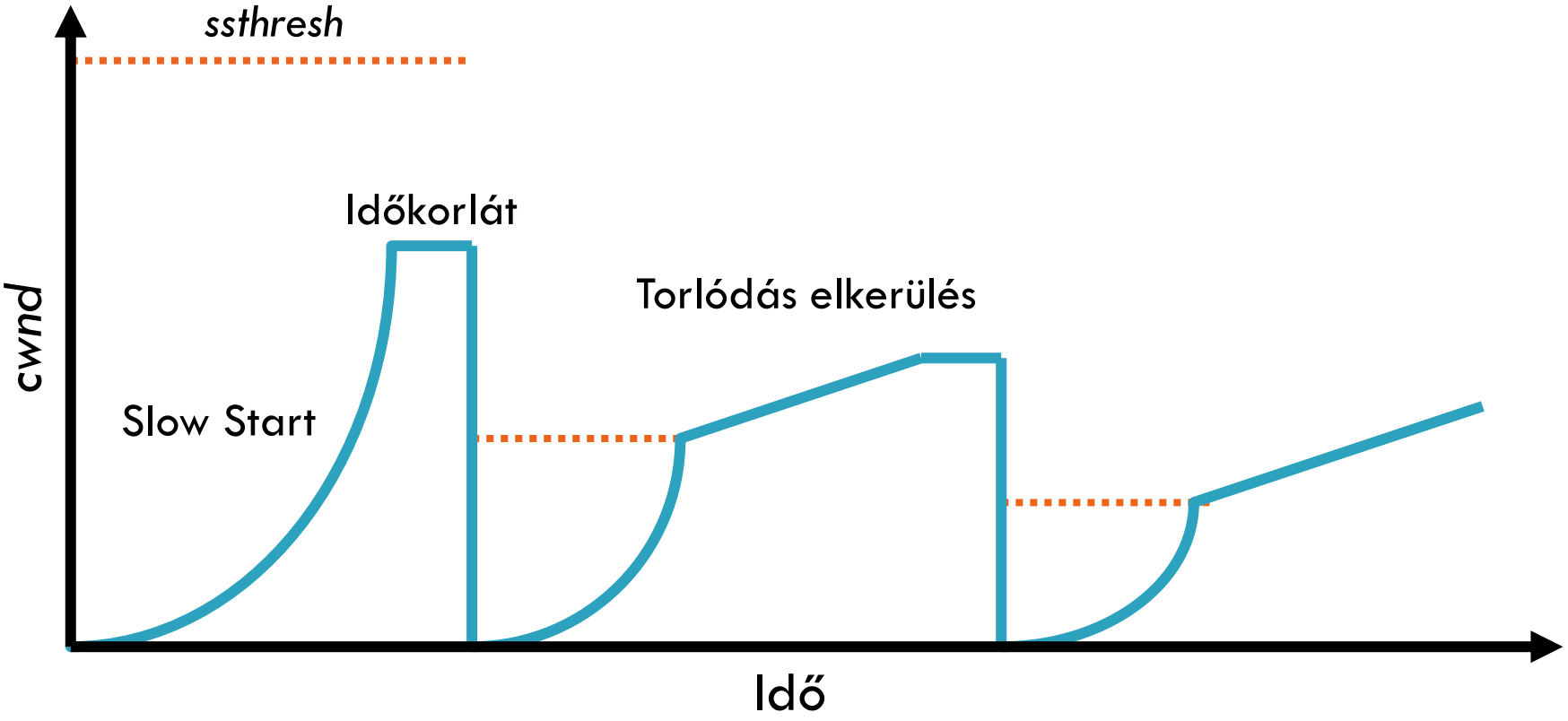
cwnd = 8

cwnd = 9



Round Trip Times

A teljes kép – TCP Tahoe (az eredeti TCP)



Összefoglalás - TCP jellemzői

„A *TCP* egy kapcsolatorientált megbízható szolgáltatás kétirányú bájt-folyamokhoz.”

KAPCSOLATORIENTÁLT

- Két résztvevő, ahol egy résztvevőt egy *IP-cím* és egy *port* azonosít.
- A kapcsolat egyértelműen azonosított a résztvevő párral.
- Nincs se *multi-*, se *broadcast* üzenetküldés.
- A kapcsolatot fel kell építeni és le kell bontani.
- Egy kapcsolat a lezárásáig aktív.

Összefoglalás - TCP jellemzői

„A *TCP* egy kapcsolatorientált megbízható szolgáltatás kétirányú bájt-folyamokhoz.”

MEGBÍZHATÓSÁG

- Minden csomag megérkezése nyugtázásra kerül.
- A nem nyugtázott adatcsomagokat újraküldik.
- A fejléchez és a csomaghoz ellenőrzőösszeg van rendelve.
- A csomagokat számozza, és a fogadónál sorba rendezésre kerülnek a csomagok a sorszámaik alapján.
- Duplikátumokat törli.

Összefoglalás - TCP jellemzői

„A *TCP* egy kapcsolatorientált megbízható szolgáltatás kétirányú bájt-folyamokhoz.”

KÉTIRÁNYÚ BÁJTFOLYAM

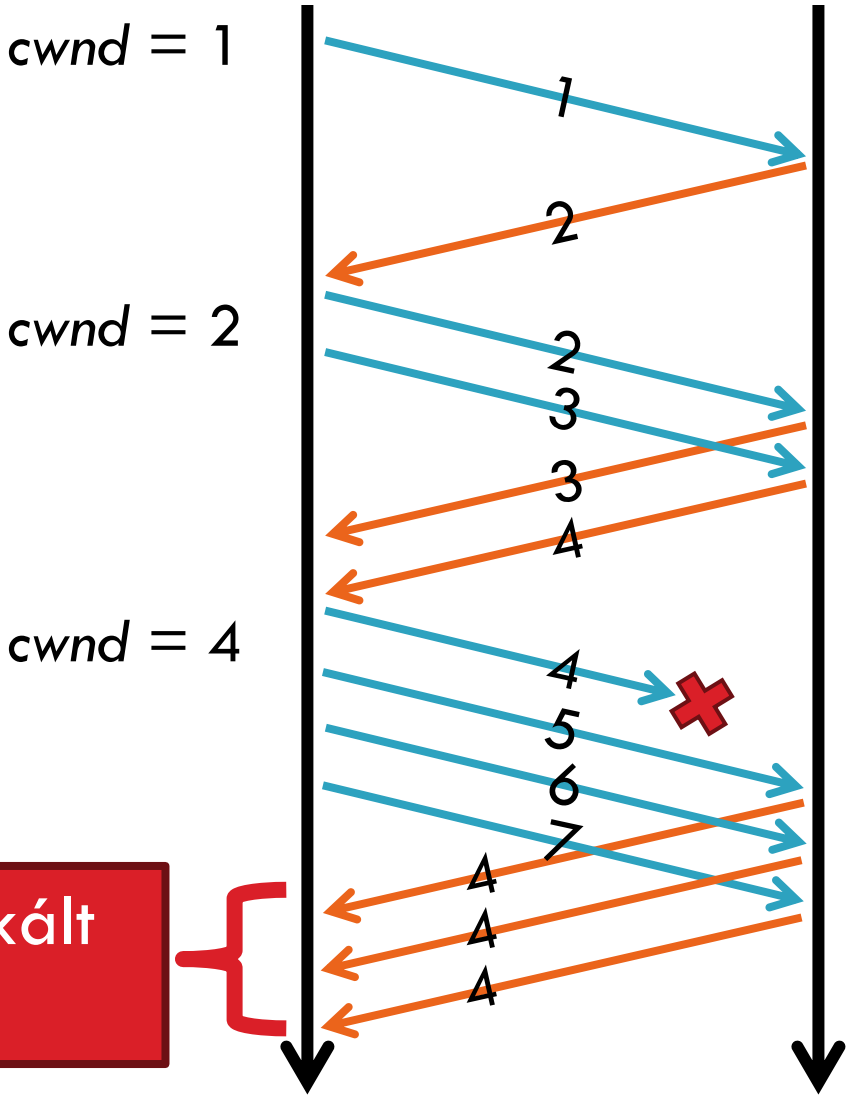
- Az adatok két egymással ellentétes irányú bájt-sorozatként kerülnek átvitelre.
- A tartalom nem interpretálódik.
- Az adatcsomagok időbeli viselkedése megváltozhat: átvitel sebessége növekedhet, csökkenhet, más késés, más sorrendben is megérkezhetnek.
- Megpróbálja az adatcsomagokat időben egymáshoz közel kiszállítani.
- Megpróbálja az átviteli közeget hatékonyan használni.

A TCP evolúciója

- Az eddigi megoldások a TCP Tahoe működéshez tartoztak
  - Eredeti TCP
- A TCP-t 1974-ben találták fel!
  - Napjainkba számos változata létezik
- Kezdeti népszerű változat: TCP Reno
  - Tahoe lehetőségei, plusz…
  - Gyors újraküldés (Fast retransmit)
    - 3 duplikált ACK? -> újraküldés (ne várjunk az RTO-ra)
  - Gyors helyreállítás (Fast recovery)
    - Csomagvesztés esetén:
      - set cwnd = cwnd/2 (ssthresh = az új cwnd érték)

TCP Reno: Gyors újraküldés

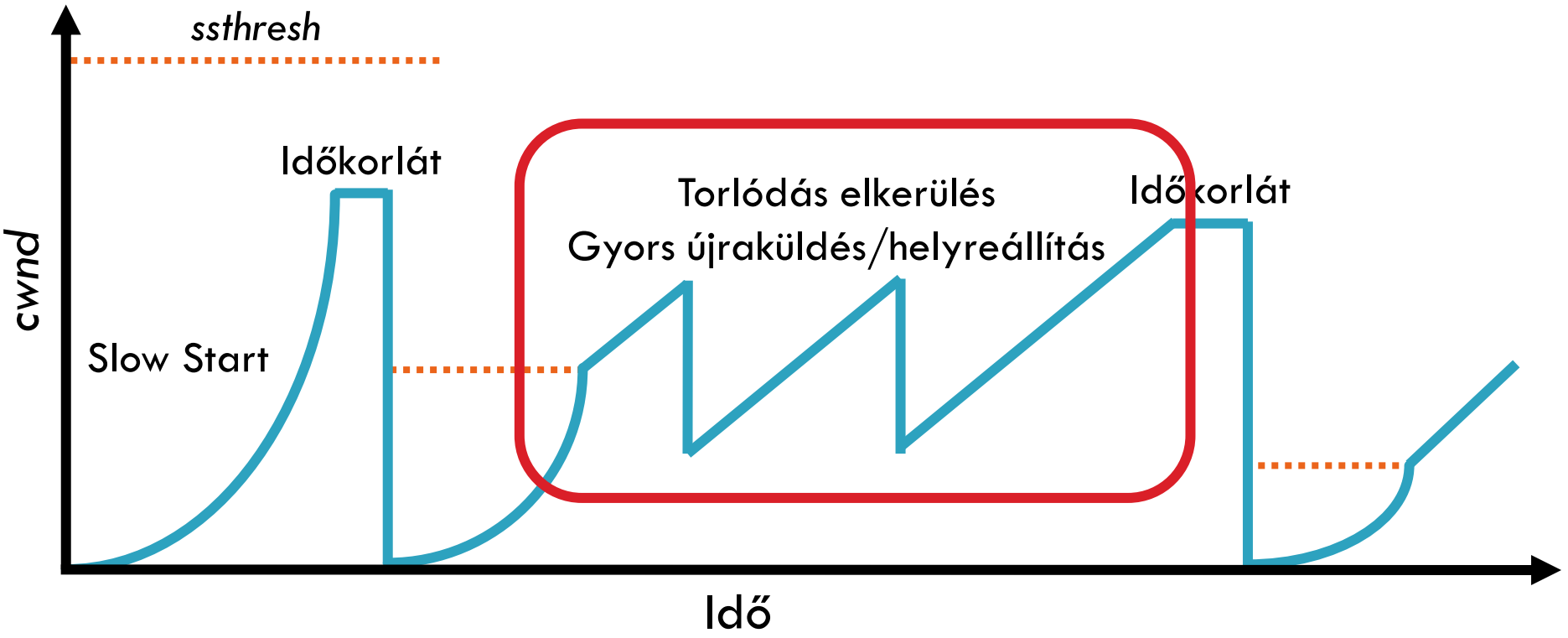
- Probléma: Tahoe esetén ha egy csomag elveszik, akkor hosszú a várakozás az RTO-ig
- Reno: újraküldés 3 duplikált nyugta fogadása esetén
- Duplikált: ugyanaz a sorszám
  - Explicit jele a csomagvesztésnek



TCP Reno: Gyors helyreállítás

- Gyors újraküldés után módosítjuk a torlódási ablakot:
  - *cwnd* := *cwnd*/2 *(valójában ez a Multiplicative Decrease)*
  - ssthresh := az új cwnd
  - Azaz nem álltjuk vissza az eredeti 1-re a cwnd-t!!!
  - Ezzel elkerüljük a felesleges slow start fázisokat!
  - Elkerüljük a költséges időkorlátokat
- Azonban ha az RTO lejár, továbbra is *cwnd* = 1
  - Visszatér a slow start fázishoz, hasonlóan a Tahoe-hoz
  - Olyan csomagokat jelez, melyeket egyáltalán nem szállítottunk le
  - A torlódás nagyon súlyos esetére figyelmeztet!!!

Példa: Gyors újraküldés/helyreállítás



- Stabil állapotban, a cwnd az optimális ablakméret körül oszcillál
- TCP mindig csomagdobásokat kényszerít ki…

Számos TCP változat…

- Tahoe: az eredeti
  - Slow start és AIMD
  - Dinamikus RTO, RTT becsléssel
- Reno:
  - fast retransmit (3 dupACKs)
  - fast recovery (cwnd = cwnd/2 vesztés esetén)
- NewReno: javított gyors újraküldés
  - Minden egyes duplikált ACK újraküldést vált ki
  - Probléma: >3 hibás sorrendben fogadott csomag is újraküldést okoz (hibásan!!!)…
- Vegas: késleltetés alapú torlódás elkerülés
- …

TCP a valóságban

- Mi a legnépszerűbb variáns napjainkban?
  - Probléma: TCP rosszul teljesít nagy késleltetés-sávszélesség szorzattal rendelkező hálózatokban (a modern Internet ilyen)
  - Compound TCP (Windows)
    - Reno alapú
    - Két torlódási ablak: késleltetés alapú és vesztés alapú
    - Azaz egy összetett torlódás vezérlést alkalmaz
  - TCP CUBIC (Linux)
    - Fejlettebb BIC (Binary Increase Congestion Control) változat
    - Az ablakméretet egy harmadfokú egyenlet határozza meg
    - A legutolsó csomagvesztéstől eltelt T idővel paraméterezett

Nagy késleltetés-sávszélesség szorzat (Delay-bandwidth product)

- Probléma: A TCP nem teljesít jól ha
  - A hálózat kapacitása (sávszélessége) nagy
  - A késleltetés (RTT) nagy
  - Vagy ezek szorzata nagy
    - b * d = maximális szállítás alatt levő adatmennyiség
    - Ezt nevezzük késleltetés-sávszélesség szorzatnak
- Miért teljesít ekkor gyengén a TCP?
  - A slow start és az additive increase csak lassan konvergál
  - A TCP ACK ütemezett (azaz csak minden ACK esetén történik esemény)
    - A nyugták beérkezési gyorsasága határozza meg, hogy milyen gyorsan tud reagálni
    - Nagy RTT → késleltetett nyugták → a TCP csak lassan reagál a megváltozott viszonyokra

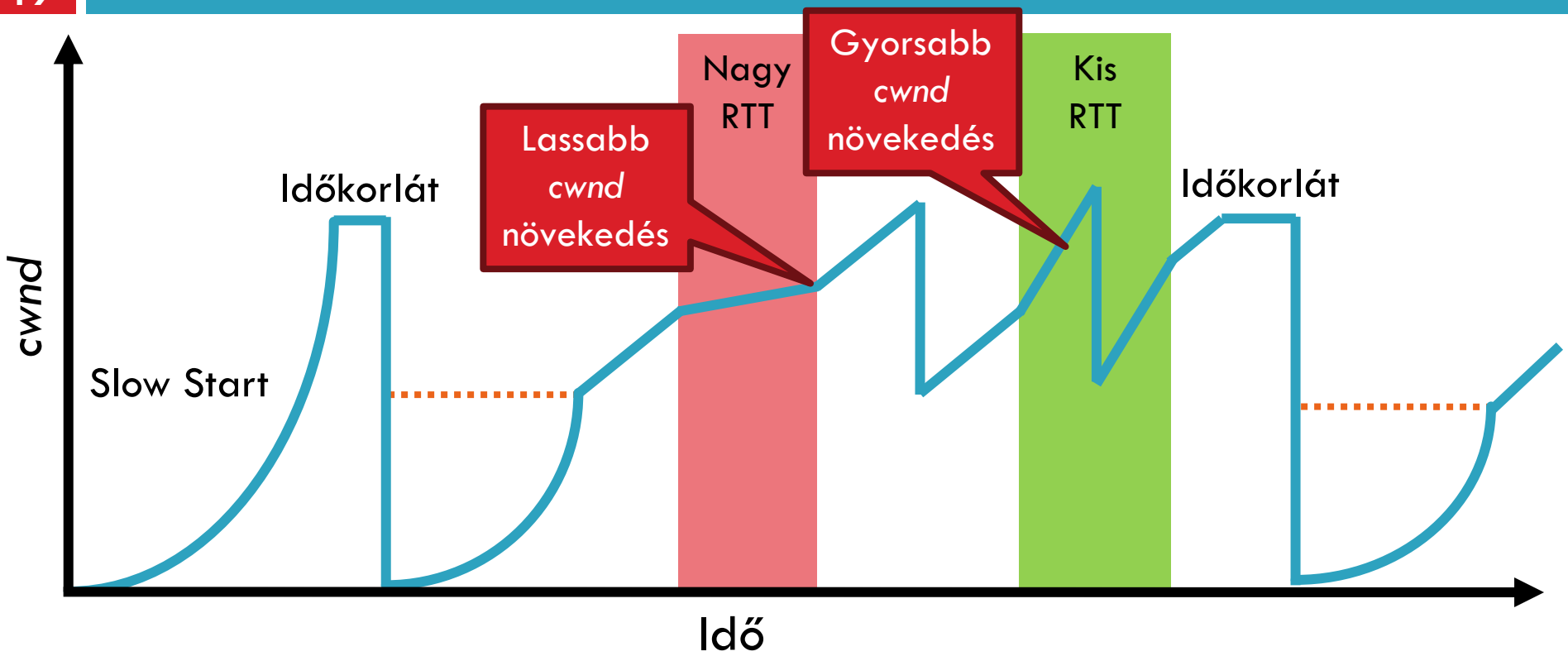
Célok

- A TCP ablak gyorsabb növelése
  - A slow start és az additive increase túl lassú, ha nagy a sávszélesség
  - Sokkal gyorsabb konvergencia kell
- Fairség biztosítása más TCP változatokkal szemben
  - Az ablak növelése nem lehet túl agresszív
- Javított RTT fairség
  - A TCP Tahoe/Reno folyamok nem adnak fair erőforrás-megosztást nagyon eltérő RTT-k esetén
- Egyszerű implementáció

Compound TCP

- Alap TCP implementáció Windows rendszereken
- Ötlet: osszuk a *torlódási ablakot* két különálló ablakba
  - Hagyományos, vesztés alapú ablak
  - Új, késleltetés alapú ablak
- *wnd* = min(*cwnd* + *dwnd*, *adv_wnd*)
  - *cwnd*-t az AIMD vezérli AIMD
  - *dwnd* a késleltetés alapú ablak
- A *dwnd* beállítása:
  - Ha nő az RTT, csökken a *dwnd* (*dwnd* >= 0)
  - Ha csökken az RTT, nő a *dwnd*
  - A növekesés/csökkenés arányos a változás mértékével

Compound TCP példa



- Agresszívan reagál az RTT változására
- Előnyök: Gyors felfutás, sokkal fairebb viselkedés más folyamokkal szemben eltérő RTT esetén
- Hátrányok: folyamatos RTT becslés

TCP CUBIC

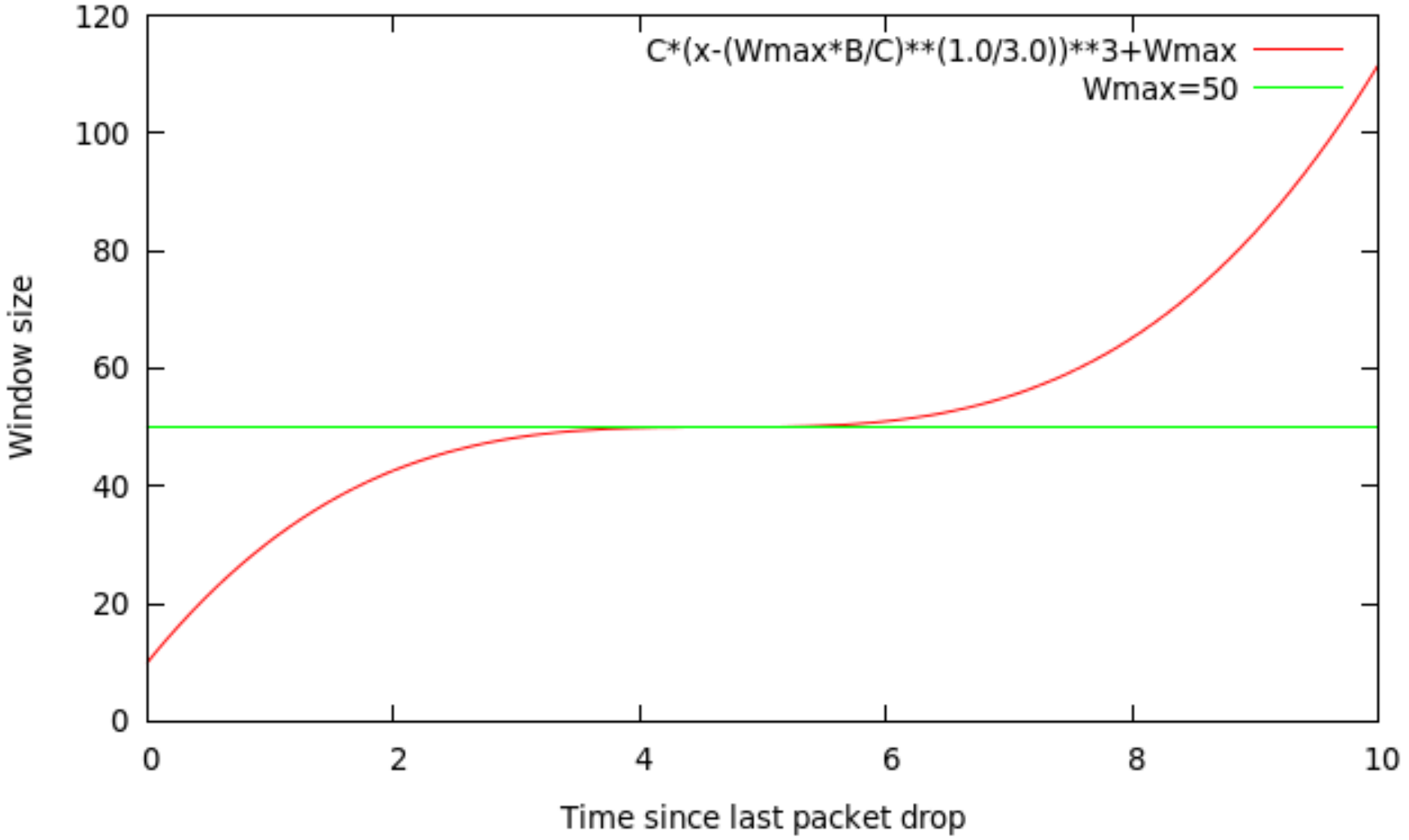
- Alap TCP implementáció Linux rendszereken
- Az AIMD helyettesítése egy „köbös” (CUBIC) függvénnyel

$$W_{cubic} = C(T - K)^3 + W_{max} \quad (1)$$

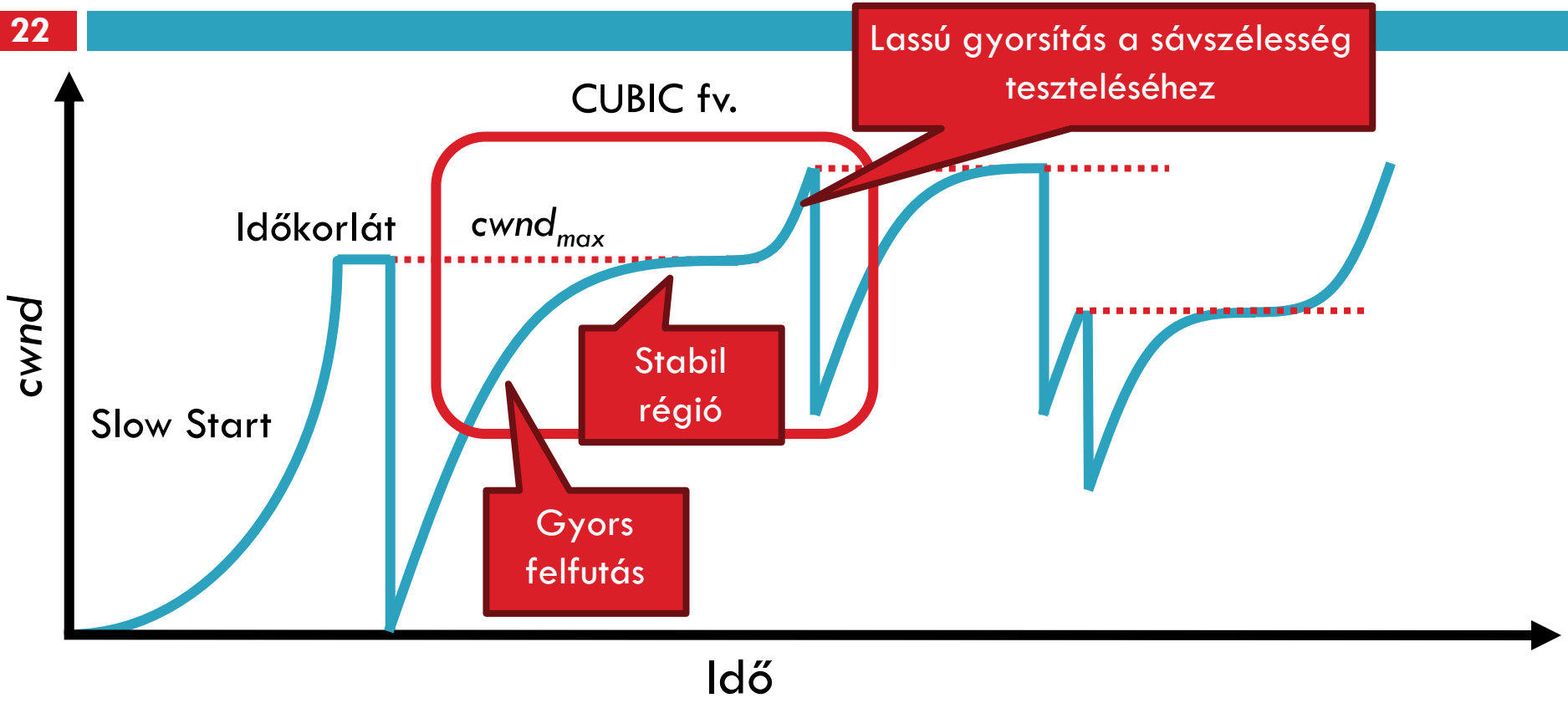
$$\text{C is a scaling constant, and K} = \sqrt[3]{\frac{W_{max}\beta}{C}}$$

  - B → egy konstans a multiplicative increase fázishoz
  - T → eltelt idő a legutóbbi csomagvesztés óta
  - W_max → cwnd a legutolsó csomagvesztés idején

TCP CUBIC


C*(x-(Wmax*B/C)**(1.0/3.0))**3+Wmax
Wmax=50
Window size
Time since last packet drop
0
20
40
60
80
100
120
0
2
4
6
8
10

TCP CUBIC példa



- Kevésbé pazarolja a sávszélességet a gyors felfutások miatt
- A stabil régió és a lassú gyorsítás segít a fairség biztosításában
  - A gyors felfutás sokkal agresszívabb, mint az additive increase
  - A Tahoe/Reno variánsokkal szembeni fairséghez a CUBIC-nak nem szabad ennyire agresszívnak lennie

Problémák a TCP-vel

- Az Internetes forgalom jelentős része TCP
- Azonban számos probléma okozója is egyben
  - Gyenge teljesítmény kis folyamok esetén
  - Gyenge teljesítmény wireless hálózatokban
  - DoS támadási felület

Kis folyamok (flows)

- Probléma: kis folyamok esetén torz viselkedés
  - 1 RTT szükséges a kapcsolat felépítésére (SYN, SYN/ACK)
    - pazarló
  - *cwnd* mindig 1-gyel indul
    - Nincs lehetőség felgyorsulni a kevés adat miatt
- Az Internetes forgalom nagy része kis folyam
  - Többnyire HTTP átvitel, <100KB
  - A legtöbb TCP folyam el se hagyja a slow start fázist!!!
- Lehetséges megoldás (Google javaslat):
  - Kezdeti *cwnd* megnövelése 10-re
  - TCP Fast Open: kriptográfiai hashek használata a fogadó azonosítására, a három-utas kézfogás elhagyható helyette hash (cookie) küldése a syn csomagban

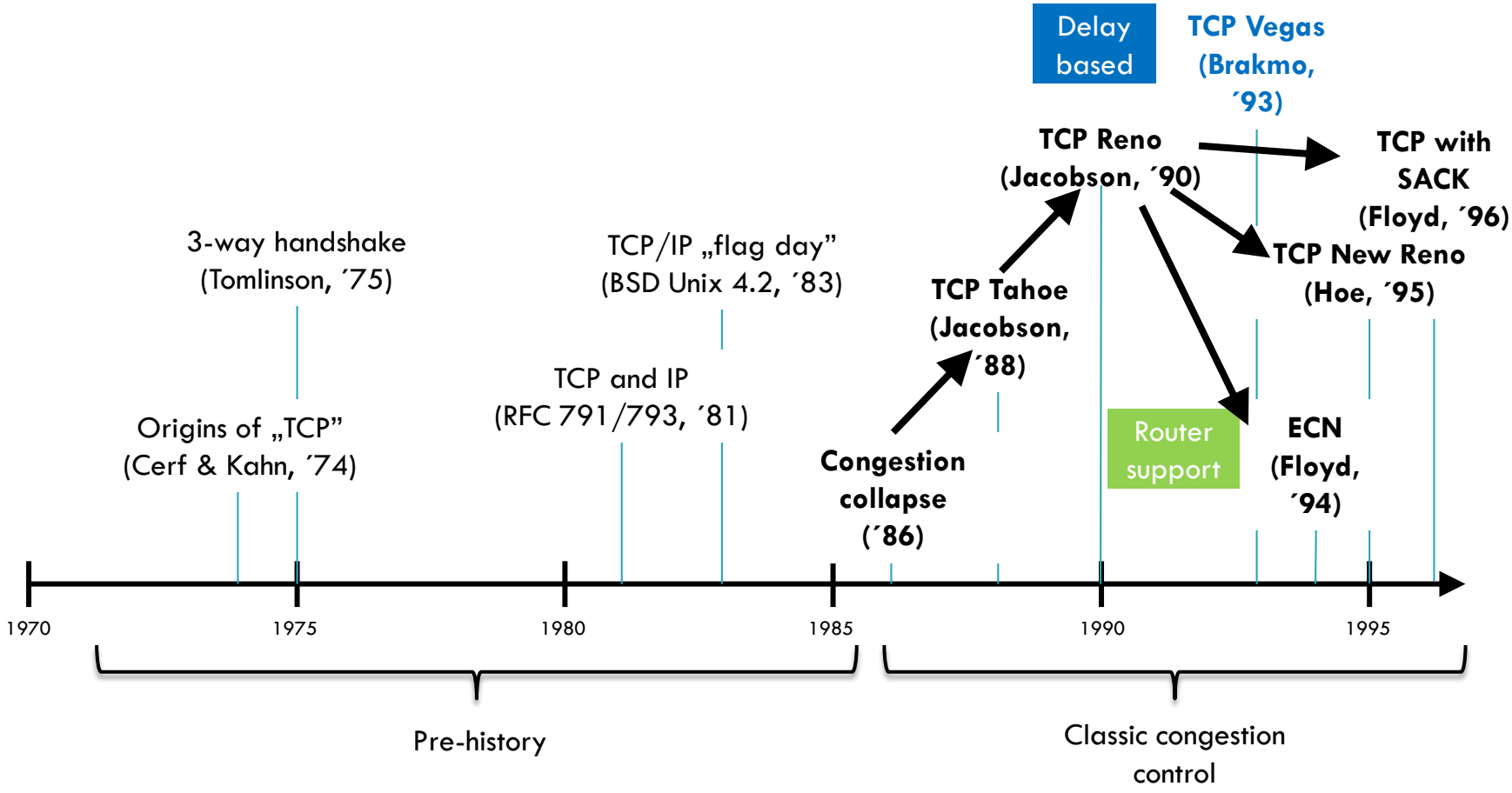
Wireless hálózatok

- Probléma: A Tahoe és Reno esetén csomagvesztés = torlódás
  - WAN esetén ez helyes, ritka bit hibák
  - Azonban hamis vezeték nélküli hálózatokban, gyakori interferenciák
- TCP átvitel ~ $1/\sqrt{vesztési\ ráta}$
  - Már néhány interferencia miatti csomagvesztés elég a teljesítmény drasztikus csökkenéséhez
- Lehetséges megoldások:
  - Réteg modell megsértése, adatkapcsolati információ a TCP-be
  - Késleltetés alapú torlódás vezérlés használata (pl. TCP Vegas)
  - Explicit torlódás jelzés - Explicit congestion notification (ECN)

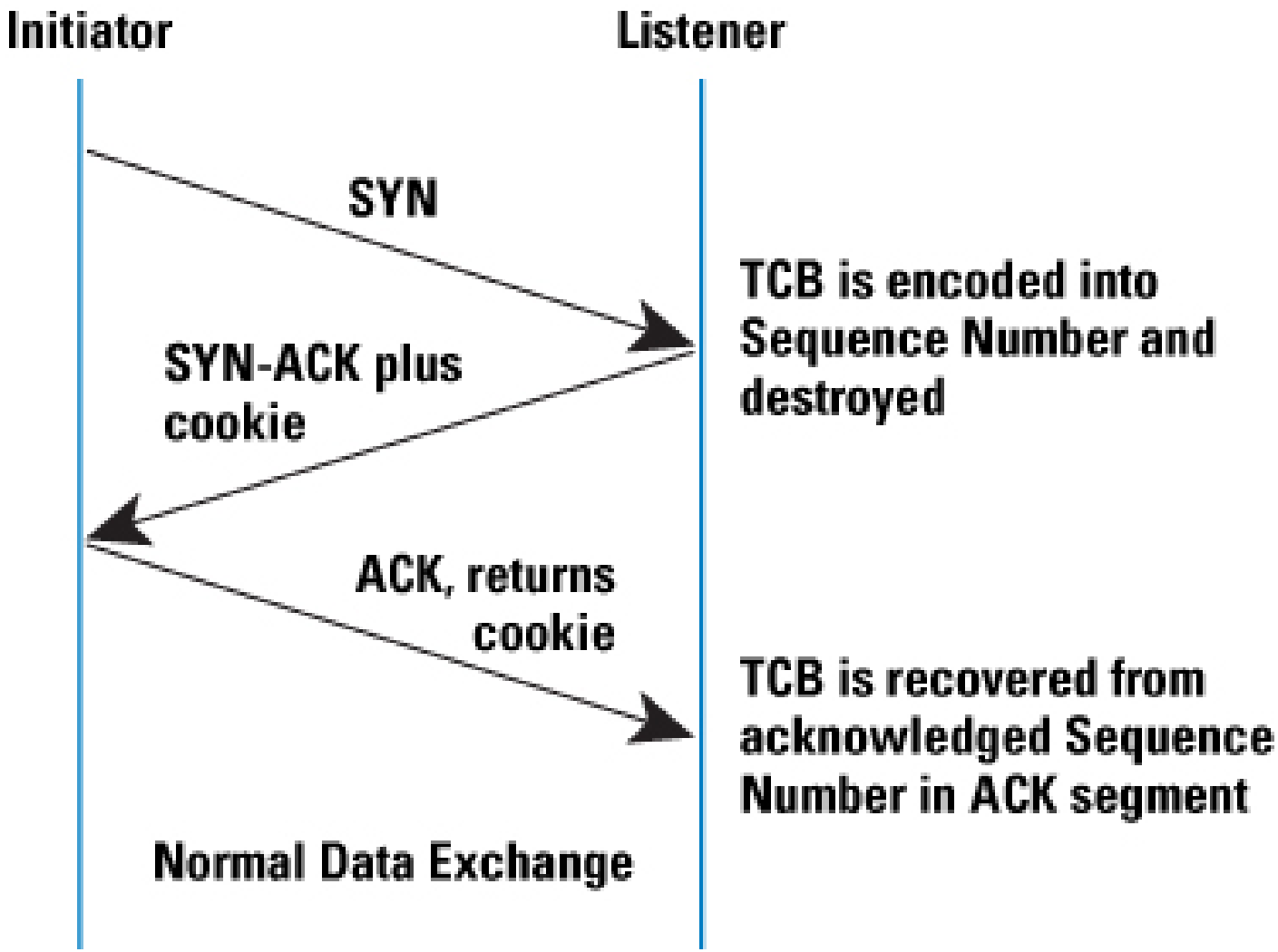
Szolgáltatás megtagadása
Denial of Service (DoS)

- Probléma: a TCP kapcsolatok állapottal rendelkeznek
  - A SYN csomagok erőforrásokat foglalnak az szerveren
  - Az állapot legalább néhány percig fennmarad (RTO)
- SYN flood: elég sok SYN csomag küldése a szervernek ahhoz, hogy elfogyjon a memória és összeomoljon a kernel
- Megoldás: SYN cookie-k
  - Ötlet: ne tároljunk kezdeti állapotot a szerveren
  - Illesszük az állapotot a SYN/ACK csomagokba (a sorszám mezőbe (sequence number mező))
  - A kliensnek vissza kell tükrözni az állapotot…

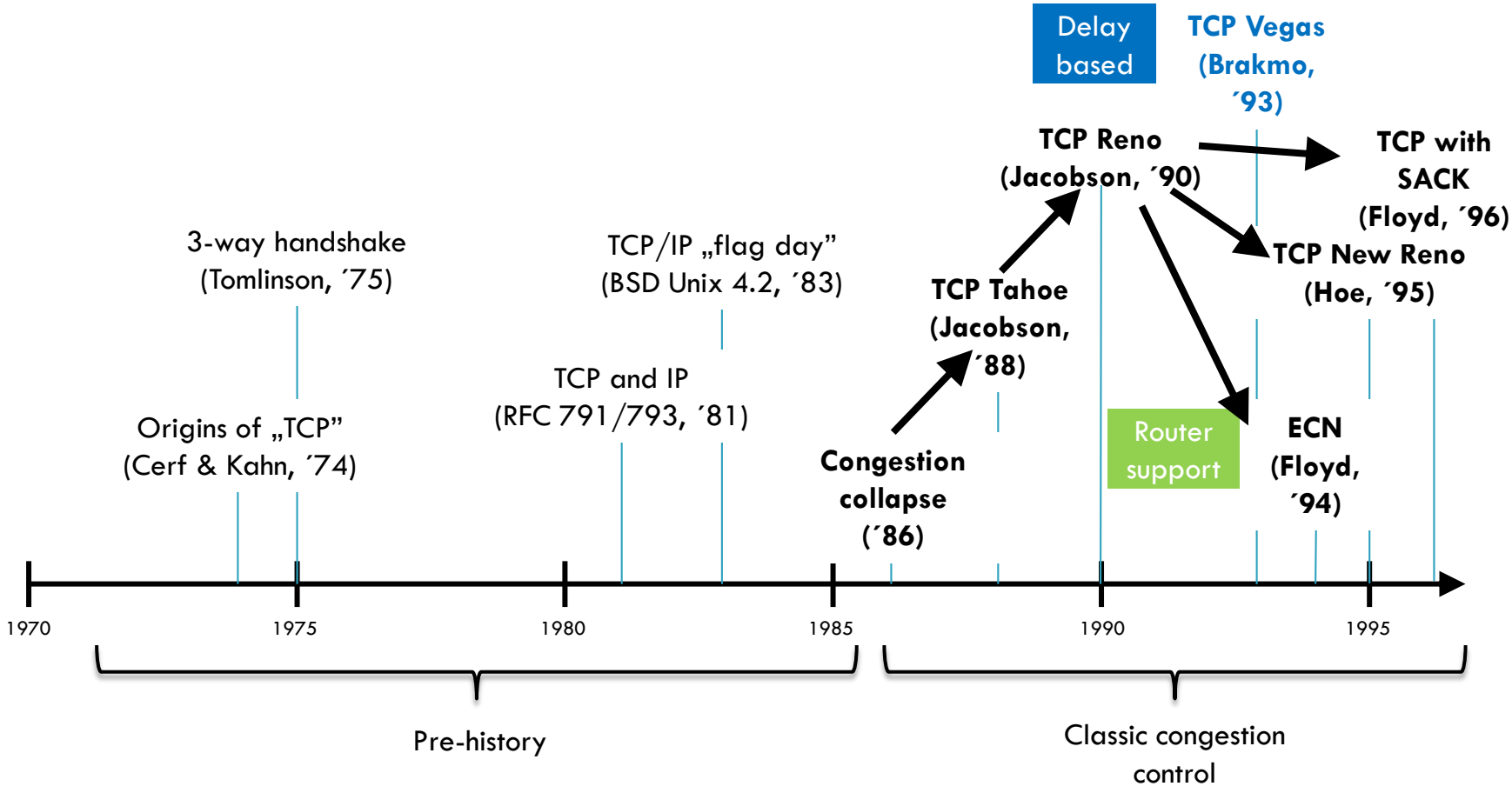
Transport layer evolution



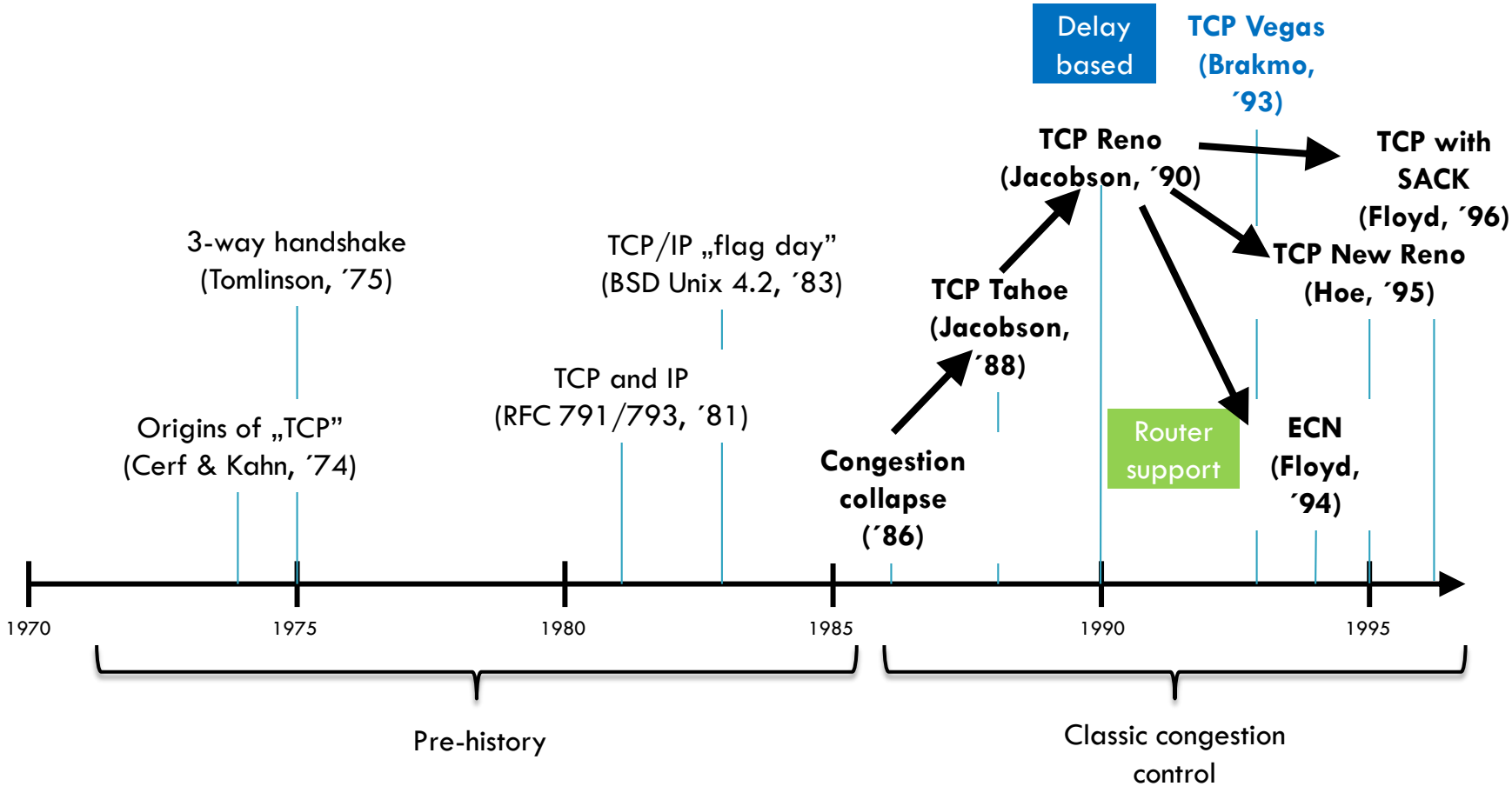
Szolgáltatás megtagadása
Denial of Service (DoS)



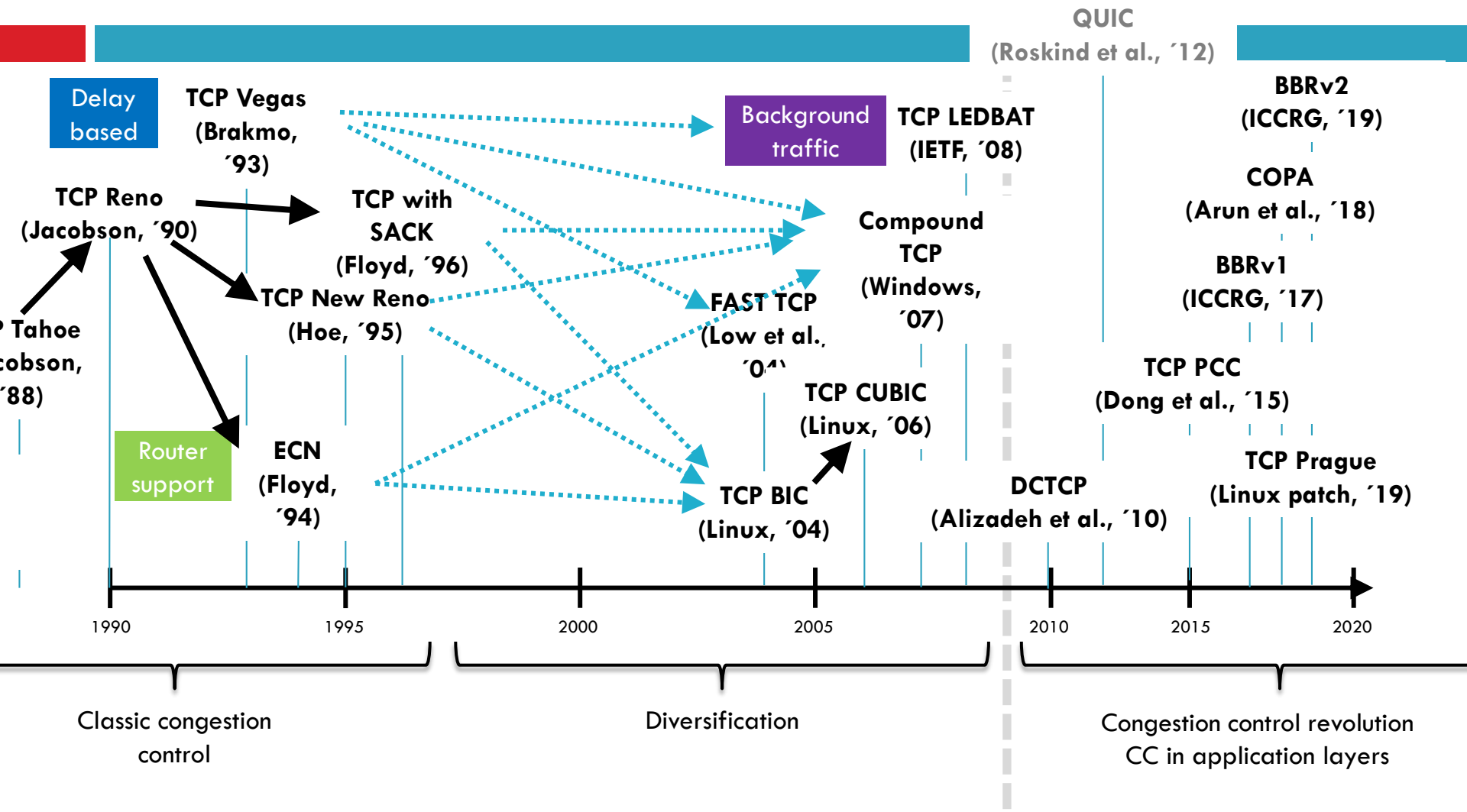
Transport layer evolution



Transport layer evolution



Transport layer (r)evolution



Who will Save the Internet from the Congestion Control Revolution?

Ferenc Fejes, Gergő Gombos and Sándor Laki
ELTE Eötvös Loránd University
Budapest, Hungary

Szilveszter Nádas
Ericsson
Budapest, Hungary
szilveszter.nadas@ericsson.com

ABSTRACT

Active queue management (AQM) techniques have evolved in the recent years, after defining the bufferbloat problem. In parallel novel congestion control (CC) algorithms have been developed to achieve better data transport performance, often assuming simple tail dropping buffers. On the other hand, AQM algorithms usually assume legacy CC (Cubic). Though all of the novel AQM and CC algorithms improve the performance under these assumptions, their co-existence has not or only partially been tested so far. Similarly, router buffer

long buffers full by design, leading to high queueing delay and causing bufferbloat.

The question on how buffers in the routers shall be sized to achieving good utilization with reasonable queuing delay was studied quite in detail in the literature of the 2000s. One of the most comprehensive survey paper [16] from 2009 summarizes existing buffer sizing algorithms, the Bandwidth Delay Product (BPD) rule, the Stanford (or small-buffer model) [1] and the tiny buffer model. Most of the studies assume 1) a Tail Drop buffer in the bottleneck, 2) homogeneous TCP

Kitekintés

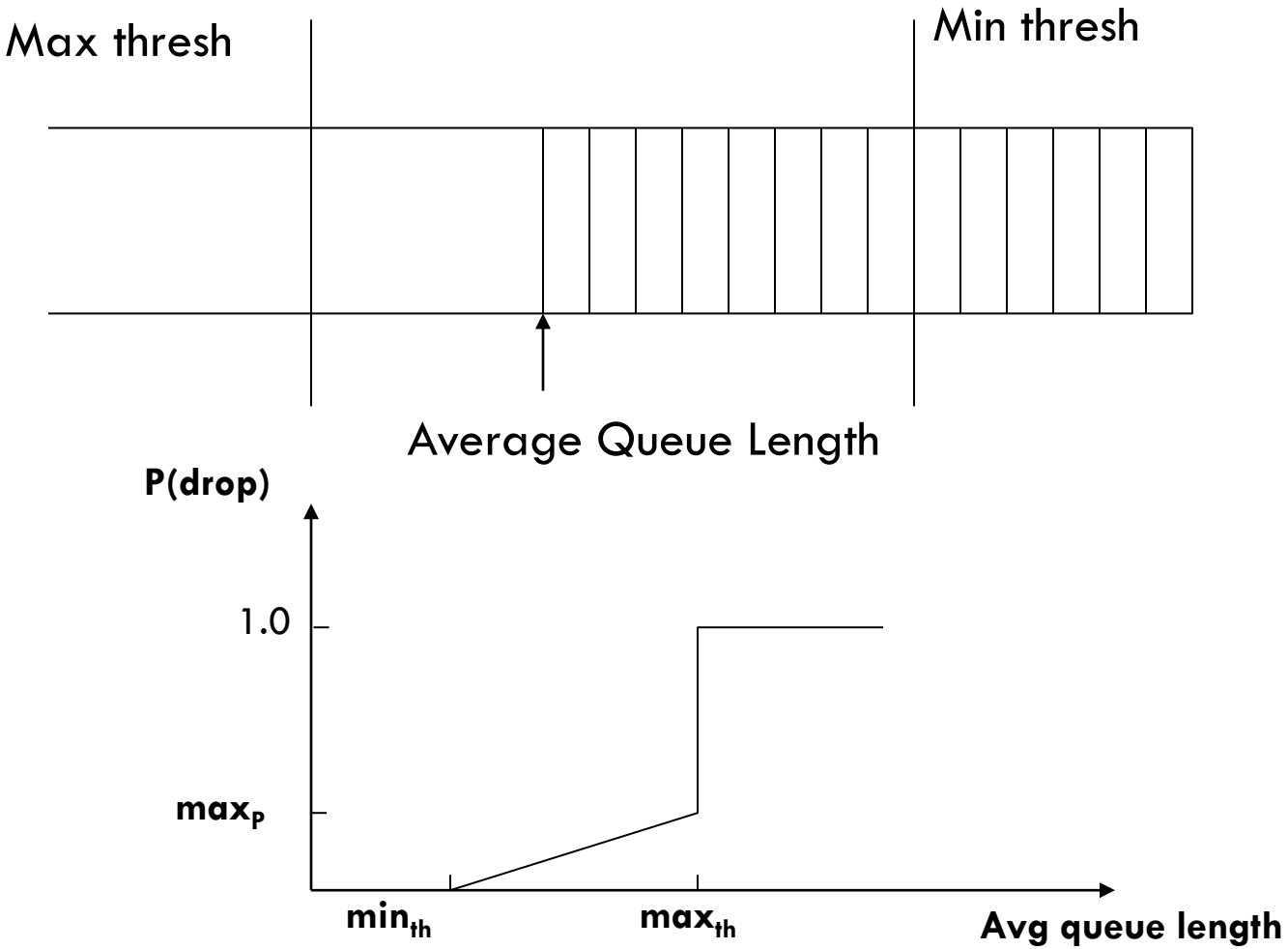
Typical Internet Queuing

- FIFO + drop-tail
  - Simplest choice
  - Used widely in the Internet
- FIFO (first-in-first-out)
  - Implies single class of traffic
- Drop-tail
  - Arriving packets get dropped when queue is full regardless of flow or importance
- Important distinction:
  - FIFO: scheduling discipline
  - Drop-tail: drop policy

RED Algorithm

- Maintain running average of queue length
- If avgq < min_{th} do nothing
  - Low queuing, send packets through
- If avgq > max_{th}, drop packet
  - Protection from misbehaving sources
- Else mark packet in a manner proportional to queue length
  - Notify sources of incipient congestion
  - E.g. by ECN IP field or dropping packets with a given probability

RED Operation

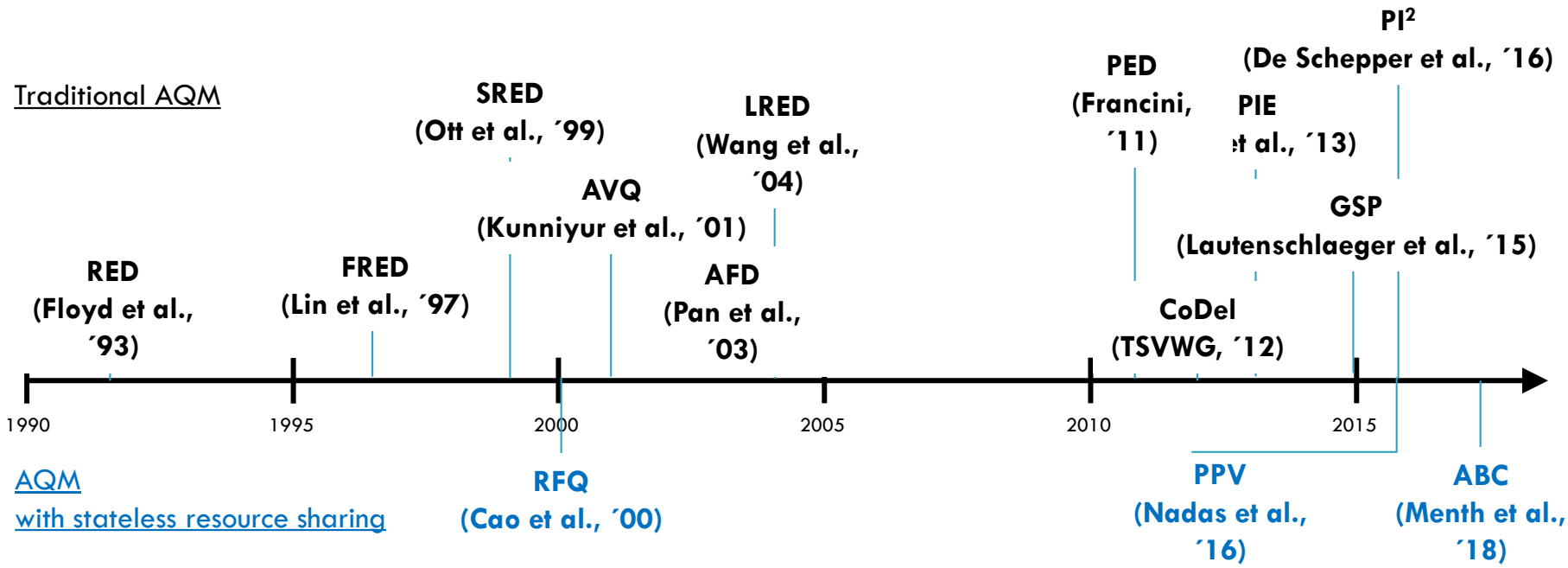


RED Algorithm

- Maintain running average of queue length
- For each packet arrival
  - Calculate average queue size (avg)
  - If $min_{th} \leq avgq < max_{th}$
    - Calculate probability P_a
    - With probability P_a
      - Mark the arriving packet: drop or set-up ECN
    - Else if $max_{th} \leq avg$
      - Mark the arriving packet: drop, ECN

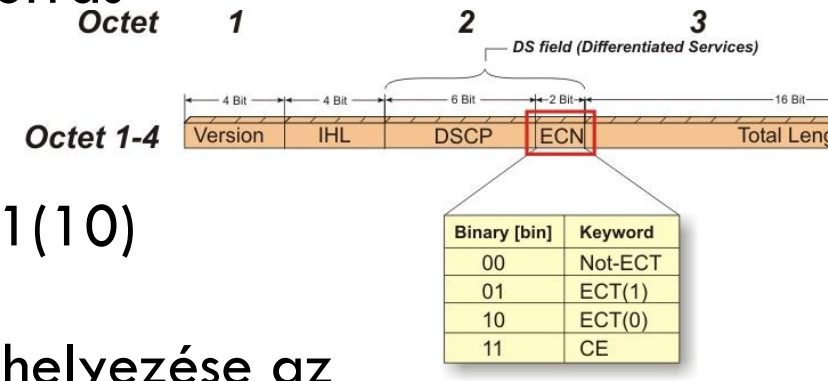
2010s – reducing queuing delay

- Active Queue Management (AQM)
  - Goal is to reduce the average queuing delay, but allow temporal overshoots
  - Proactively starts dropping or marking packets to reduce queuing delay



Csomag dobás vagy ECN jelölés

- Csomag dobás
  - Újraküldés szükséges
  - Egyszerűbb megvalósítás
  - Timout lejárta után tud reagálni a forrás
- ECN jelölés
  - Végpont támogatás szükséges
  - Az IP csomag ECT-0 (01) vagy ECT-1(10) jelöléssel
  - Dobás helyett -> ECN CE (11) jel elhelyezése az IP fejlécben
  - A fogadó érzékeli a CE jelet, majd a visszamenő TCP nyugtába bebillent egy ECE flaget, mely jelzi a forrásnak a torlódást
  - Hagyományos TCP (CUBIC, RENO, stb.) források az ECE flaget csomagvesztésnek értelmezik, de újraküldés nem szükséges.



Binary [bin]	Keyword
00	Not-ECT
01	ECT(1)
10	ECT(0)
11	CE

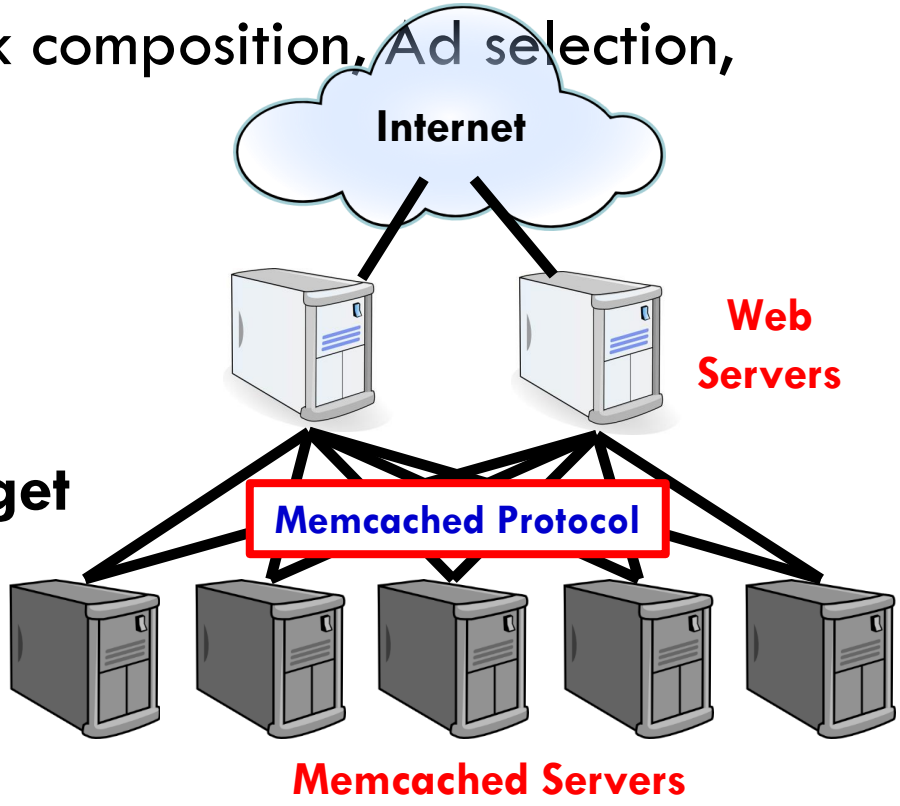
Data Center TCP: DCTCP

Generality of Partition/Aggregate

- The foundation for many large-scale web applications.
  - Web search, Social network composition, Ad selection, etc.
- Example: **Facebook**

Partition/Aggregate ~ Multiget

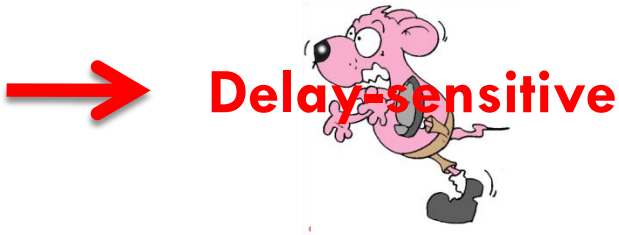
- Aggregators: **Web Servers**
- Workers: **Memcached Servers**



Workloads

- Partition/Aggregate

  (Query)



- Short messages [50KB-1MB]

  (Coordination, Control state)

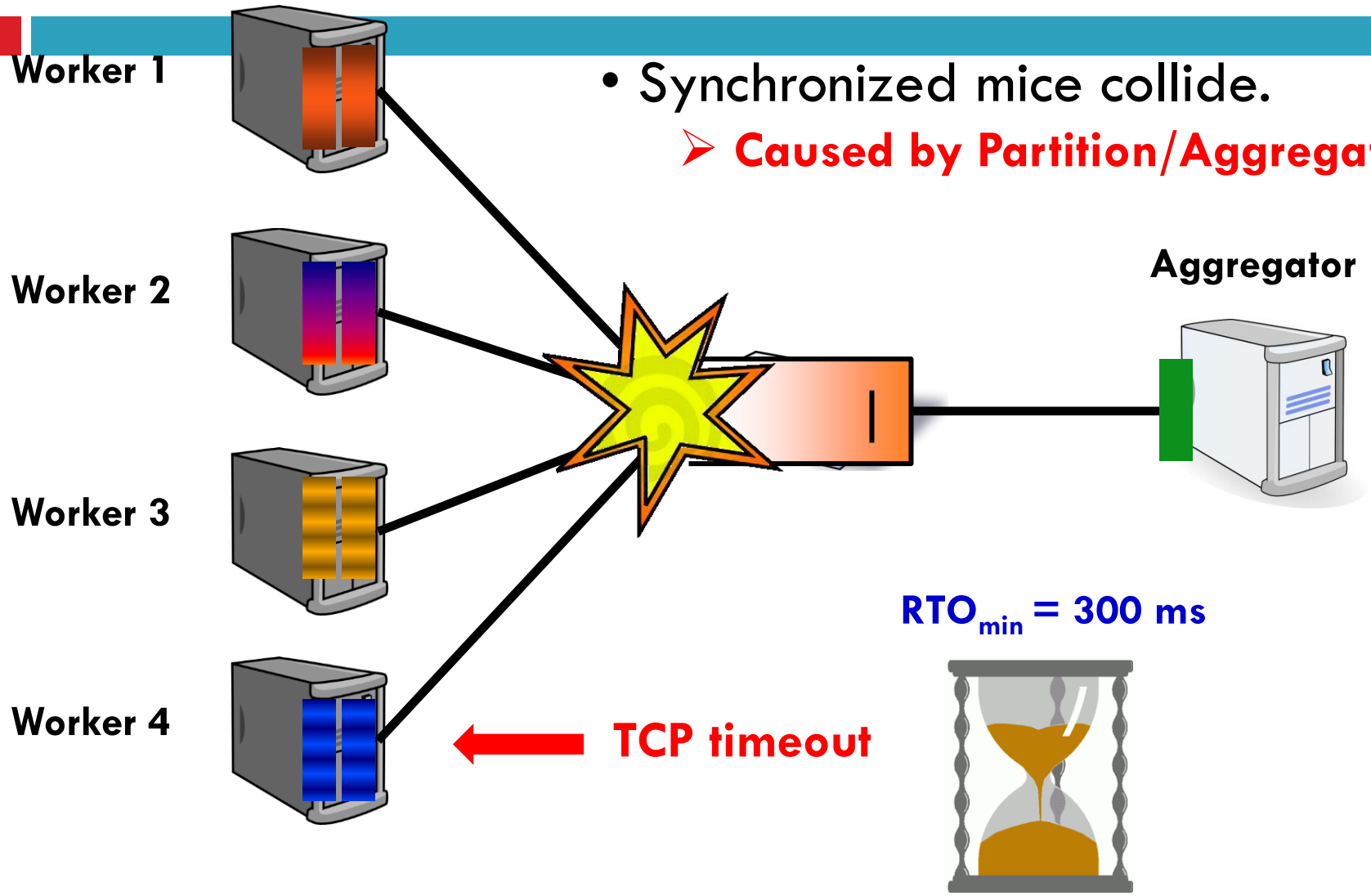
- Large flows [1MB-50MB]

  (Data update)

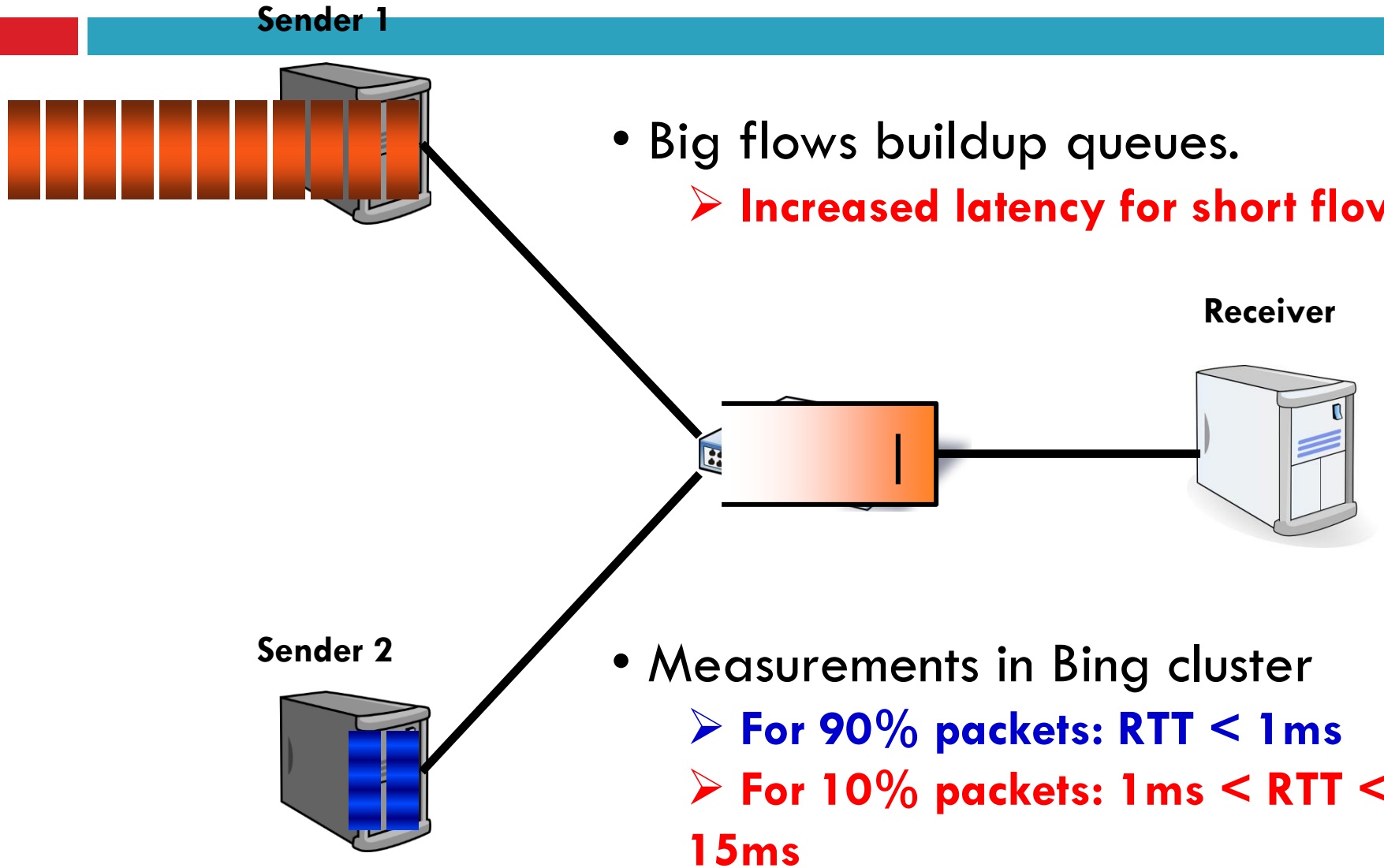
Impairments

- Incast
- Queue Buildup
- Buffer Pressure

Incast


Worker 1
Worker 2
Worker 3
Worker 4
• Synchronized mice collide.
➢ Caused by Partition/Aggregate.
Aggregator
RTO min = 300 ms
TCP timeout

Queue Buildup



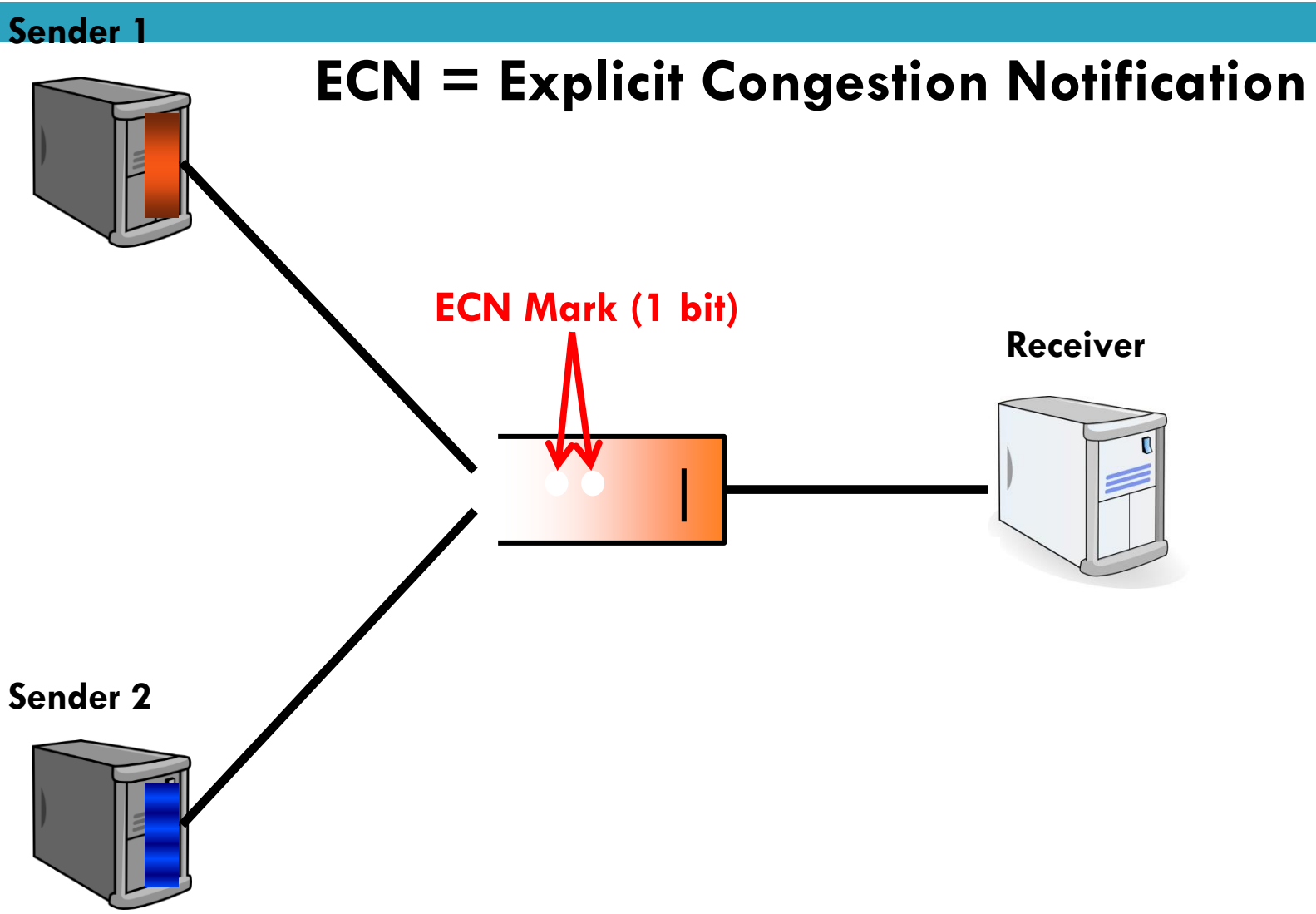
- Big flows buildup queues.
  - **Increased latency for short flows.**
- Measurements in Bing cluster
  - **For 90% packets: RTT < 1ms**
  - **For 10% packets: 1ms < RTT < 15ms**

Data Center Transport Requirements

1. **High Burst Tolerance**
   - Incast due to Partition/Aggregate is common.
2. **Low Latency**
   - Short flows, queries
3. **High Throughput**
   - Continuous data updates, large file transfers

The challenge is to achieve these three together.

DCTCP: The TCP/ECN Control Loop



DCTCP: Two Key Ideas

1. React in proportion to the **extent** of congestion, not its **presence**.
   - ✓ Reduces **variance** in sending rates, lowering queuing requirements.

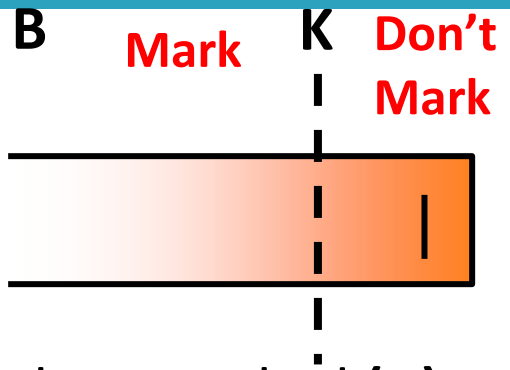
ECN Marks	TCP	DCTCP
1 0 1 1 1 1 0 1 1 1	Cut window by 50%	Cut window by 40%
0 0 0 0 0 0 0 0 0 1	Cut window by 50%	Cut window by 5%

2. Mark based on **instantaneous** queue length.
   - ✓ Fast feedback to better deal with bursts.

Data Center TCP Algorithm

Switch side:

- Mark packets when **Queue Length > K.**



Sender side:

– Maintain running average of ***fraction*** of packets marked **(α)**.

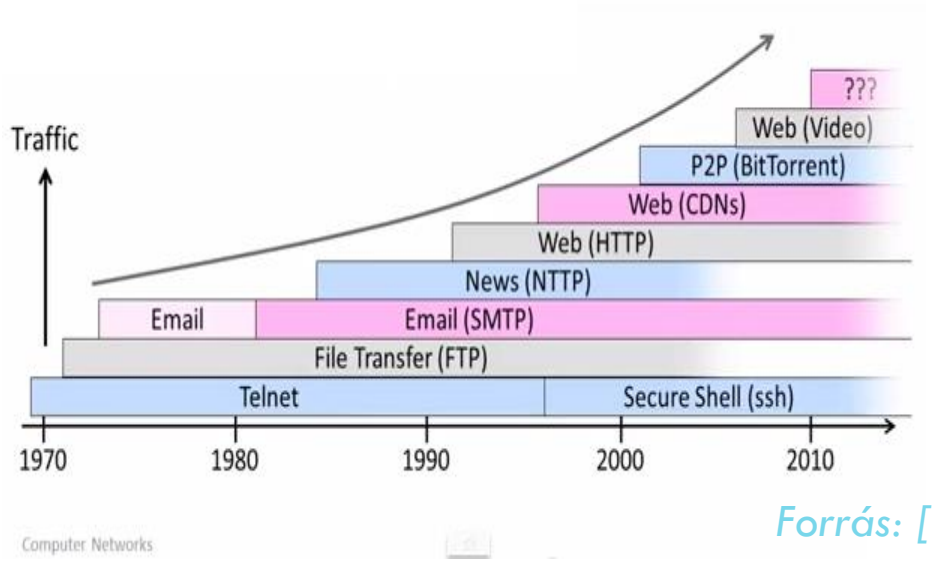
In each RTT:

$$F = \frac{\#\ of\ marked\ ACKs}{Total\ \#\ of\ ACKs} \qquad \alpha \leftarrow (1-g)\alpha + gF$$

➢ **Adaptive window decreases:** $Cwnd \leftarrow (1 - \frac{\alpha}{2})Cwnd$

– Note: decrease factor between 1 and 2.

Internetes alkalmazások evolúciója



Forrás: [1]

- Folyamatosan változik, növekszik…
  - www.evolutionoftheweb.com

DNS

„8. réteg” (A szénalapú csomópontok)

- Ha…
  - Fel szeretnél hívni valakit, akkor el kell kérned a telefonszámát
    - Nem hívhatod csak úgy “P I S T Á T”
  - Levelet küldenél valakinek, akkor szükséged van a címére
- Mi a helyzet az Internettel?
  - Ha el akarod érni a Google-t, szükséges annak IP címe
  - Tudja valaki a Google IP címét???
- A probléma bennünk van:
  - Az emberek nem képesek IP címek megjegyzésére
  - Ember számára értelmes nevek kellenek, melyek IP címekre képezhetők

Internetes nevek és címek

- Címek, pl. 129.10.117.100
  - Számítógépek által használt címkék a gépek azonosítására
  - A hálózat szerkezetét tükrözi
- Nevek, pl. www.northeastern.edu
  - Ember számára értelmes címkék a gépeknek
  - A szervezeti struktúrát tükrözi
- Hogyan képezzünk az egyikről a másikra?
  - Domain Name System (DNS)

Réges régen…

- A DNS előtt minden név-IP leképezés egy *hosts.txt*-ben volt
  - */etc/hosts* - Linuxon
  - C:*Windows**System32**drivers**etc**hosts* - Windowson
- Központosított, manuális rendszer
  - A változásokat emailben kellett beküldeni a SRI-nek
    - SRI=Stanford Research Institute
  - A gépek periodikus időközönként letöltötték (FTP) a *hosts.txt* fájlt
  - Minden név megengedett volt – nem volt benne hierarchia („flat” (sík) felépítés)
    - alans_server_at_sbu_pwns_joo_lol_kthxbye

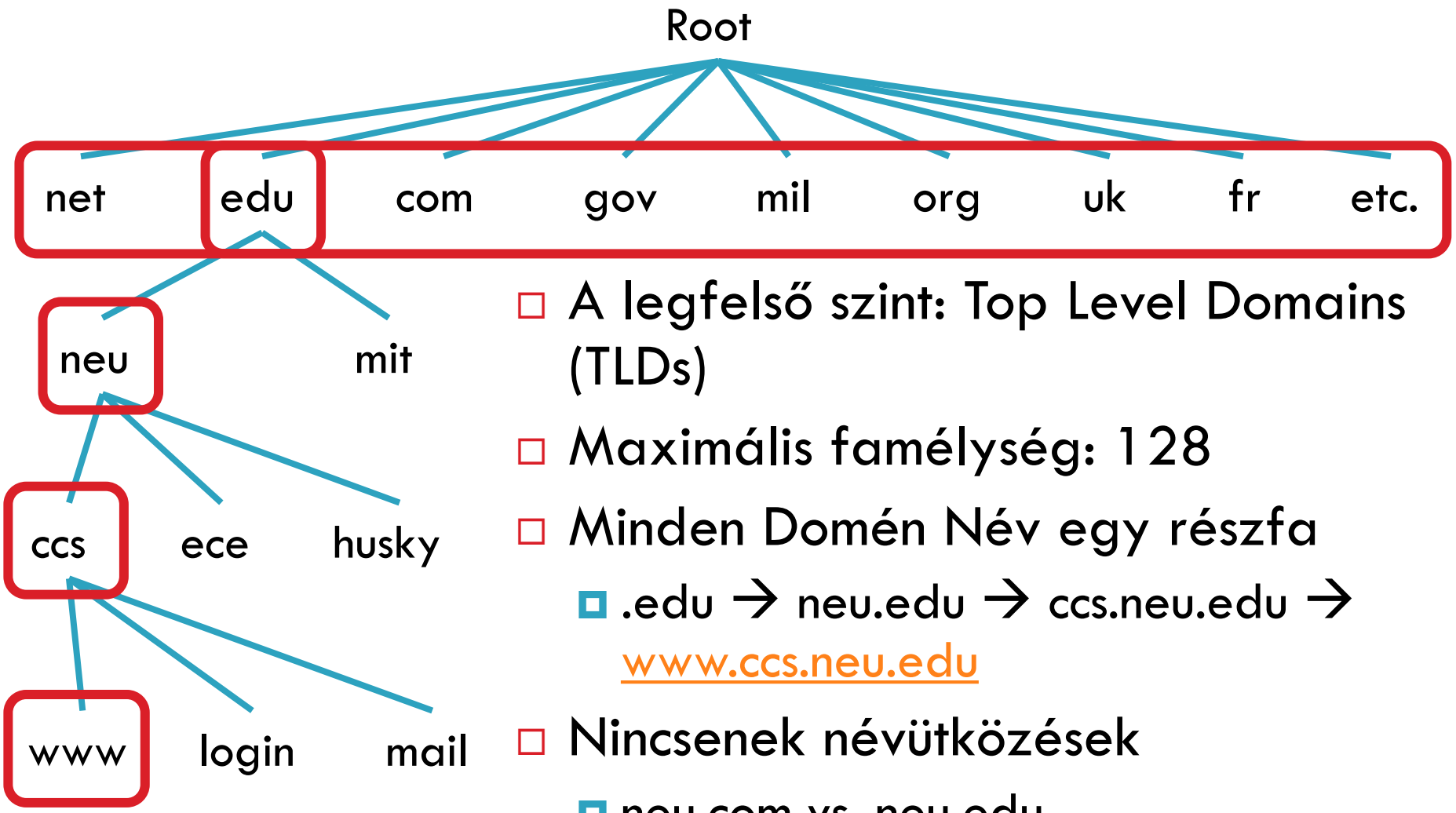
A DNS felé

- Végül a *hosts.txt* alapú rendszer szétesett
  - Nem skálázható, SRI nem bírt a terheléssel/igényekkel
  - Nehéz volt a nevek egyediségének biztosítása
    - Pl. MIT
      - Massachusetts Institute of Technology?
      - Melbourne Institute of Technology?
  - Számos gép rendelkezett nem naprakész *hosts.txt*-vel
- Ez vezetett a DNS megszületéséhez…

DNS általánosságban

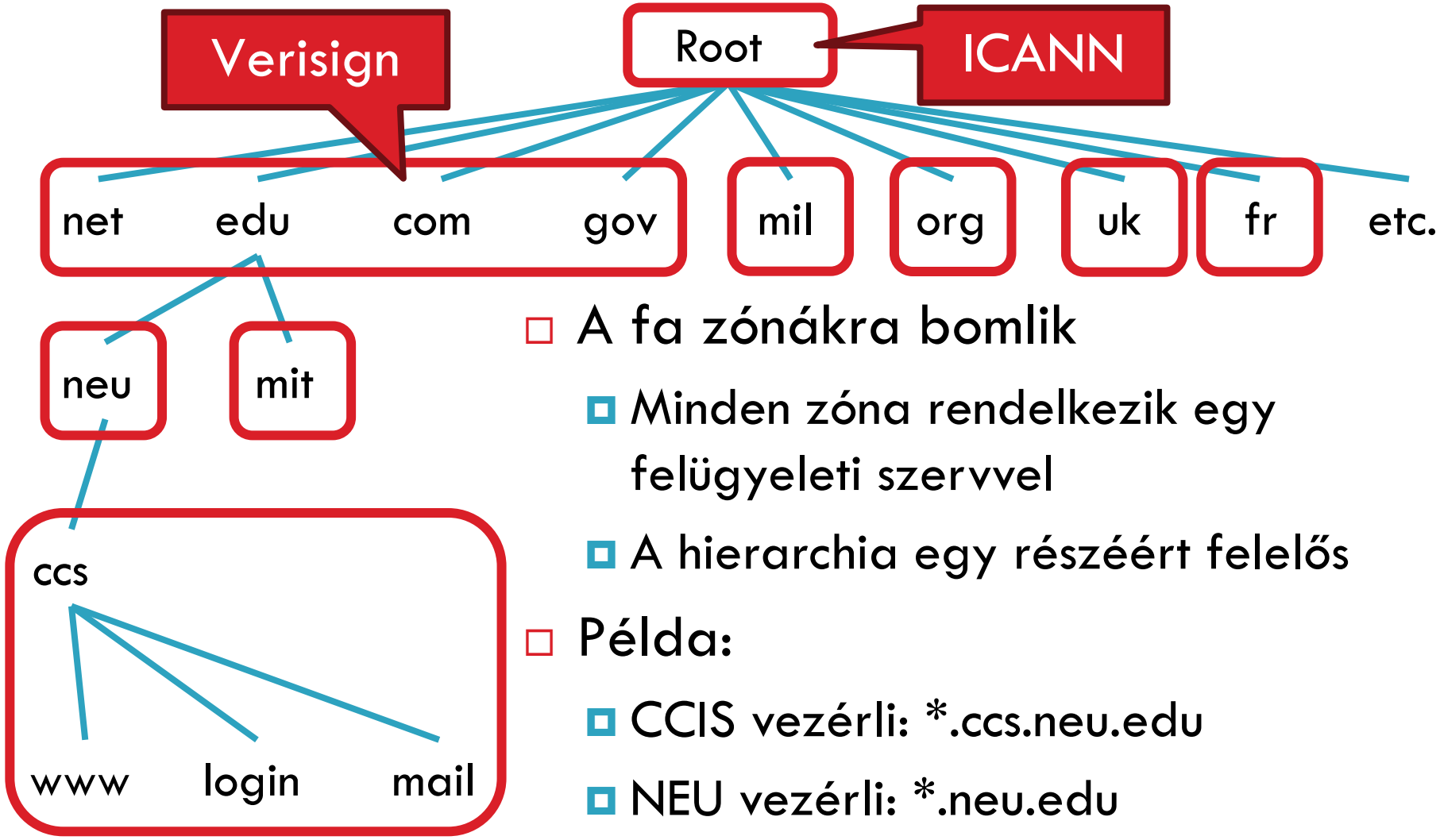
- Domain Name System
- Elosztott adatbázis
  - Nem központosított
- Egyszerű kliens-szerver architektúra
  - UDP 53-as port, vannak TCP implementációk is
  - Rövid kérések – rövid válaszok; kérés-válasz típusú kommunikáció
- Hierarchikus névtér
  - Szemben a hosts.txt alapú flat megoldással
  - pl. .com → google.com → mail.google.com

Név hierarchia



- A legfelső szint: Top Level Domains (TLDs)
- Maximális famélység: 128
- Minden Domén Név egy részfa
  - .edu → neu.edu → ccs.neu.edu → www.ccs.neu.edu
- Nincsenek névütközések
  - neu.com vs. neu.edu

Hierarchikus adminisztráció



- A fa zónákra bomlik
  - Minden zóna rendelkezik egy felügyeleti szervvel
  - A hierarchia egy részéért felelős
- Példa:
  - CCIS vezérli: *.ccs.neu.edu
  - NEU vezérli: *.neu.edu

Szerver hierarchia

- Egy DNS szerver funkciói:
  - A hierarchia egy részét felügyeli
    - Nem szükséges minden DNS nevet tárolnia
  - A zónájához tartozó összes hoszt és domén rekordjainak tárolása
    - Másolatok lehetnek a robosztusság növelés végett
  - Ismeri a root szerverek címét
    - Ismeretlen nevek feloldása miatt kell
- A root szerverek minden TLD-t ismernek
  - Azaz innen indulva fel lehet tárni…

Top Level Domains

- **I**nternet **C**orp. **A**ssigned **N**ames and **N**umbers (1998)
- 22+ **általános *TLDs*** létezik
  - klasszikusok: *.com*, *.edu*, *.gov*, *.mil*, *.org*, *.net*
  - később keletkeztek: *.aero*, *.museum*, *.xxx*
- ~250 TLDs a különböző **ország kódok**nak
  - Két betű (mint például .au, .hu), 2010-től plusz nemzetközi karakterek (például kínai)
  - Több elüzletisedett, például a .tv (Tuvalu)
  - Példa domén hack-ekre: instagr.am (Örményország), goo.gl (Grönland)

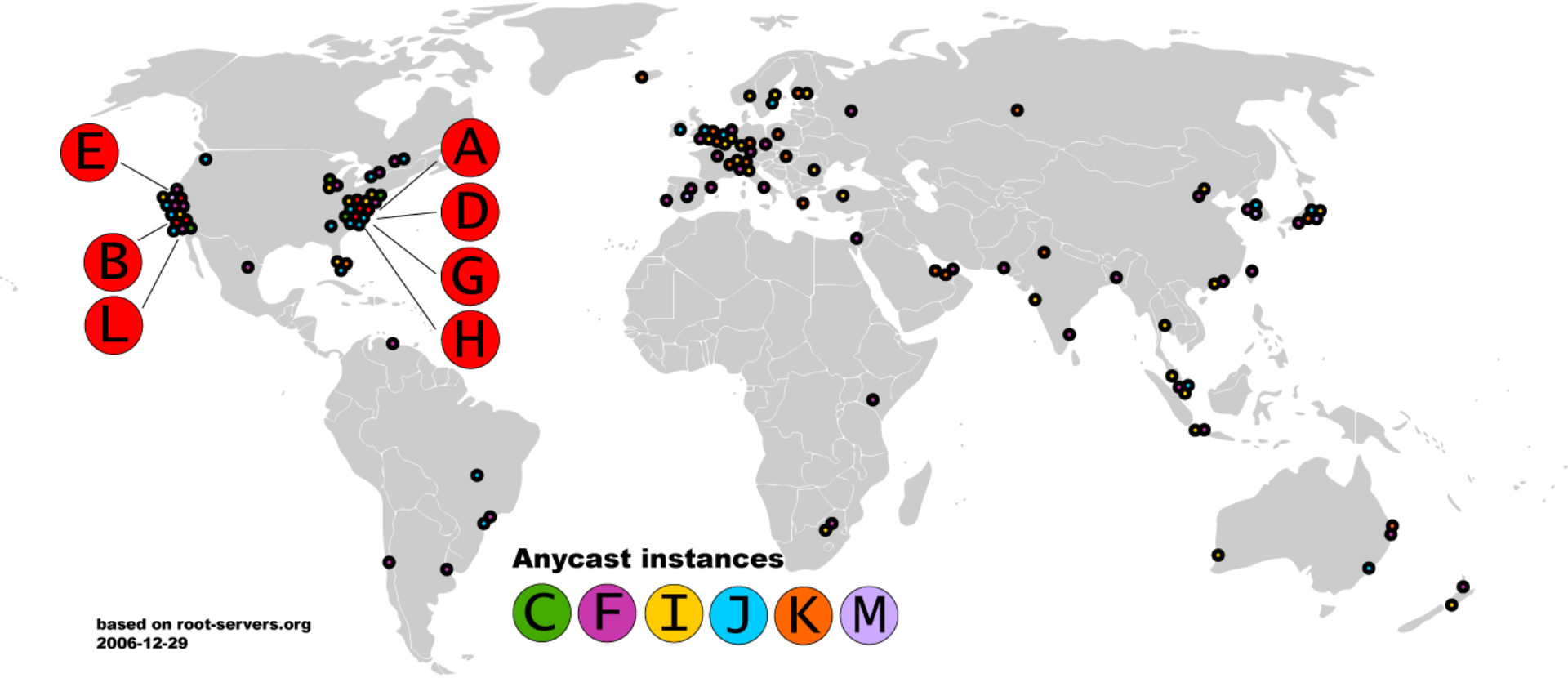
Root Name Servers

- A Root Zone Fájlért felelős
  - Listát vezet a TLD-kről és arról, hogy ki felügyeli őket.
  - ~272KB a fájl mérete
  - Pl. bejegyzése:

com.	172800	IN	NS	a.gtld-servers.net.
com.	172800	IN	NS	b.gtld-servers.net.
com.	172800	IN	NS	c.gtld-servers.net.

- Az ICANN adminisztrálja
  - 13 root szerver, címkék: A→M
  - Pl.: i.root-servers.net
  - 6 db ezek közül „anycastolt”, azaz globálisan számos replika létezik
- Ha név nem feloldható (lokálisan), akkor hozzájuk kell fordulni
  - A gyakorlatban a legtöbb rendszer lokálisan tárolja ezt az információt (cache)

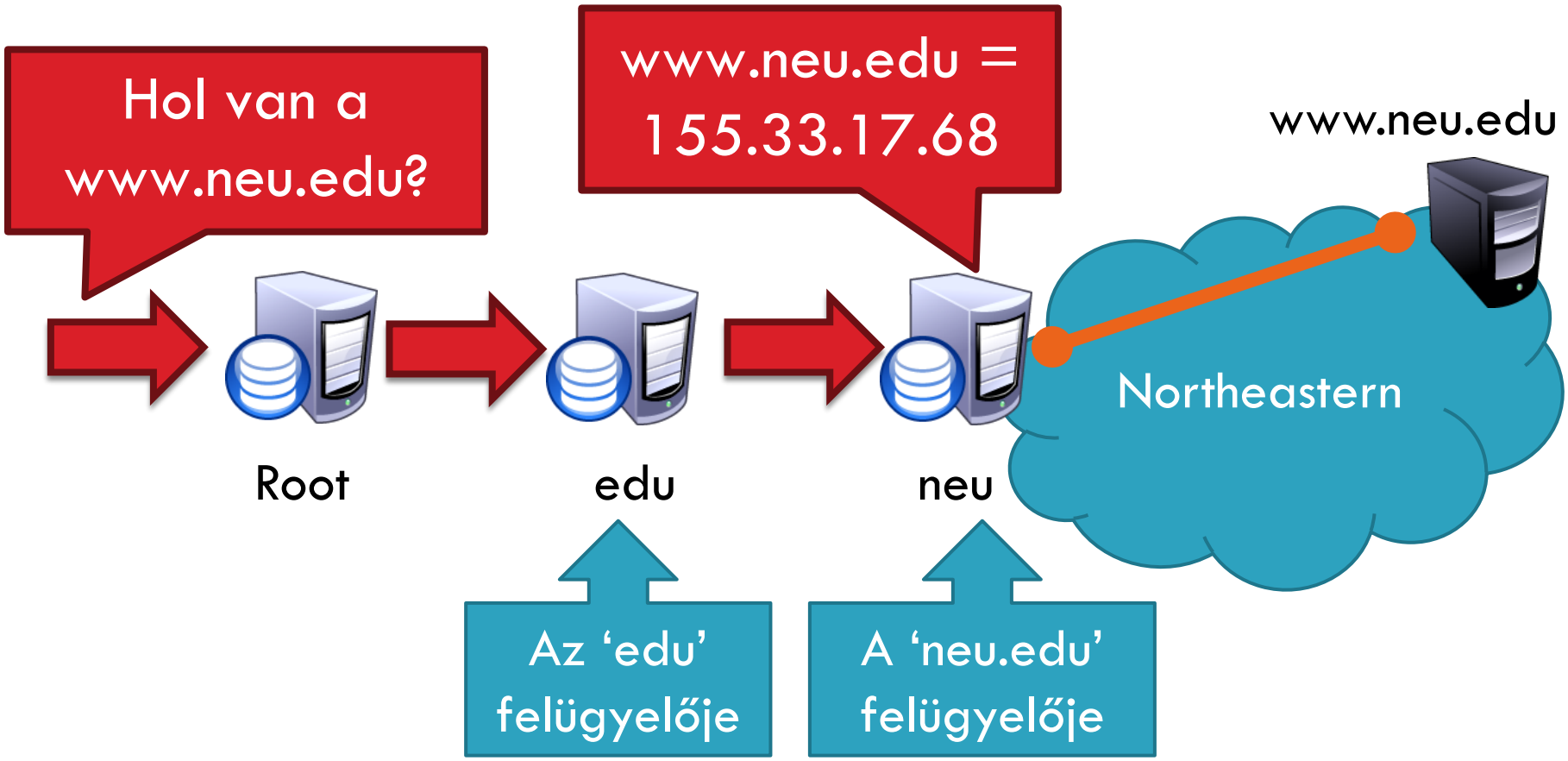
Map of the Roots



Lokális névszerverek

- Minden ISP/cég rendelkezik egy lokális, default névszerverrel
- Gyakran a DHCP konfigurálja fel
- A DNS lekérdezések a lokális névszervernél kezdődnek
- Gyakran cache-be teszik a lekérdezés eredményét

Authoratív Névszerverek



- név→IP leképezéseket tárolja egy adott hoszthoz

Egyszerű doménnév feloldás

- Minden hoszt ismer egy lokális DNS szervert
  - Minden kérést ennek küld
- Ha a lokális DNS szerver tud válaszolni, akkor kész…
  1. A lokális szerver a felügyelő szerver az adott névhez
  2. A lokális szerver cache-ében van rekord a keresett névhez
- Különben menjünk végig a teljes hierarchián felülről lefelé egészen a keresett név felügyeleti szerveréig
  - Minden lokális DNS szerver ismeri a root szervereket
  - Cache tartalma alapján bizonyos lépések átugrása, ha lehet
    - Pl. ha a root fájl tárolva van a cache-ben, akkor egyből ugorhatunk az „.edu” szerverére.

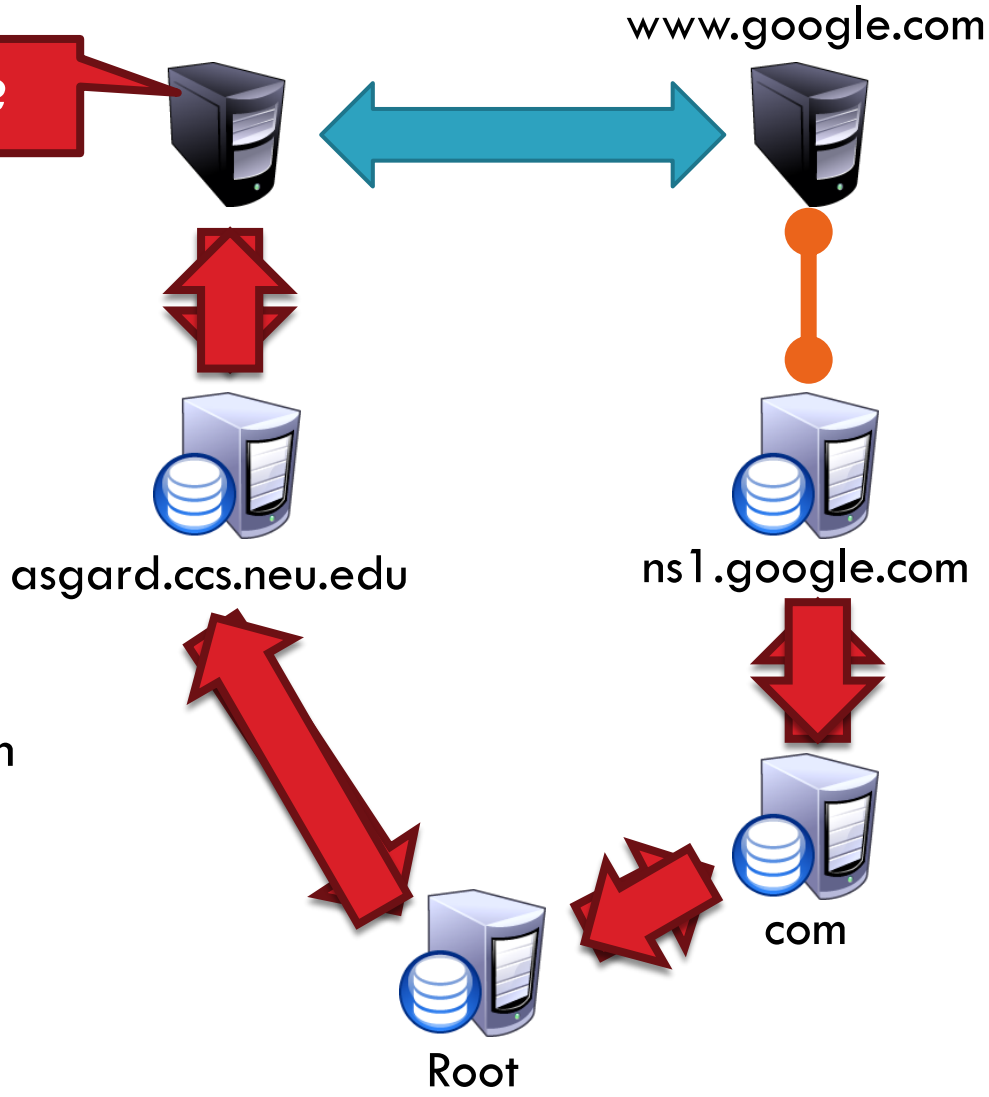
Lekérdezések

- A lekérdezésnek két fajtája van:
  - *Rekurzív lekérdezés* – Ha a névszerver végzi el a névfeloldást, és tér vissza a válasszal.
  - *Iteratív lekérdezés* – Ha a névszerver adja vissza a választ vagy legalább azt, hogy kitől kapható meg a következő válasz.
- Melyik a jobb?
  - *Rekurzív jellemzői*
    - Lehetővé teszi a szervernek a kliens terhelés kihelyezését a kezelhetőségért.
    - Lehetővé teszi a szervernek, hogy a kliensek egy csoportja felett végezzen *cache*lést, a jobb teljesítményért.
  - *Iteratív jellemzői*
    - Válasz után nem kell semmit tenni a kéréssel a névszervernek.
    - Könnyű magas terhelésű szervert építeni.

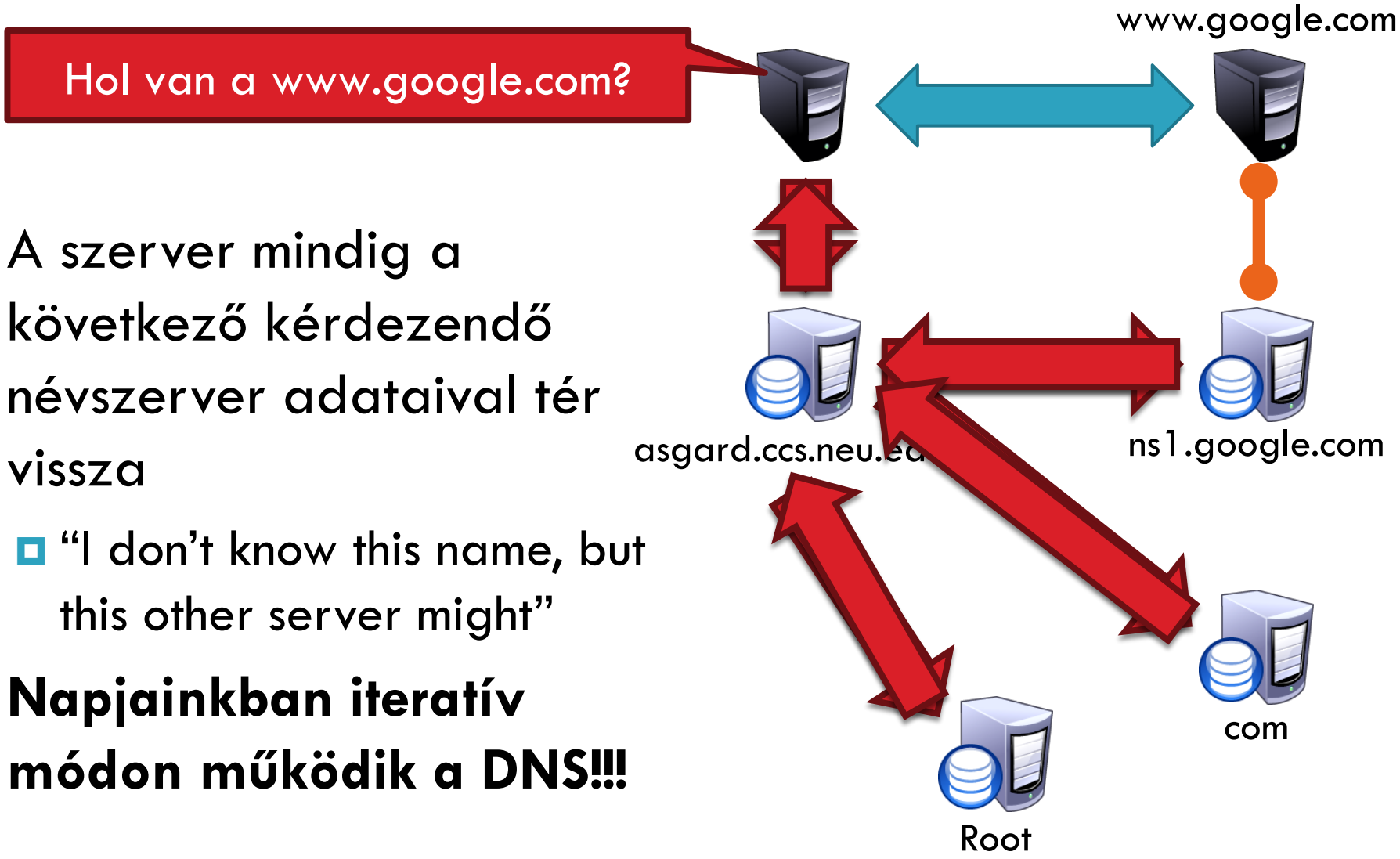
Rekurzív DNS lekérdezés

Hol van a www.google.com?

- A lokális szerver terhet rak a kérdezett névszerverre (pl. root)
- Honnan tudja a kérdezett, hogy kinek továbbítsa a választ?
  - Random ID a DNS lekérdezésben



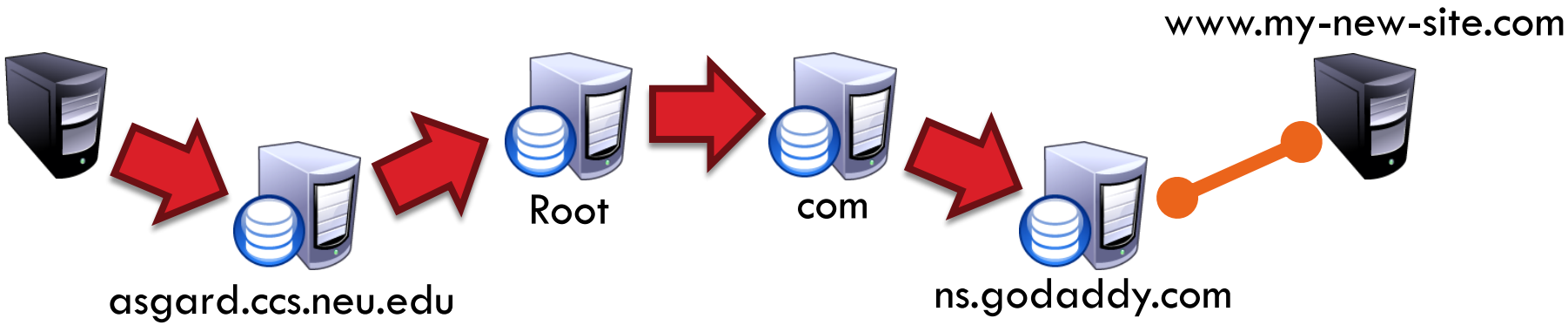
Iteratív DNS lekérdezés



- A szerver mindig a következő kérdezendő névszerver adataival tér vissza
  - "I don't know this name, but this other server might"
- **Napjainkban iteratív módon működik a DNS!!!**

DNS bejegyzés elterjedése

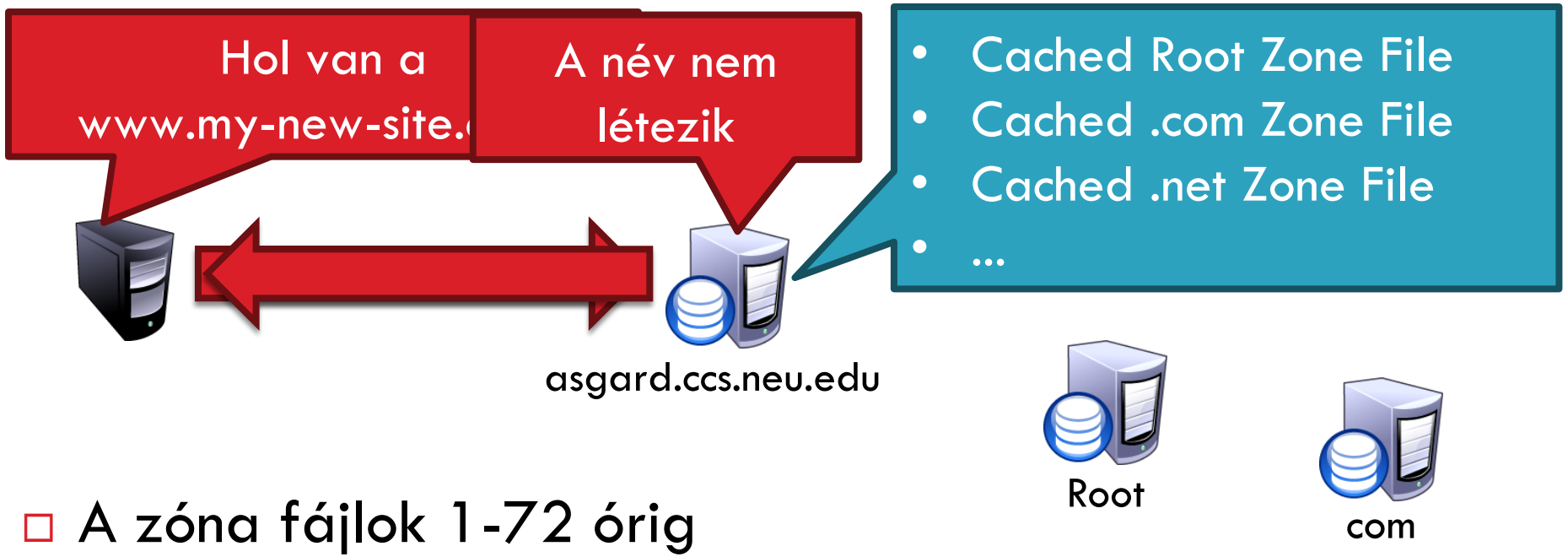
- Van-e a teremben olyan, aki vásárolt már domén nevet?
  - Észrevettétek-e, hogy kb. 72 óra kell ahhoz, hogy elérhető legyen a bejegyzés után?
  - Ez a késés a DNS Propagáció/DNS bejegyzés elterjedése



- Miért nem sikerül ez egy új DNS név esetén?

Cachelés VS frissesség

- DNS elterjedés késését a cache okozza



- A zóna fájlok 1-72 órig élnek a cache-ben

DNS Erőforrás rekordok (Resource Records)

- A DNS lekérdezéseknek két mezőjük van
  - name és type
- Az erőforrás rekord válasz egy DNS lekérdezésre
  - Négy mezőből áll: (name, value, type, TTL)
  - Egy lekérdezésre adott válaszban több rekord is szerpelhet
- Mit jelent a name és a value mező?
  - Ez a lekérdezés típusától (type) függ

DNS lekérdezés típusok

- Type = A / AAAA
  - Name = domén név
  - Value = IP cím
  - A = IPv4, AAAA = IPv6

Query
Name: www.ccs.neu.edu
Type: A

Resp.
Name: www.ccs.neu.edu
Value: 129.10.116.81

- Type = NS
  - Name = rész domén
  - Value = a rész doménhez tartozó DNS szerver neve
  - "Menj és küldd a kérésed ehhez a szerverhez"

Query
Name: ccs.neu.edu
Type: NS

Resp.
Name: ccs.neu.edu
Value: 129.10.116.51

DNS lekérdezés típusok

- Type = CNAME
  - Name = domén név
  - Value = kanonikus név
  - Alias nevek használatához
  - CDN használja

Query
Name: foo.mysite.com
Type: CNAME

Resp.
Name: foo.mysite.com
Value: bar.mysite.com

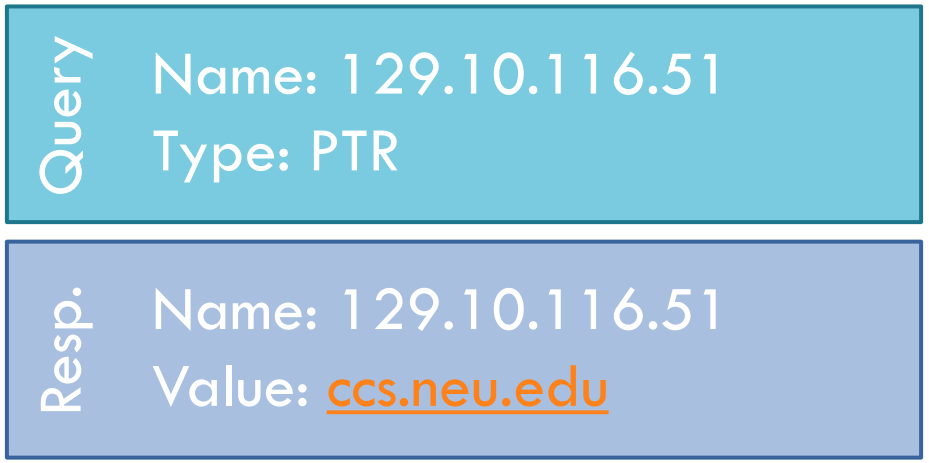
- Type = MX
  - Name = emailben szereplő domén név
  - Value = mail szerver kanonikus neve

Query
Name: ccs.neu.edu
Type: MX

Resp.
Name: ccs.neu.edu
Value: amber.ccs.neu.edu

Fordított lekérdezés (PTR rekord)

- Mi a helyzet az IP→név leképezéssel?
- Külön hierarchia tárolja ezeket a leképezéseket
  - Gyökér pont: in-addr.arpa és ip6.arpa
- DNS rekord típusa (type): PTR
  - Name = IP cím
  - Value = domén név
- Nincs garancia arra, hogy minden IP címre működik



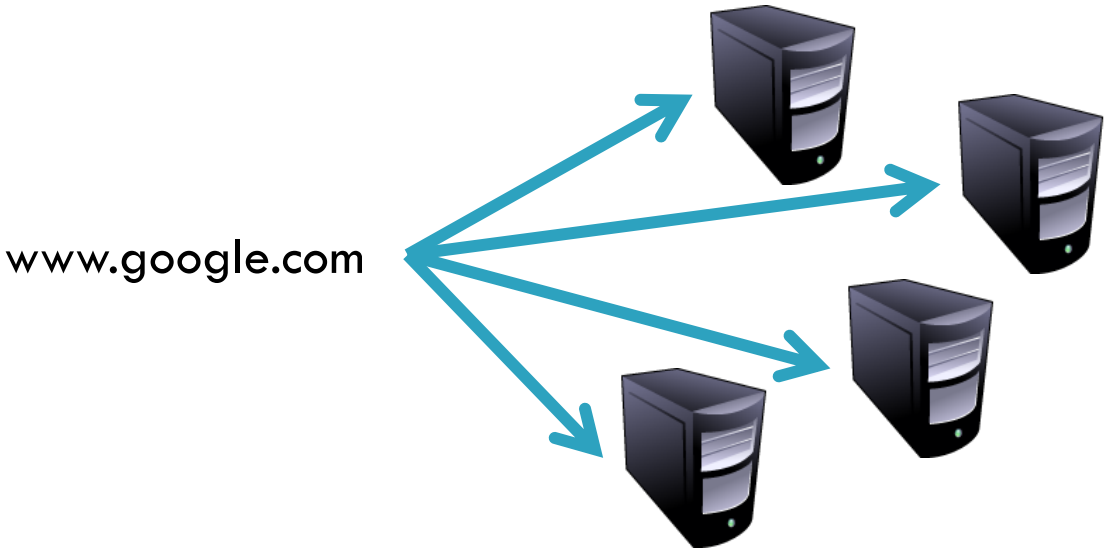
DNS as Indirection Service

- DNS számos lehetőséget biztosít
  - Nem csak a gépekre való hivatkozást könnyíti meg!
- Egy gép IP címének lecserélése is triviális
  - Pl. a web szervert átköltöztetjük egy új hosztra
  - Csak a DNS rekord bejegyzést kell megváltoztatni!

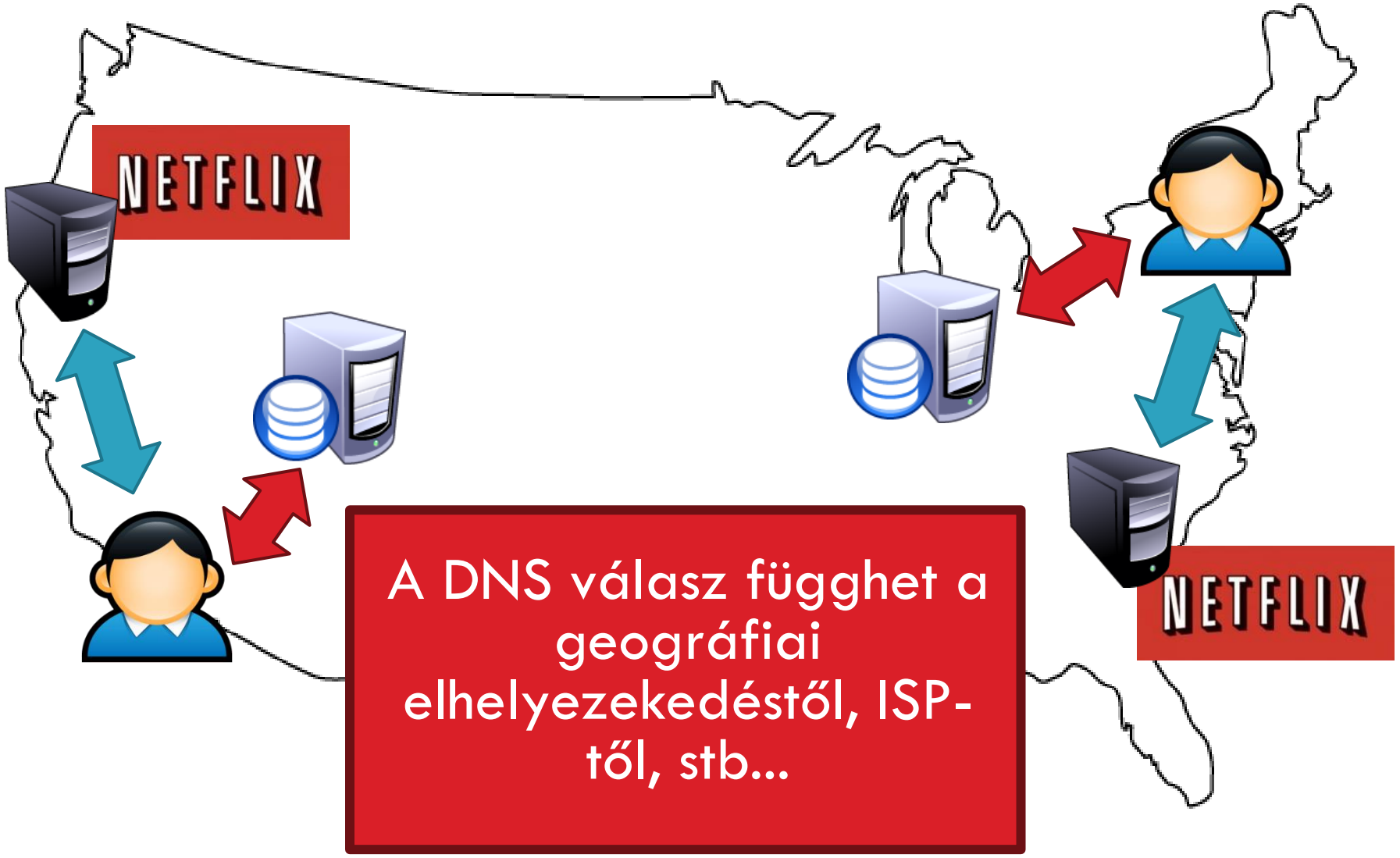
Aliasing/Kanonikus nevek és Load Balancing/Terhelés elosztás

- Egy gépnek számos alias neve lehet

- Egy domén névhez számos IP cím tartozhat



Content Delivery Networks



A DNS fontossága

- DNS nélkül…
  - Hogyan találjunk meg egy weboldalt?
- Példa: a mailszervered azonosít
  - Email címet adunk meg weboldalakra való feliratkozásnál
  - Mi van, ha valaki eltéríti a DNS bejegyzést a mailszerveredhez?
- DNS a bizalom forrása a weben
  - Amikor a felhasználó begépeli a www.bankofamerica.com címet, azt várja, hogy a bankja honlapja jelenjen meg.
  - Mi van, ha a DNS rekordot meghackelték?

Vizsgákról

- Online vizsga = bármilyen segédanyag használható

- Vizsga – Canvas modul, ami két „vizsgarészből” áll
  - Teszt – 20 pont (egyben beugró is)
    - 20 kérdés 20 perc, szekvenciális kitöltés
    - min. 11 pont szükséges
  - Kifejtős rész – 25 pont
    - Sikeres teszt után nyílik meg a Canvas felületen
    - 5 kérdés 30 perc időkeret, tetszőleges sorrendben tölthető
    - 5 pontos kérdések
    - **Megértésre kérdezünk rá és nem a diák lemásolása a cél**

Teszt minta

5. kérdés 1 pont

Mely modulációs technika használja a vivőhullám több jellemzőjét is a szimbólumok kifejezésére?

- Fázis moduláció
- Amplitúdó moduláció
- Frekvencia moduláció
- QAM-16 technika

6. kérdés 1 pont

Két szimbólum használata esetén a szimbólum ráta 4 Baud. Négy szimbólum használata mellett mekkora lesz a szimbólum ráta, ha semmi mást nem változtatunk?

Baud

Teszt minta

5. kérdés 1 pont

Mely modulációs technika használja a vivőhullám több jellemzőjét is a szimbólumok kifejezésére?

- Fázis moduláció
- Amplitúdó moduláció
- Frekvencia moduláció
- QAM-16 technika

6. kérdés 1 pont

Két szimbólum használata esetén a szimbólum ráta 4 Baud. Négy szimbólum használata mellett mekkora lesz a szimbólum ráta, ha semmi mást nem változtatunk?

Baud

Teszt minta

7. kérdés 1 pont

Mely állítások igazak a Hamming-kódra? (3 állítás igaz)

- [ ] Nem használ redundanciát, emiatt nagyon kompakt kódolás.
- [ ] Mindegyik ellenőrző bit a bitek valamilyen csoportjának a paritását állítja be párosra (vagy páratlanra)
- [ ] Paritást használó technika
- [ ] Minden bitet kétszer küldünk át.
- [ ] A generátor polinómot a protokoll definálja
- [ ] 2 egészhatvány sorszámú pozíciói lesznek az ellenőrző bitek, azaz 1,2,4,8,16,..., a maradék helyeket az üzenet bitjeivel töltjük fel
- [ ] A polinóm aritmetika mod 2 felett történik.
- [ ] Polinom osztáson alapuló technika

8. kérdés 1 pont

Egy kód Hamming-távolsága 8. Hány egyszerű bithibát tudunk detektálni ezzel a kóddal?

Teszt minta

14. kérdés 1 pont

Egy távolságvektor routing protokollt használó hálózatban az A állomás routing táblája a következő:

host | költség | next hop
B | 7 | B
C | 10 | C
D | 1 | D
E | 14 | D

B szomszédtól a következő távolságvektort kapja:

C | 2
D | 3
E | 3

Mi lesz C költsége A állomás routing táblájában? [Kiválaszt]

15. kérdés 1 pont

Mely állítások igazak a BGP protokollra?

- A politikai jellegű szabályokat kézzel konfigurálják a BGP-routeren.
- A BGP egy intra-domain routing protokoll
- A BGP alapvetően link state protokoll, viszont a router nyomon követi a használt útvonalat, és az útvonalat mondja meg a szomszédjainak.
- Megadható olyan szabály, hogy ne legyen átmenő forgalom bizonyos AS-eken keresztül.

Kifejtős minta

- Mi a lényege az ISO/OSI rétegmodellnek? Milyen problémát old meg? Milyen rétegeket definiál a modell és ezek közül melyekkel írható le az Internet működése?

- Adatátvitel esetén milyen problémát okozhat a küldő és a fogadó óráinak elcsúszása? Megoldja-e ezt a problémát az RZ kódolás(elején: 1-es bit magas jelszint/ 0-s bit alacsony jelszint, közepén: 1-es bit esetén váltás, végén: semmi)? Milyen kódolást alkalmaznak a 10 Mbps és 100 Mbps Ethernet szabványok? Milyen hátrányai vannak ennek a megoldásnak? Indokolja válaszát!

Kifejtős minta

- Tegyük fel, hogy minden keretbe 1000 bájt hasznos adat fér (Megj.: a keretezés után a keret mérete lehet más, de a legrosszabb esetben is befér 1000 adatbájt a keretbe.). Az előadáson látott karakterszámlálás, bájt beszúrás és bit beszúrás módszerek esetén mekkora teljesítmény csökkenést kapunk a legjobb és a legrosszabb esetben? Indokolja a válaszát példák segítségével!

- A különböző autonóm rendszereken belül inkompatibilis routing protokollok (távolság vektor vs kapcs.állapot alapú) is használhatók. Ennek ellenére a globális forgalomirányítás működik. Hogyan lehetséges ez?

Köszönöm a figyelmet!