

The background features a close-up of a laptop keyboard with a blue color scheme. Overlaid on the keyboard is a glowing globe of the Earth, surrounded by a network of white lines and nodes, symbolizing global connectivity and the internet. The overall aesthetic is futuristic and digital.

# **Az Internet ökoszisztémája és evolúciója**

**Rétvári Gábor, Heszberger Zsolt**

# Tartalom

- Domain Name System alapvető jellemzői
  - DNS feladata
  - Domain nevek szerkezete
  - Architektúra, alapvető működés
- DNS rendszer konfigurációja - BIND
  - Név állományok, szerver konfiguráció
  - DNS rekord típusok
  - Levelezés útvonalválasztás
- Egyéb DNS funkciók
  - Biztonsági kérdések – DNSSec, FCrDNS, DNS cache-poisoning, adathalászat
  - IPV6 módosítás – AAAA
  - Dynamic DNS rendszerek

# DNS feladata



- Az internetes útvonalválasztás IP cím alapján történik, elvileg ezzel minden megoldható az Interneten
- Az emberek számára ugyanakkor a számok nehezen megjegyezhetők, különösen ha IPv6 címekről beszélünk! Betűkből álló szavakra könnyebben emlékszünk
- Nem csak böngészéskor fontos, hanem pl. weboldalakba beágyazva is teszünk címeket: könnyebben ellenőrizhető mint a számok, de azért itt is lehetnek problémák: pl. googl.com, gogle.com, googlo.com stb.

**A DNS általános feladata, hogy emberi olvasásra alkalmas alfanumerikus karaktereket tartalmazó neveket, internetes erőforrásokra fordítson át!**

# Az Internetes erőforrások (CIR) irányítása



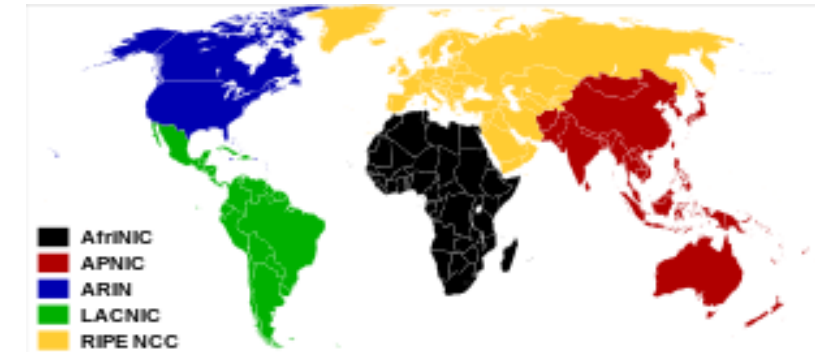
- Internetes erőforrásokon általános értelemben sok mindent érthetünk pl. fizikai infrastruktúra (optikai szál stb.), szűken véve azonban kifejezetten a globális virtuális infrastruktúrákat értjük ez alatt:
  - Internet címek – IP address
    - Egy IP egy gépet azonosít globálisan az Interneten
    - IPv4 – 32 bites címek
    - IPv6 – 128 bites címek
  - **Internetes névfeloldási rendszer – DNS**
    - IP címhez rendel olvasható szöveges nevet
    - alfanumerikus karakterekből áll
    - adatbázis, protokoll és hálózati rendszer is egyben
  - Internet szolgáltatói hálózatok számozása (ASN)
    - ASN - Autonomous System Number
    - Egy adott adminisztratív domainhez tartozó szám (kb. mint az irányítószám)
    - Útvonalválasztáshoz kritikus – BGP
    - Régebben 16 bites szám, ma már 32 bites: jelölése x.y , ahol x és y is 16 bites. Kompatibilitás a régivel: o.y



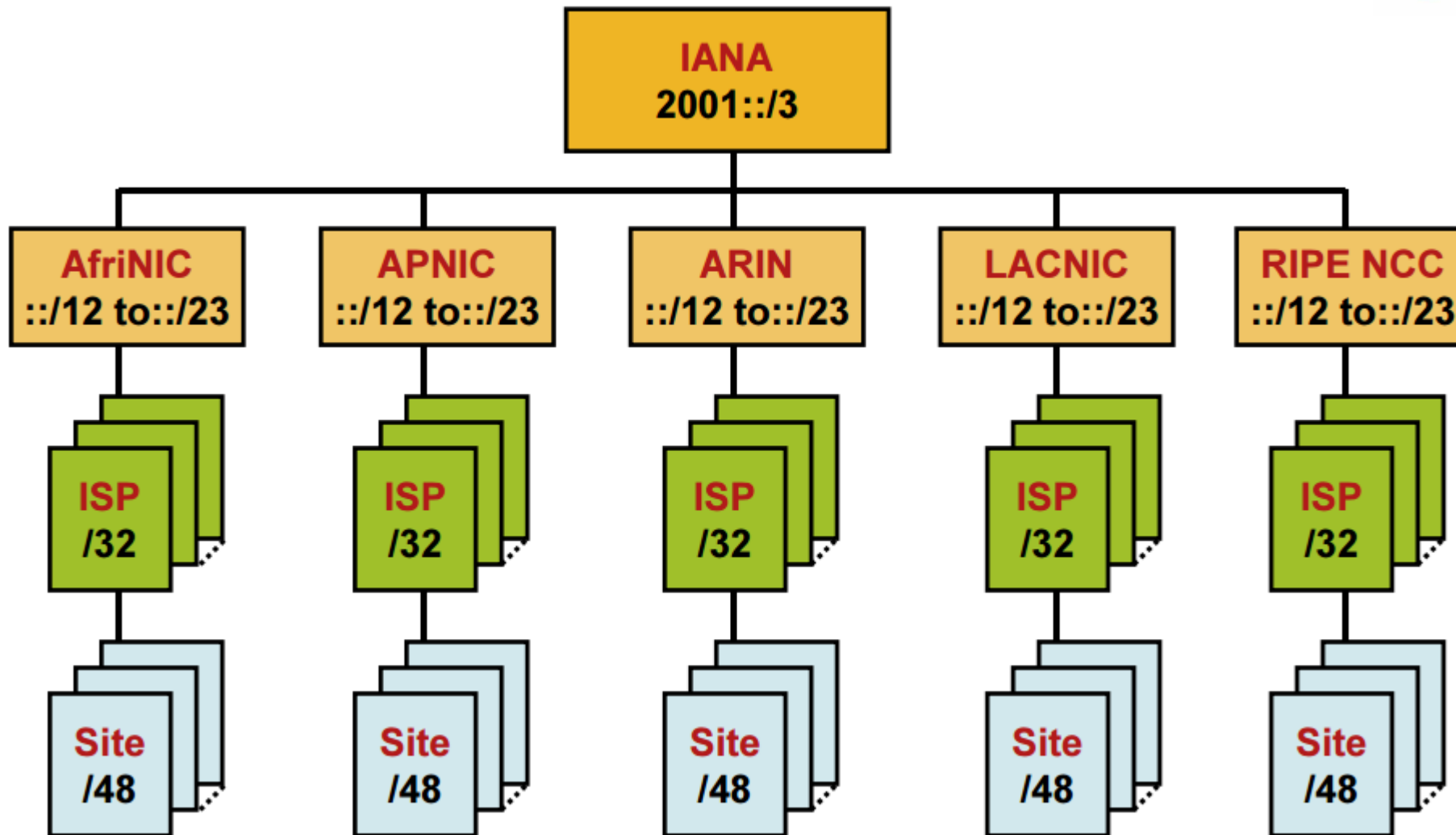
# Az Internetes erőforrás (CIR) szervezetek



- ICANN – Internet Corporation for Assigned Names and Numbers
  - A globáisan egyedi számtartományok adatbázisának koordinátora
  - IANA – Internet Assigned Numbers Authority részlege végzi a címzéspolitikai és menedzsment feladatokat
    - DNS – ez a legösszetettebb feladata, új TLD-k bevezetése, root szerverek (13 db – egy-egy klasztert alakjában) menedzselése, stb.
    - IPv4 és IPv6 címek
    - ASN számok
    - RIR szervezetek számára az IP címtartományok hozzárendelése
- RIR – Regional Internet Registry
  - Az IP címek és ASN számok helyi szervezetekhez vonnak továbbosztva (ők a szétesztásban a következő hierarchia szint - IETF RFC 7020)
  - 5 nagy RIR van: ld ábra, ők a Number Resource Organization-be (NRO) tömörülnek
  - Az Address Supporting Organisation (ASO)-t alapítják meg a számosztási politikához
  - A RIR szolgálja ki végül az ISP-ket (mint viszonteladó is) és a végfelhasználókat (prov. aggregatable és prov. independent)
- LIR – A RIR-ek LIR-eknek (Local Internet Registry) osztják tovább a címeket
  - Ők tehát a hierarchia következő szintje pl. ISP-k (legalul van a végfelhasználó pl. BME)
- **NTIA – National Telecommunications and Information Administration**
  - **USA Kereskedelmi Minisztériumhoz tartozó távközlési tanácsadó szakszervezet**
  - **a DNS root zone korábbi tulajdonosa és a mai napig felügyeleti szerve, melyet tervez leadni az üzemeltetést 1997-től a Verisign privát cég üzemelteti**



# Nevek regisztrálása hasonló a címekéhez



# A DNS nevek alapvető szerkezete

- A skálázhatóság érdekében a menedzsment hierarchikusan van szervezve, hasonlatosan más hasonló címekhez, pl. lakáscím, telefonszám
- Tartalmi szempontból nem feltétlen (gyakorlatilag nem) van hierarchizálva. Ilyen értelemben hasonló az IP címekhez is

**A DNS ugyanakkor teljesen független az IP címektől, egy teljesen független hálózat, bár infrastruktúráisan néha összenőnek, leginkább menedzsment okokból!**

# A DNS rövid története

- Az ARPANET esetében egy konkrét gépen található HOSTS.TXT file valósította meg ezt a feladatot
- A hálózat növekedése során
  - nőtt a hálózati terhelés
  - nem volt szinkronizálva, hogy ki milyen nevet választott, választhatott
- A jelenlegi DNS-t végül 1983-ban alapították (Paul Mockapetris - RFC 1034 és 1035)
- Azóta számos fejlesztésen módosításon ment át
- Mára a rendszer lényegében egy nagy méretű elosztott adatbázissá változott, de feladata továbbra is kb. a HOSTS.TXT



# A DNS lényegi részei

- Egy nagy méretű elosztott adatbázis, számos szerverrel és a közöttük lévő kommunikációval (protokollal)
- A névfeloldó eljárás, melynek segítségével egyik típusból a másikba konvertálunk
- Funkcionálisan az alábbi részekből áll:
  - A névtér, amit az adott erőforrásokhoz rendelünk (többnyire az IP címekhez)
  - A szerverek, amelyek az adatokat tartalmazzák és a lekérdezésekre válaszolnak, bárki számára válaszolnak (bizonyos felelősségi körrel persze)
  - A lekérdező szoftverek (Resolver), melyek nem önálló szolgáltatások, hanem programokból meghívható szubrutinok formájában találhatók meg. Minden program, ami igényli a címfordítást külön meghívja. Pl. libc:
    - `gethostbyname(char *name);`
    - `gethostbyaddr(char *addr, int len, type);`

# A DNS mint elosztott adatbázis

- Az elosztottságból számos kihívás adódik:
- Adatok koherenciája: nincs egy olyan gép sem, ami mindenről tud (nem olyan pl. mint a DFZ, ahol bizonyos core routerek minden tudnak). Hiba esetén inkonzisztencia léphet fel, ami azonban előbb utóbb megszűnik. Szándékos is lehet: pl. cache poisoning ld. később. Egy gépen belül az adatokhoz sorozatszámot rendelünk (kb. dátumot), ebből látszik, hogy mennyire friss.
- Kommunikációs protokoll trade-off: Ha túl gyakran frissítünk, akkor nagy a hálózati overhead. Ha túl ritkán frissítünk nő az inkonzisztencia, esetleg szolgáltatás kiesés vagy reakció változási igényre
- Rendszerek együttműködése: programverziók, lekérdezések, működési idők, tartalom pl. karakterkódolás

# A DNS skálázhatósági tulajdonságok

- Az egy gépen található adatbázis mérete tetszőlegesen nagy lehet. Csak a gép adattárolója szab határt.
  - Nagyságrendileg akár sok millió nevet is tartalmazhat, ez azonban jelentősen lassíthatja gépet, ráadásul sérülékennyé válik a rendszer (single point of failure)
- Hála az egyszerű protokollnak a kiszolgálható lekérdezések száma igen magas lehet.
  - 10-20 ezer lekérdezés másodpercenként nem jelenthet gondot.
  - Mindazonáltal a hatékonyságot az utóbbi időben gyorsabb adatbázis-elérési technikákkal igyekeznek növelni
- A lekérdezések által okozott forgalomterhelés eloszlik elsődleges, másodlagos szerverek és gyorsító tár megoldások által

# A DNS rendszer megbízhatósága

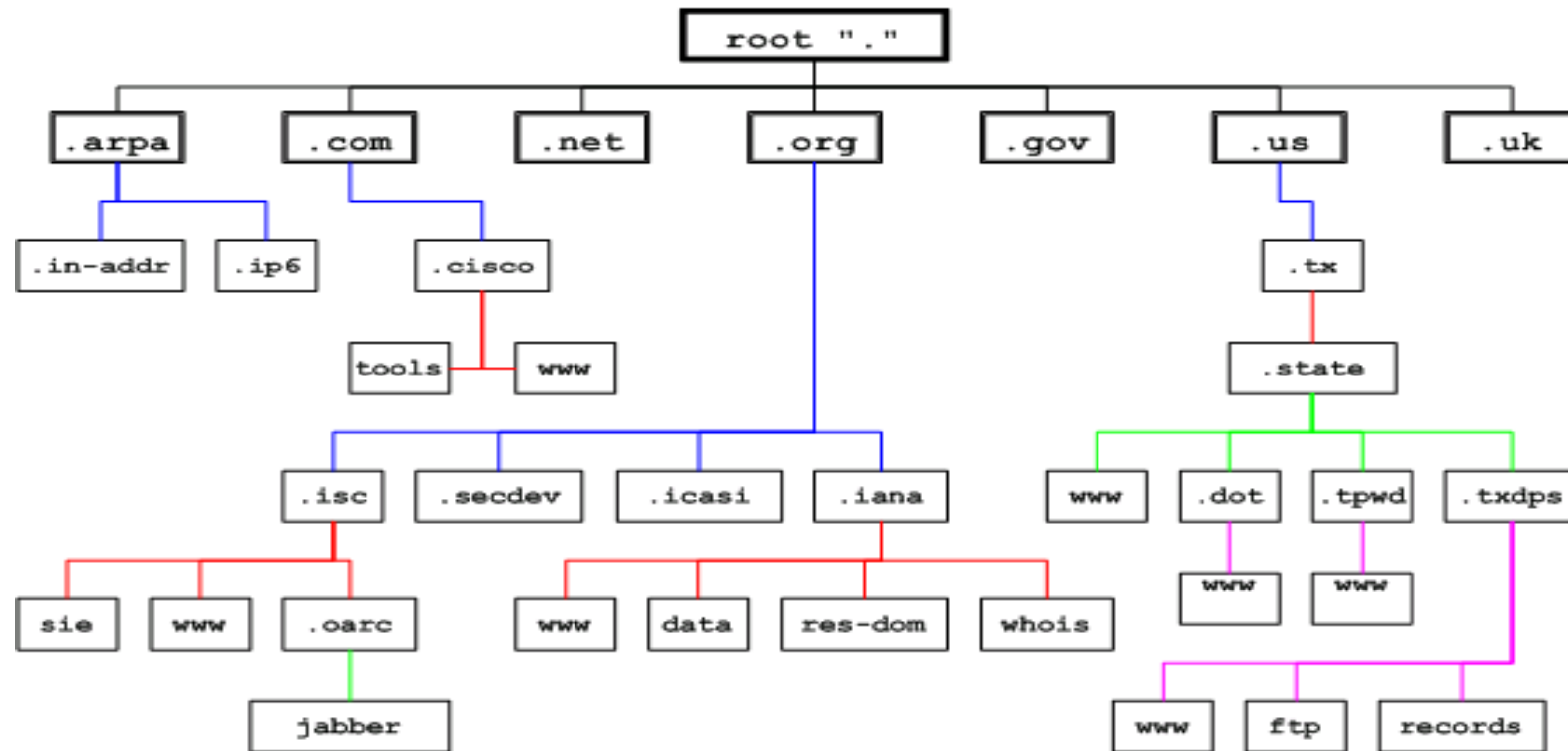
- Az elsődleges és másodlagos szerverek is tárolják ugyanazt az adatot: a másodlagos szervereken vannak backup mentések.
- A másodlagos szerverek kérdezhetnek az elsődlegesektől, de kérdezhetnek további másodlagos szerverektől is. Mivel a gép nem tudja, hogy elsődlegestől vagy másodlagostól kérdezte, így az out-of-date adatok elérhetősége sajnos megnőhet.
- A kliensek gyorsítótárakból is tudnak adatokat lekérdezni.
- A DNS protokollok UDP és TCP kapcsolattal is tudnak kommunikálni, UDP esetén a biztonságos kommunikációt a felsőbb rétegek (DNS protokollok) biztosítják

# A DNS névtér szerkezete

- A névtér tehát hierarchikusan épül fel (skálázhatóság)
- A hierarchia alapja lehet:
  - Az elnevezendő szervezet típusa: pl. .com, .org stb.
  - Geográfiai elhelyezkedés alapú, pl. országok: .hu, .de
  - Szervezeti felépítés, működés alapján: bme.hu, tmit.bme.hu, tszvez.tmit.bme.hu
  - Konkrét objektumok pl. személyek vagy személyek által adott nevek
- Hogy mi alapján történik a hierarchia felosztása, az domain-ektől, azon belüli szintektől, de az országoktól, geográfiai lokációtól is változik
- Az utóbbi időben ezek a szabályok szinte teljesen kezdenek megszűnni. Mindent lehet...

# A DNS névtér felépítése

- A DNS név pontokkal elválasztott szavakból (betűk+számok+mínusz jel) áll. Minden rész legalább 2 betű pl.: heszi.tmit.bme.hu. Max 128 mélység



`.arpa`: primarily used for address to host mappings

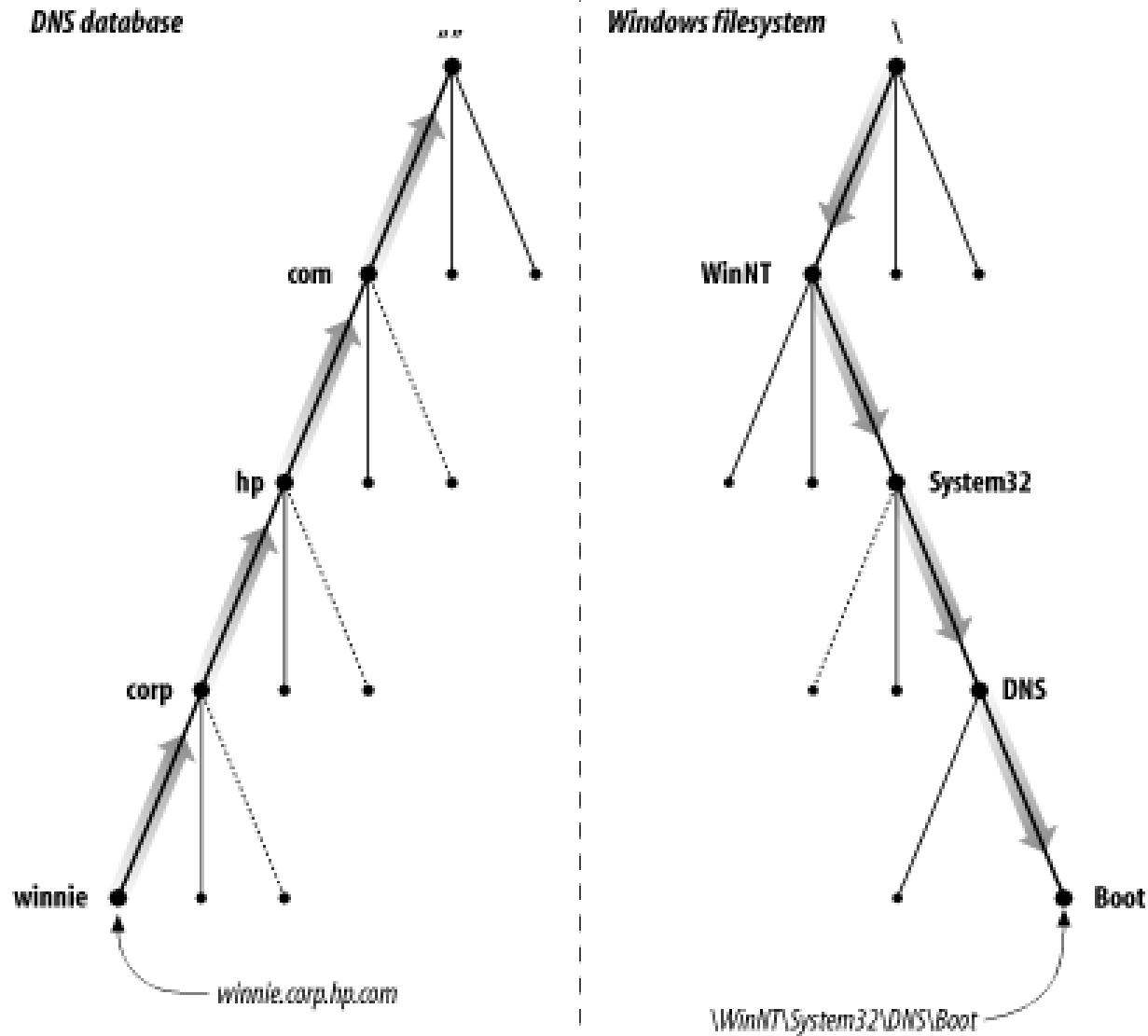
`.com`, `.net`, `.org`, `.org`: are generic TLDs (gTLD)

`.us`, `.uk`: are country code TLDs (ccTLD)

# A DNS nevek szerkezete

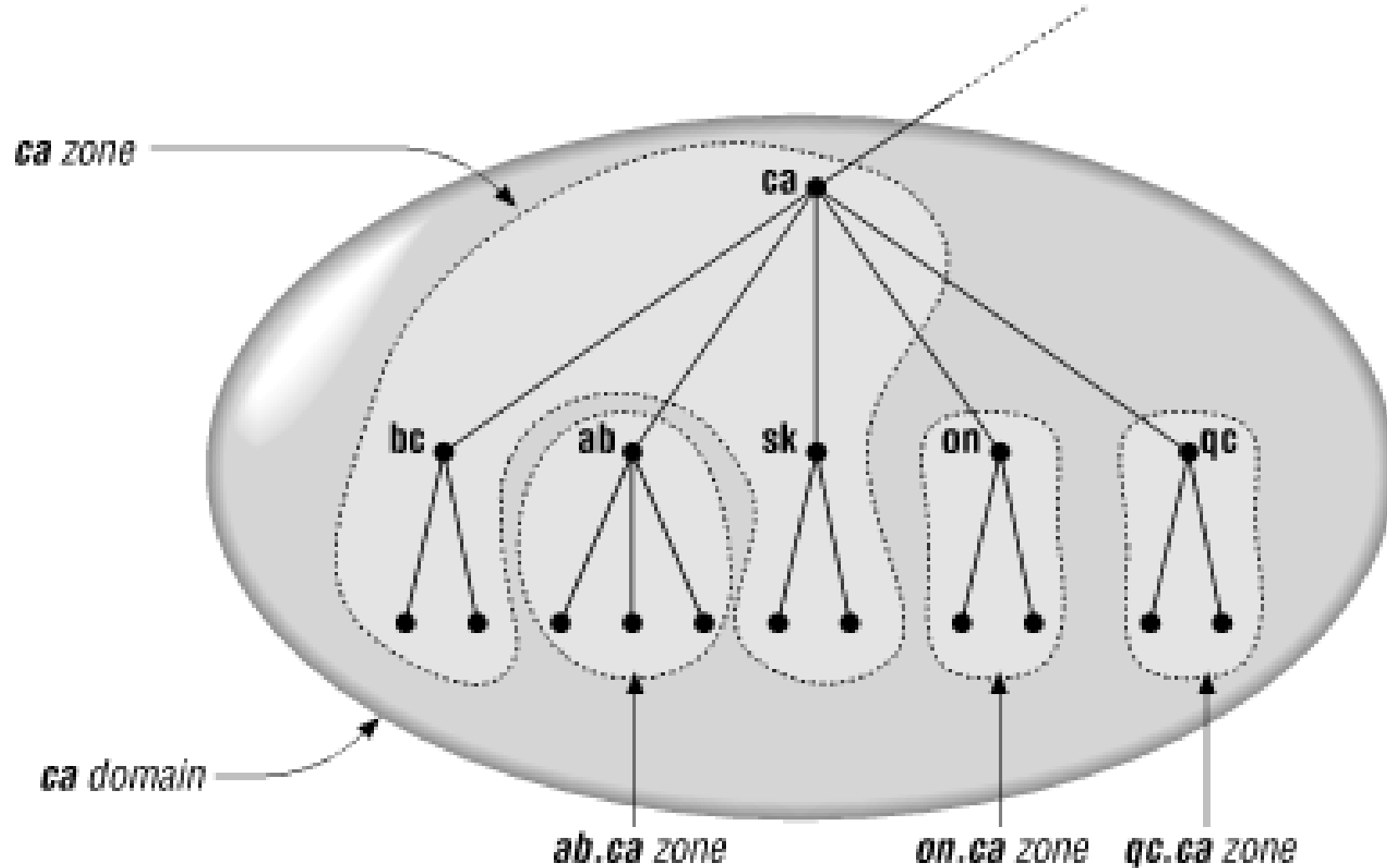
- heszi.tmit.bme.hu.:
- . → Root Domain
- .hu → Top Level Domain TLD
- .bme → Secondary Subdomain
- .tmit → további aldomain-ek...
- heszi → gép neve
- .com, .gov, .org, .net, .mil, .int, .edu, ... gTLD (generic TLD)
- .info, .name, .pro, .coop, .biz, .museum, .aero, ... további új gTLD-k
- .hu, .us, .uk, .de, .... ccTLD (country code TLD)
- .arpa (reverse DNS lekérdezéshez használt domain)
- .test (non-ASCII kódolt TLD-k használatának tesztelése)

# A DNS névtér és a Unix fájlrendszer

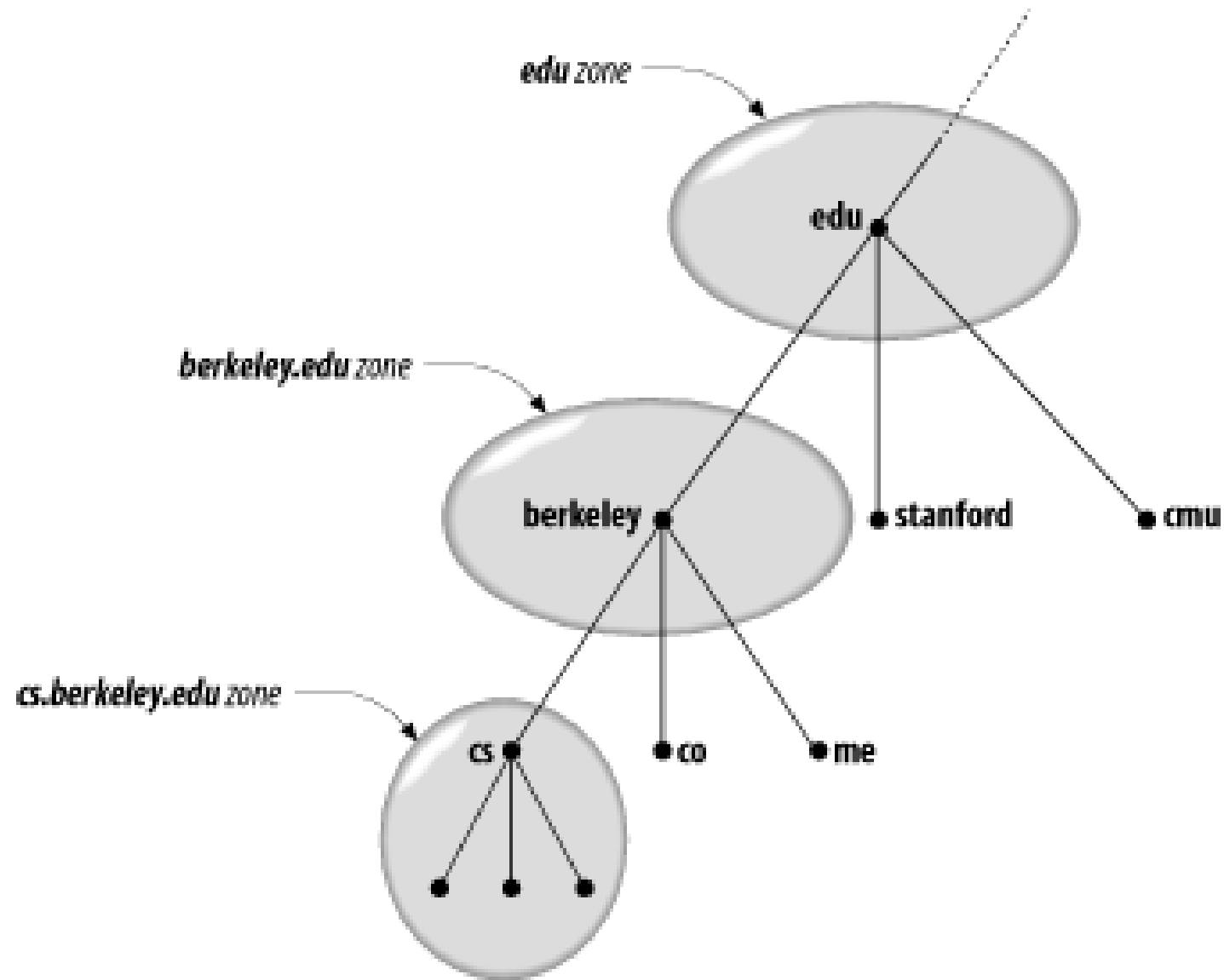




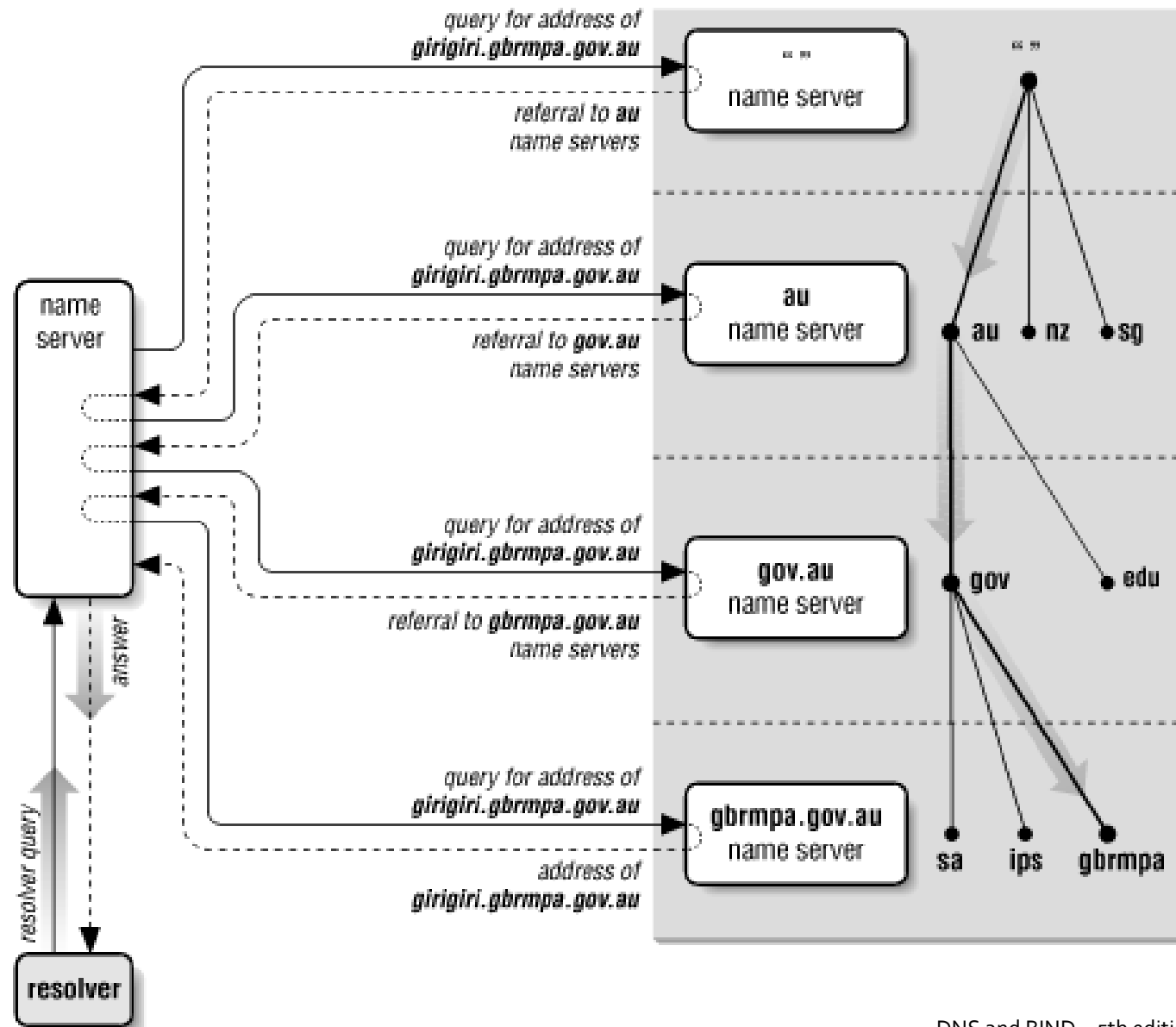
# Subdomain-ek szerkeze: subdomain vs. zone



# Subdomain-ek szerkeze: a delegálás



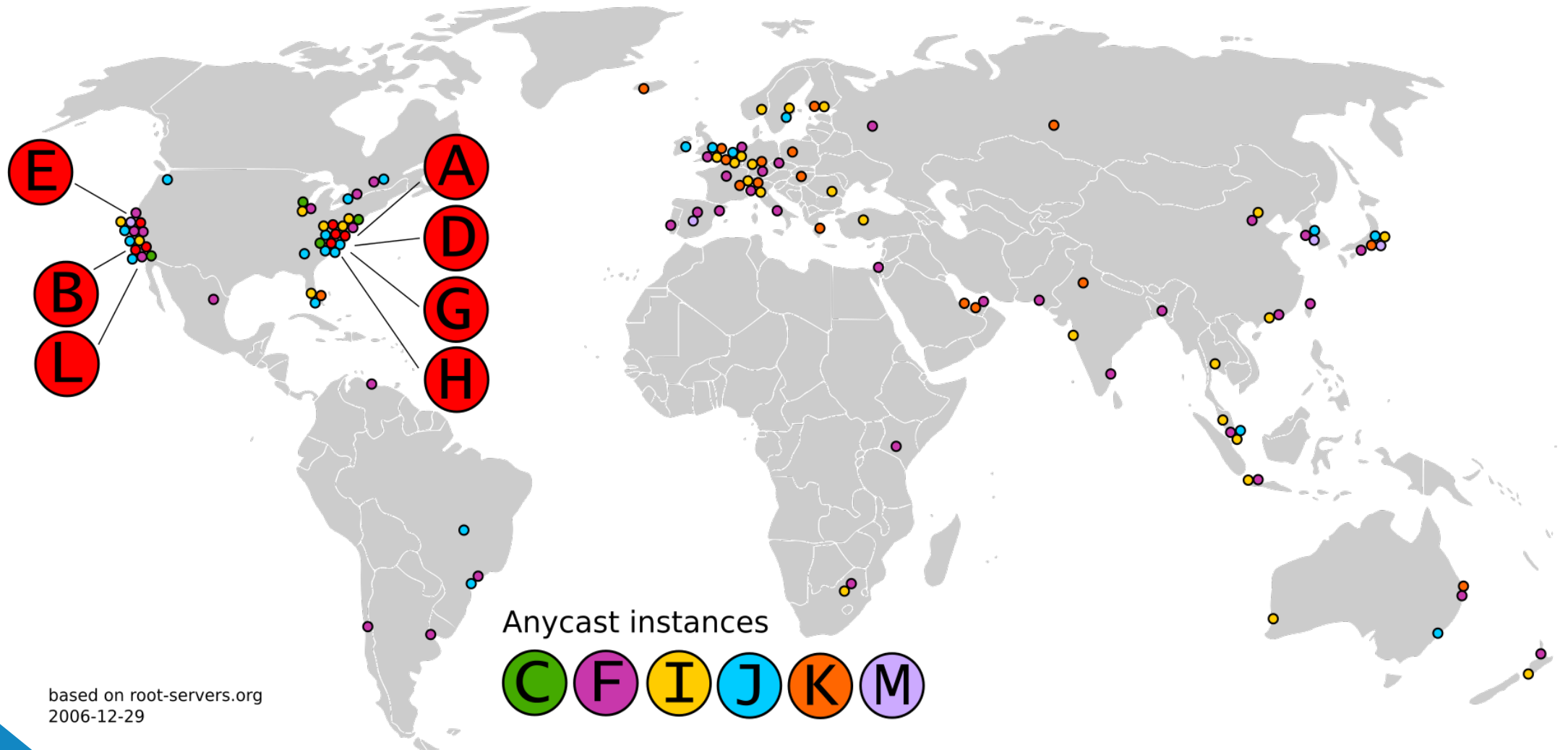
# A rekurzív névlekérdezés folyamata



# A lekérdezés láncolata – a root domain szerverek

1. Ha lekérdezzük pl. a heszi.tmit.bme.hu címet, akkor először mi fordulunk a saját névszerverünkhöz.
  2. A lokális névszerver megállapítja, hogy helyi domain-e, és ha nem akkor a végéről indul visszafele: először a root domain-t (.) kérdezi meg, hogy hol a .hu domain. Ehhez minden resolver klincs (ez esetben a helyi névszerver) tudja az IP címet: root domain (root name) szerverek, ld. ábra. (ezek kezdetben pl. kézzel vannak letöltve a gépre egy ún. root hint fájlba).
  3. A root névszerverek bármelyike megmondja a .hu domain-hez tartozó névszerverek IP címét
  4. A .hu névszerverhez fordulva, ő megmondja a bme.hu névszerverét
  5. A bme.hu névszerver megmondja a tmit.bme.hu névszerverét
  6. Végül a tmit.bme.hu névszervere közvetlen tudja a heszi.tmit.bme.hu IP címét (152.66.247.138)
- A root névszerverekből jelenleg 13 db különböző nevű van. Azért pont ennyi mert amikor a teljes információt egy UDP csomagban elküldik, ennyi fragmentáció nélkül is belefér biztosan egy csomagba a mai link layer hálózati technológiákkal.
  - Persze valójában ennyi nem tudná kiszolgálni az egész világot, ezen aliasok alatt számos anycasttal címzett szerver dolgozik még (nem számítva melegtartalékokat)

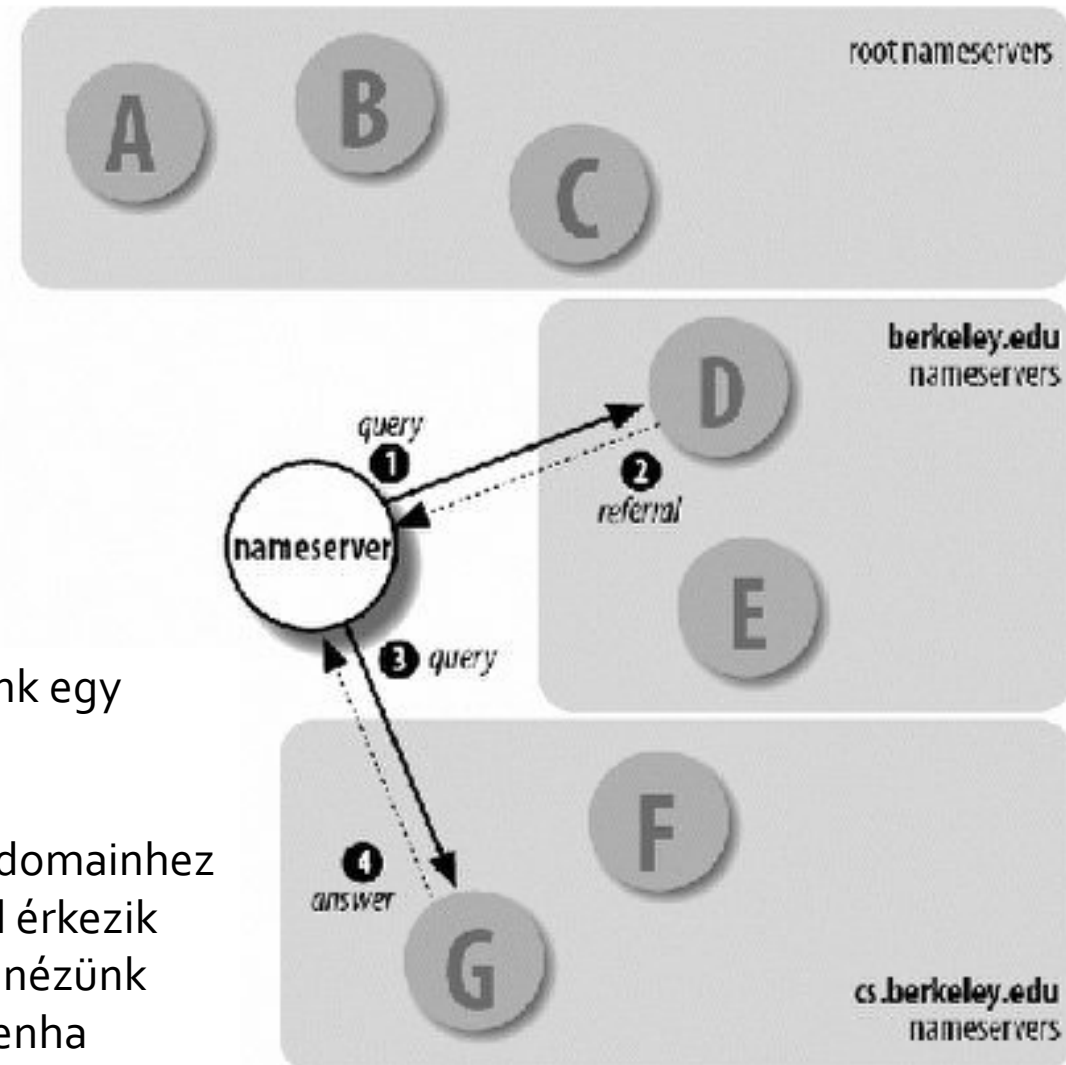
# A root domain szerverek a világon



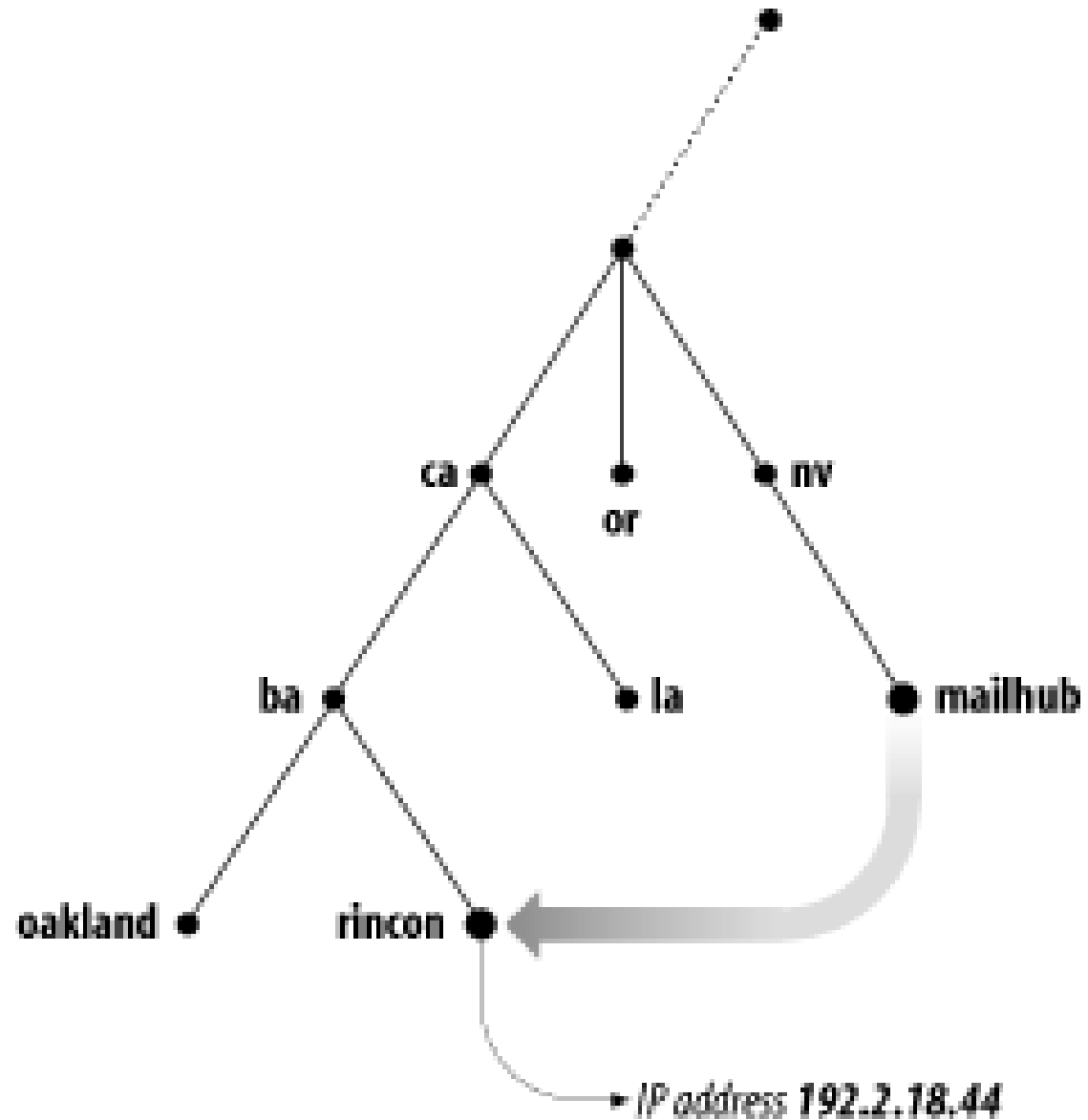
based on root-servers.org  
2006-12-29

# A gyorsítótáras/iteratív névlekérdezés folyamata

1. Egy korábbi kérés miatt lekérdezzük egy magasabb domainből adatokat\*
  2. Válasz megérkezik
  3. Ismét kérdés indul ugyanahhoz a domainhez
  4. A válasz most már a gyorsítótárból érkezik
- \* Iteratív lekérdezés során, akkor sem nézünk utána a válasznak felsőbb domainekbenha közvetlenül nem tudjuk még.

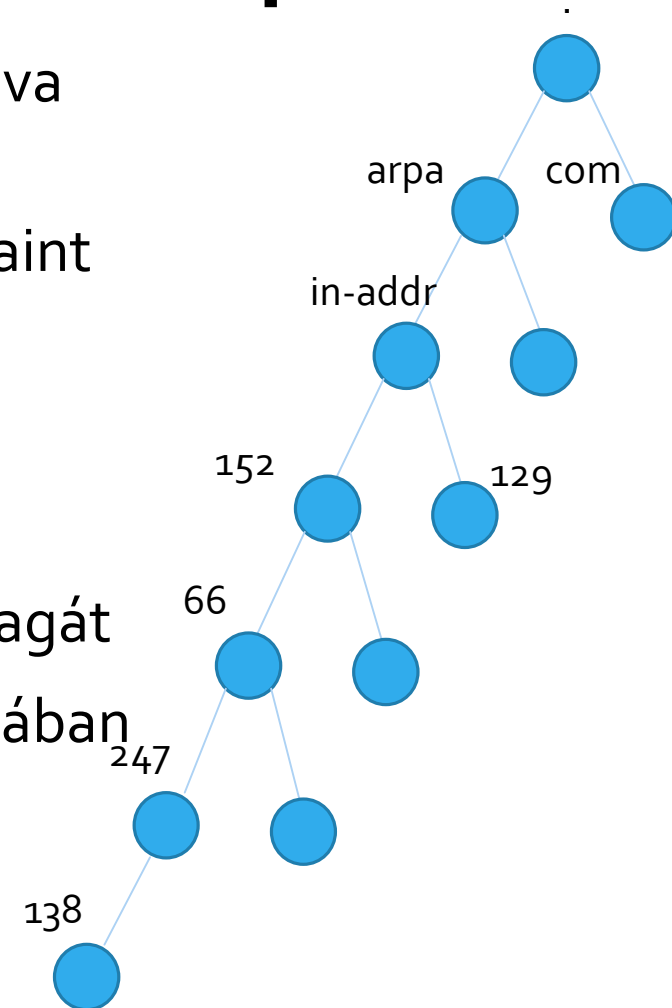


# A DNS árnevek (alias)



# Hogy működik a reverse DNS lookup?

- A probléma ugye, hogy a nevek alapján van struktúrálva az IP cím hozzárendelés
- Ha az IP cím alapján akarnánk keresni, minden aldomaint végig kellene nézni.
- Megoldás: az IN-ADDR.ARPA aldomain
- Minden aldomain regisztrál külön enne a domainbe is
- Az IP címnek megfelelő tartományba bejegyezteti magát
- Pl. ha rákeresek a 152.66.247.138 IP címre, akkor valójában a 138.247.66.152.in-addr.arpa név mellé bejegyzett információkat keresem.
- Ehhez a névszerverem elmegy az in-addr.arpa névszerveréhez, aki megmondja hol van a 152.in-addr.arpa névszervere, stb., analóg módon a korábbiakhoz





# Hogyan szerzünk domain nevet?

1. Kitalálunk egy TLD-t, ami alá tartozni szeretnénk (ma már szinte minden domain elérhető általános célra). Ilyet újat bejegyezni valószínűleg nem fogunk tudni (Internet politikai kérdés). Ld. pl. a .tv domain-ért fizetett pénzeket...
  2. Ennek alapján utánanézzünk, hogy annak milyen aldomainjei vannak, és azok közül keressünk-e egyet, vagy újat akarunk.
  3. Ellenőrizzük, hogy a név létezik-e már: pl. nslookup segítségével
  4. Ha még nincs, akkor megkeressük azt az aldomain üzemeltetőt, aki alá be szeretnénk tartozni.
  5. Megegyezünk az aldomain üzemeltetővel, hogy mennyi pénzért csatlakozhatnánk be (tipikusan pl ~10e Ft/év).
- A feladat egy kis nyomozást igényel, kezdhetjük pl. a [www.allwhois.com](http://www.allwhois.com) oldalon
  - Ugyanakkor ma már jónéhány regisztrátor létezik, aki utánanéz ennek helyettünk és a regisztrációt végigcsinálja a megbízásunkból
  - Különböző regisztrátorok különböző szolgáltatásokat nyújtanak, egyesek hatékonyabban dolgoznak mint mások. Híres pl. a goddady.com

# WHOIS adatbázis

- A 80-as évek eleje óta létezik egy olyan adatbázis ill. hozzá tartozó lekérdező protokoll, amely az internetes virtuális erőforrások lekérdezésére lett létrehozva
- Ahogy a DNS decentralizálttá vált, a whois adatbázisok is megszorodtak. A baj az, hogy ezek nem elosztott adatbázisként működtek, hanem egymástól független, az operátor által egyedileg update-elt rendszerek. Pl. minden RIR üzemeltet egy whois szerveret, de sok regisztrátor cég is, ennek segítségével tud könnyen szabad domaint eladni
- Működése a közelmúltig bezárólag egy teljesen szöveg alapú (gyakorlatilag egyszerű telnetet használó protokoll)
- Eredetileg bármire válaszolt, pl. a john keresőszóra minden john által regisztrált vagy ezt a karakterláncot tartalmazó domaint, visszaadott
- Ma adatbiztonsági okokból ezt már erősen lekorlátozták
- Fejlettebb implementációk már képesek iteratíván is keresni ha az adott whois szerver hierarchikusan tárolja egy idegen domainhez tartozó whois szerver címét.
- A különböző regisztrátorok különböző módszerekkel működtetik az adatbázisukat
- A használatukhoz ld. pl. a második gyakorlaton alkalmazott hálózati címek lekérésének folyamatát

# BIND

- Messze a legnépszerűbb és legszélesebb körben használt DNS implementáció, egyben a referencia implementáció is
- Készült: California Berkeley (UCB) 1983: *Berkeley Internet Name Domain (BIND)*
- A legfrissebb elterjedt verziója a 9.x (9.5?), az Internet Software Consortium (ISC) pár éve készítette a legújabb 10-es verziót, melyből azonban közben kiszállt és átnevezte BINDY-nak. Nyílt forráskódúként az internetes közösség fejlesztésére bízta
- Az új verziók a terv szerint már nem csak a txt fájlokat, de az adatbázisokat is támogatni fogják, sőt akár kevert adatbázisokat is (MySQL, PostgreSQL stb.)
- Számos alternatív implementáció létezik extra funkciókkal (pl DDNS), de nem népszerű egyik sem, fizetős szolgáltatások megvalósítására

# A BIND felkonfigurálása

- Két fontos fájl típusra van szüksége a névszervernek:
  - A konkrét információkat tároló zóna adatfájlokra
  - A BIND konfigurációs fájljára
- A konkrét információkat tároló zóna adatfájlok tartalmazzák:
  - Azoknak a zónáknak az adatait amiért felel az adott szerver: az adott zónán belül, konkrétan mely gépek vannak és azoknak mi az IP címe
  - Tartalmazza az adott zóna névszervereinek IP címét
  - Álneveket
  - A domainhez tartozó levelezőszerverek IP címét
  - Aldomainekhez tartozó információkat

# Példa zóna adatfájl egy névszerveren

\$TTL 3h

tmit.bme.hu. IN SOA nev1.tmit.bme.hu. admin.tmit.bme.hu. (

1 ; sorozatszám

3h ; adatok frissítése 3 óránként

1h ; újrapróbálkozás 1 óránként

1w ; lejárát 1 hét múlva

1h ) ; negatív caching TTL 1 óra

*; Name servers*

tmit.bme.hu. IN NS nev1.tmit.bme.hu.

tmit.bme.hu. IN NS nev2.tmit.bme.hu.

*; Addresses for the canonical names*

localhost.tmit.bme.hu. IN A 127.0.0.1

heszi.tmit.bme.hu. IN A 152.66.247.138

retvari.tmit.bme.hu. IN A 152.66.246.231

*; Aliases*

zalan.tmit.bme.hu. IN CNAME heszi.tmit.bme.hu

# A reverse DNS-hez is szükség van egy zóna adatfájltra!

**\$TTL 3h**

**tmit.bme.hu. IN SOA nev1.tmit.bme.hu. admin.tmit.bme.hu. (**

**1** ; sorozatszám

**3h** ; adatok frissítése 3 óránként

**1h** ; újrapróbálkozás 1 óránként

**1w** ; lejárat 1 hét múlva

**1h )** ; negatív caching TTL 1 óra

*; Name servers*

**247.66.152.in-addr.arpa. IN PTR nev1.tmit.bme.hu**

**247.66.152.in-addr.arpa. IN PTR nev2.tmit.bme.hu**

*;A konkrét nevek*

**138.247.66.152.in-addr.arpa. IN PTR heszi.tmit.bme.hu**

**231.247.66.152.in-addr.arpa. IN PTR retvari.tmit.bme.hu**

# Az adatfájlokban szereplő leggyakoribb rekord típusok

- SOA rekord: Az adott domainhez tartozó szervezeti adatok
  - elsődleges névszerver, a domain adminisztrátorának email címe
- NS rekord: Az adott domainhez tartozó névszerverek
  - legalább kettő: elsődleges és másodlagos (master és slave)
- A rekord: A konkrét gépek IP címe
  - felsoroljuk a zónán belüli gépek neveit és mellettük az IP címet
- PTR rekord: Ez az inverz DNS keresésnél használatos
  - felsoroljuk a gépek in-addr.arpa "formátumú" neveit és után az igazi neveit
- CNAME rekord: álnevek, alternatív nevek megadása
- MX rekord: Az adott zónához tartozó mail szerverek listája

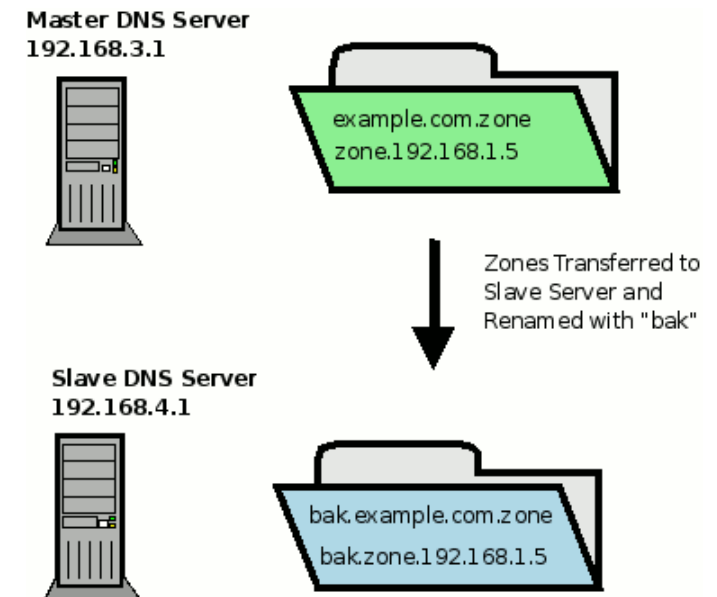
# Aldomainek létrehozása

- Két lehetőség:
  1. Az aldomain-t mi magunk adminisztráljuk tovább, tehát benne marad a zonában:
    - Ekkor egyszerűen felsoroljuk a neveket, melyek tartalmazzák az aldomain-t is pl.:  
heszi.oktato.tmit.bme.hu.      IN A    152.66.247.138
  2. Továbbdelegáljuk az aldomain működtetését. Ilyenkor a zona adatfájlban jelezni kell, hogy egy adott aldomain adatait egy másik névszerveren lehet elérni, és ennek persze meg kell adni az IP címét is (ez utóbbi az ún. glue record, valójában egy szimpla A record)
- Látható, hogy a név és az IP cím nem kell hogy bármilyen korrelációban legyen, de pl. ha olyan nevet akarok feloldani, ami nem az én zonámba tartozik azt észreveszi a BIND és nem engedi.



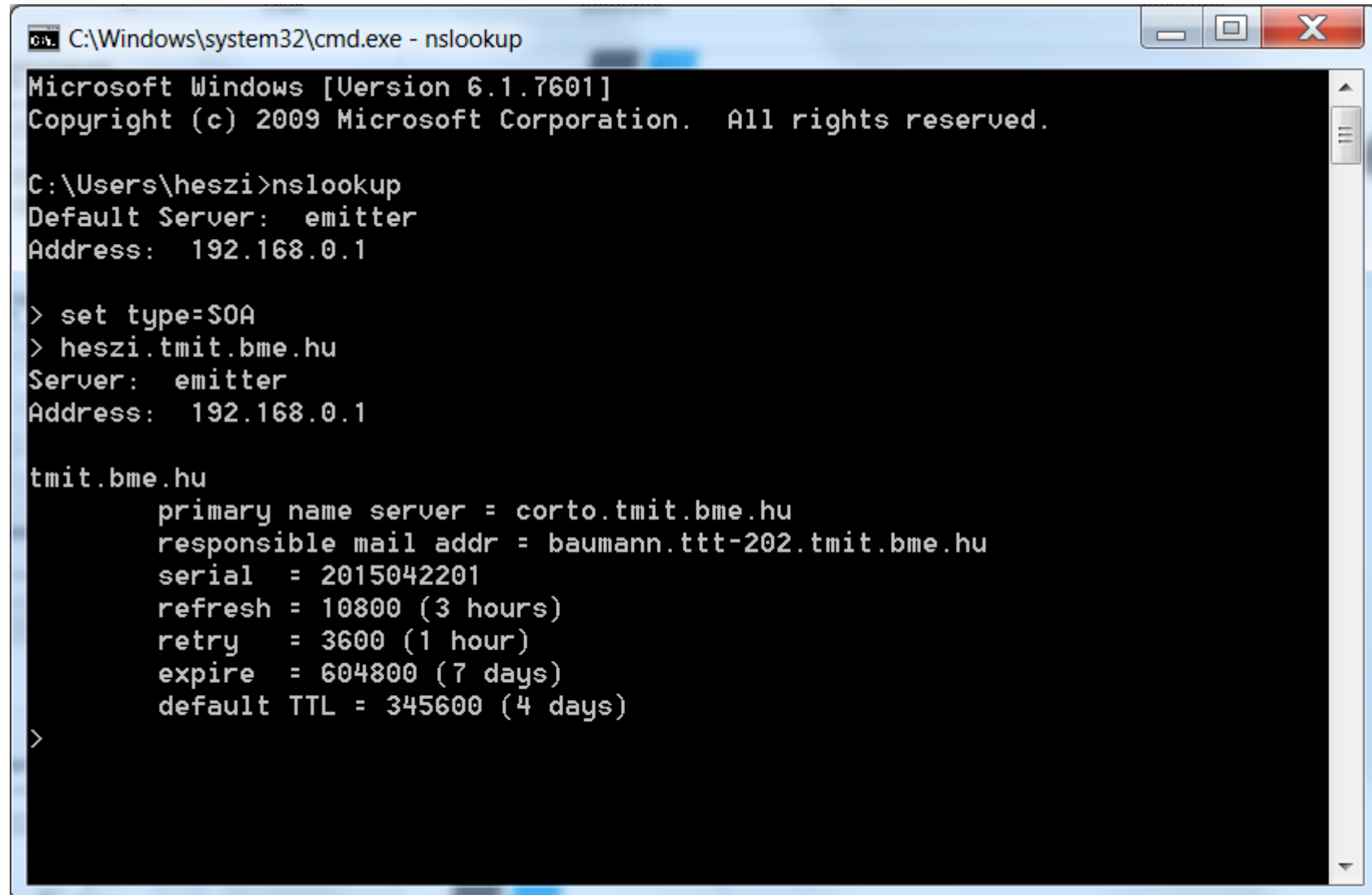
# Elsődleges és másodlagos névszerverek

- Az másodlagos (slave) szerverek rendszeresen lekérdezik az elsődleges (master) adatait
- A master és slave funkciók zonánként eltérőek lehetnek, tehát igazából egy szerver sorkszor vegyesen master és slave is
- Ha a slave lekérdezi a mestert (zone transfer) akkor az adatot elmenti pl. backupként és ha nem éri el legközelebb akkor ebből olvassa ki az adatokat egy ideig, amíg a TTL él



**Zone Transfer**

# Példa DNS adatok lekérésére



```
C:\Windows\system32\cmd.exe - nslookup

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\heszi>nslookup
Default Server: emitter
Address: 192.168.0.1

> set type=SOA
> heszi.tmit.bme.hu
Server: emitter
Address: 192.168.0.1

tmit.bme.hu
    primary name server = corto.tmit.bme.hu
    responsible mail addr = baumann.ttt-202.tmit.bme.hu
    serial = 2015042201
    refresh = 10800 (3 hours)
    retry = 3600 (1 hour)
    expire = 604800 (7 days)
    default TTL = 345600 (4 days)
>
```

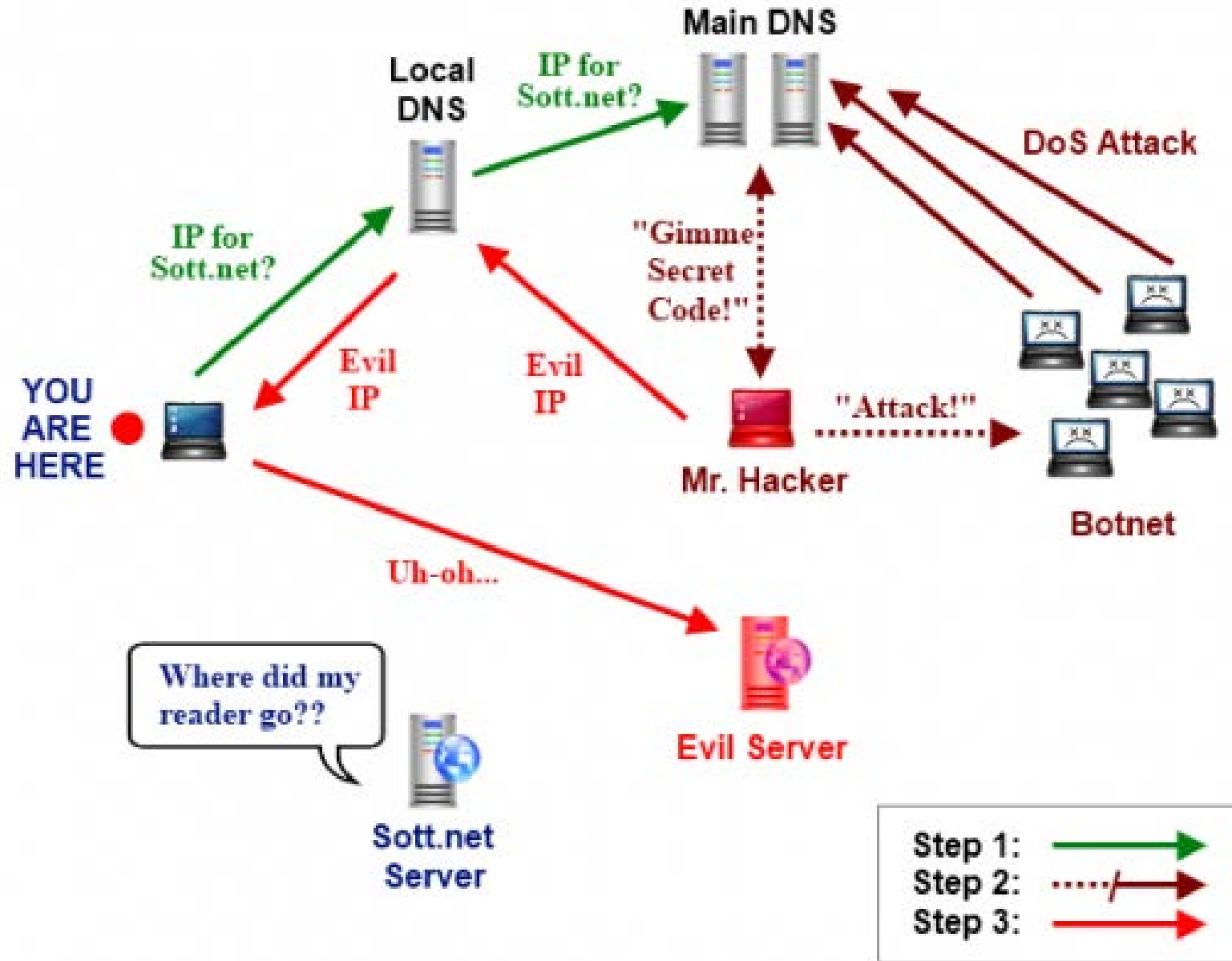
# Levelek útvonalválasztása

- Ha küldök egy levelet a telea@poca.com email címre, amelyre mondjuk korábban még sosem küldött senki, az hogyan talál el végül a megfelelő postaládába?
- A levélküldő kliens megnézi a poca.com DNS domain MX rekordját, amiből kiderül, hogy melyik szerverhez küldje
- Több MX rekordja is lehet egy domainnek, melyeknek prioritást is lehet adni
- A levelező szervernek nem kell címzett domainban lennie, csak az azért felelős DNS szerverben be kell jegyezni
- A lelevelezéshez további rekord típusok (pl. encrypter) is tartozhatnak

# A DNS biztonsági kérdései

- A DNS az Internet talán legtámadhatóbb pontja, achilles sarka. Ha pl. egy weboldalt keresünk, de a DNS névfeloldás hamis IP címet szolgáltat az gyakorlatilag számunka észrevehetetlen. Ez pl. a DNS cache poisoning segítségével történhet, ld. köv fólia
- Hamis gépre bejelentkezve, olyan adatokat adhatunk meg melyekkel visszaélhetnek: adathalászat/fishing
- DNSSEC: Cél az elérhető adatok hitelessége, integritása, (de titkosítása pl. nem!) – a megoldás a root névszerverek szintjén már működik, lejjebb még nem annyira jellemző!
- FCrDNS: Bizonyos támadások, spammek kiszűrésére könnyen kivitelezhető megoldás lehet, ha egy gépet miután feloldottunk (név->IP cím) visszafelé is lekérdezzük, hogy ugyanazt a nevet kapjuk-e vissza. Ennek megváltoztatása egy támadó által többnyire nem lehetséges, ezért könnyű azt detektálni. Néha ugyanakkor false-positive eredményt is szolgáltathat, ami igen zavaró lehet, ezért használata nem annyira közkedvelt.

# DNS Cache poisoning

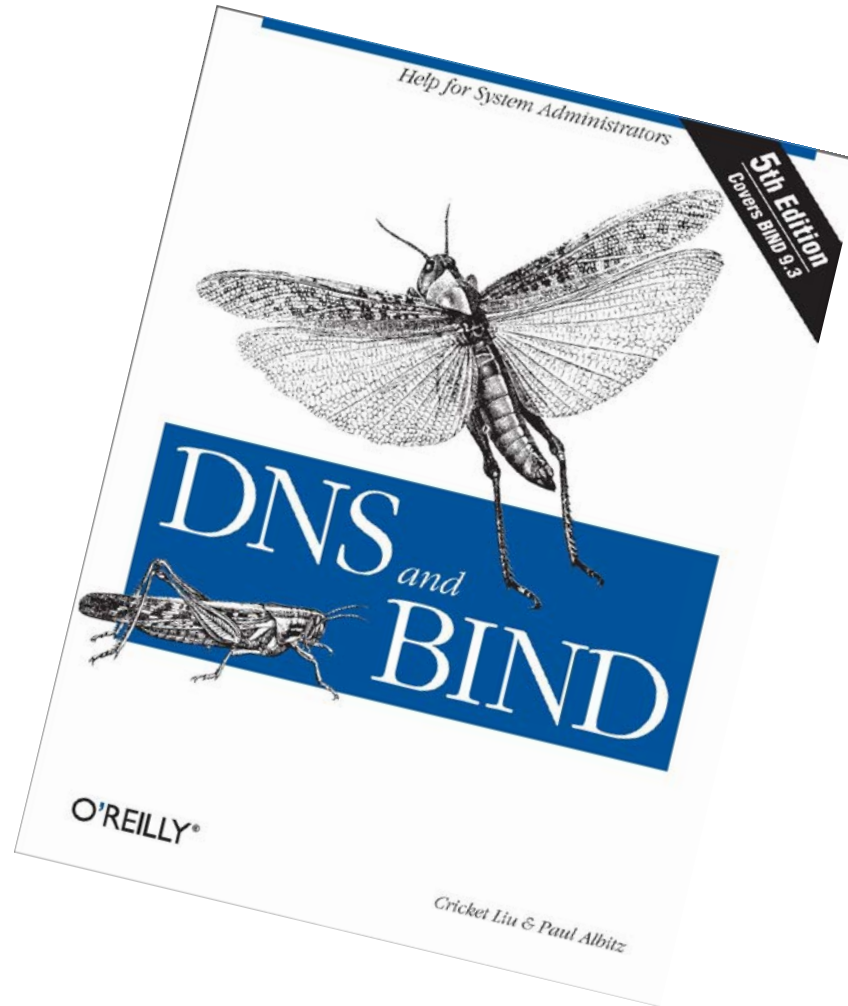




# Dynamic DNS rendszerek

- A DDNS vagy DynDNS rendszerek lényege, hogy gyakorlatilag on-the-fly képesek egy új domain nevet bejegyeztetni
- **Tehát pl. nem szükséges 3 órát várni a slave szerverek frissítésére, hanem azonnal frissül**
- Ezek a domain nevek általában egy adott DDNS szolgáltató birtokában vannak, a domaint ő tartja karban és speciális szoftverek segítségével frissíti az adatbázisát
- Használata: olyan esetekben lehet rá szükség ha rögzíteni akarunk egy domain nevet, de az IP cím rendszeresen változik, pl. amikor egy szolgáltató dinamikusan oszt ki dialup IP címeket

# Ajánlott szakirodalom



Köszönöm a figyelmet!