

Számítógépes Hálózatok

8. gyakorlat

Óra eleji kisZH


- Elérés:
 - <https://canvas.elte.hu>

☰ 2018/19/1 XK85DZ-IP-08abcSZHG - Számítógépes hálózatok GY. (BSc,08,A) > Kvízek


2018/19/1
Kezdőlap
Hirdetmények
Feladatok
Fórumok
Értékelések
Résztevők
Oldalak
Fájlok
Tematika
Tanulási
eredmények
Kvízek
Modulok
Beállítások

+ Kvíz ⚙

▼ Gyakorló kvízek

 Demo kvíz

Elérhető Többes határidő | Határidő Többes határidő | 5 pont | 5 kérdés

 ⚙

FORGALOMIRÁNYÍTÁS, HÁLÓZATI BEÁLLÍTÁSOK, FORGALOM FIGYELÉS, WIRESHARK

Forgalomirányítás

- **Definíció:** a hálózati réteg szoftverének azon része, amely azért a döntésért felelős, hogy a bejövő csomag melyik kimeneti vonalon kerüljön továbbításra.
- **Elosztott Bellman-Ford forgalomirányítási algoritmus:**
 - más néven távolságvektor alapú forgalomirányítás
 - minden router-nek egy táblázatot kell karbantartania,
 - amelyben minden célhoz szerepel a legrövidebb ismert távolság,
 - és annak a vonalnak az azonosítója, amelyiken a célhoz lehet eljutni.
 - A táblázatok a szomszédoktól származó információk alapján frissítik.

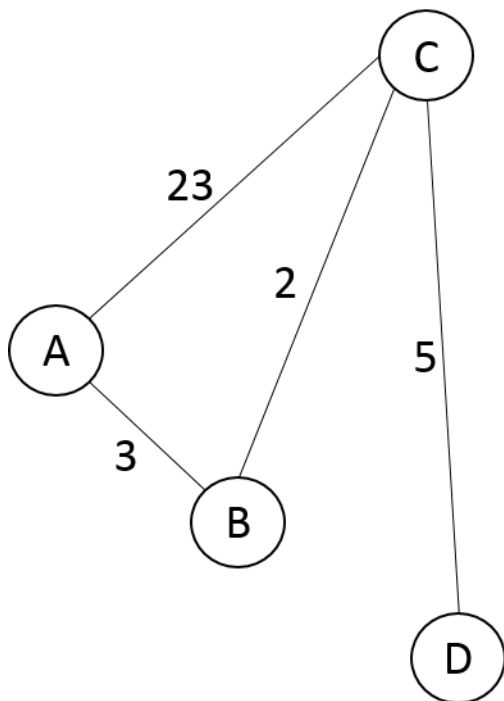
Elosztott Bellman-Ford algoritmus

- Minden csomópont csak a közvetlen szomszédjaival kommunikálhat.
- Minden állomásnak van saját távolság vektora. Ezt periodikusan elküldi a direkt szomszédoknak.
- A kapott távolság vektorok alapján minden csomópont új táblázatot állít elő.

Elosztott Bellman-Ford algoritmus pszekokódjá

1. **Initialization:**
2. **for all** neighbors V **do**
3. **if** V adjacent to A
4. $D(A, V) = c(A, V);$
5. **else**
6. $D(A, V) = \infty;$
7. **loop:**
8. **wait** (link cost update or update message)
9. **if** ($c(A, V)$ changes by d)
10. **for all** destinations Y through V **do**
11. $D(A, Y) = D(A, Y) + d$
12. **else if** (update $D(V, Y)$ received from V)
13. **for all** destinations Y **do**
14. **if** (destination Y through V)
15. $D(A, Y) = D(A, V) + D(V, Y);$
16. **else**
17. $D(A, Y) = \min(D(A, Y), D(A, V) + D(V, Y));$
18. **if** (there is a new minimum for destination Y)
19. **send** $D(A, Y)$ to all neighbors
20. **forever**

Elosztott Bellman-Ford algoritmus – példa



A kezdeti vektora		
A-ból	költség	keresztül
B-hez	3	B-n
C-hez	23	C-n
D-hez	∞	-

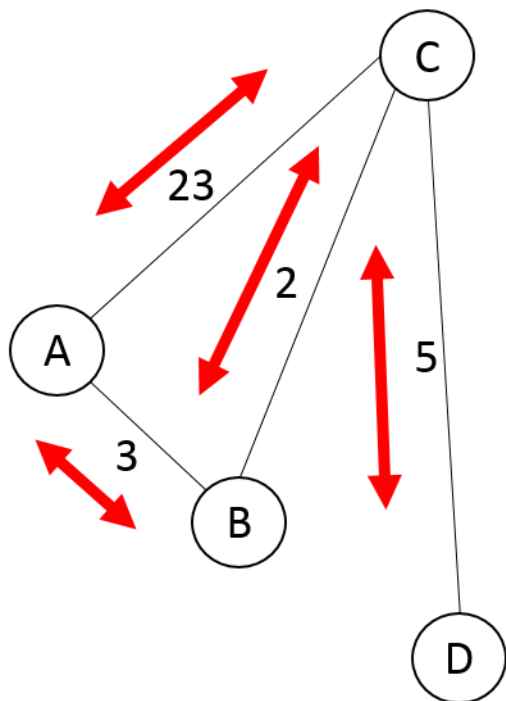
B kezdeti vektora		
B-ból	költség	keresztül
A-hoz	3	A-n
C-hez	2	C-n
D-hez	∞	-

C kezdeti vektora		
C-ből	költség	keresztül
A-hoz	23	A-n
B-hez	2	B-n
D-hez	5	D-n

D kezdeti vektora		
D-ből	költség	keresztül
A-hoz	∞	-
B-hez	∞	-
C-hez	5	C-n

- A példa a https://en.wikipedia.org/wiki/Distance-vector_routing_protocol alapján készült.

Elosztott Bellman-Ford algoritmus – példa



A vektora **B** és **C** fogadása után

A -ból	költség	keresztül
B -hez	3	B -n
C -hez	5	B -n
D -hez	28	C -n

B vektora **A** és **C** fogadása után

B -ból	költség	keresztül
A -hoz	3	A -n
C -hez	2	C -n
D -hez	7	C -n

C vektora **A**, **B** és **D** fogadása után

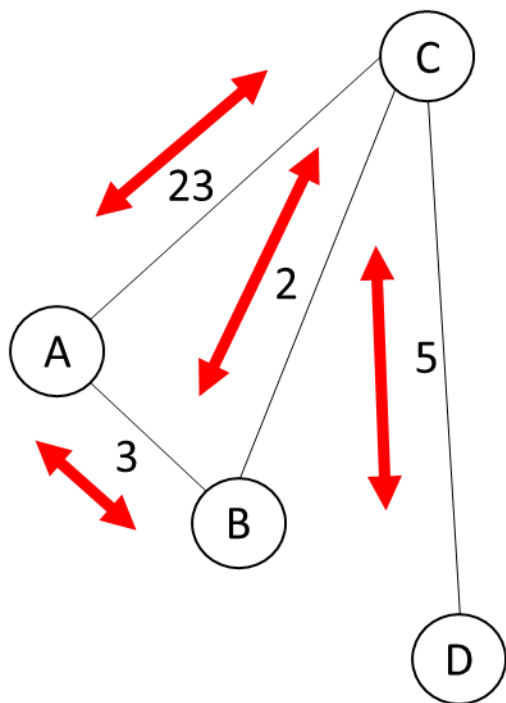
C -ből	költség	keresztül
A -hoz	5	B -n
B -hez	2	B -n
D -hez	5	D -n

D vektora **C** fogadása után

D -ből	költség	keresztül
A -hoz	28	C -n
B -hez	7	C -n
C -hez	5	C -n

- Mivel az összes csúcsnál van új legrövidebb út a kezdeti fázis után, ezért mindegyik csúcs küldeni fogja a vektorát az összes szomszédjának.
- A csúcsok újraszámítják a legrövidebb utakat ezek alapján.
- Zöld háttér → új legrövidebb út

Elosztott Bellman-Ford algoritmus – példa



A vektora **B** és **C** fogadása után

A -ból	költség	keresztül
B -hez	3	B -n
C -hez	5	B -n
D -hez	10	B -n

B vektora **A** és **C** fogadása után

B -ból	költség	keresztül
A -hoz	3	A -n
C -hez	2	C -n
D -hez	7	C -n

C vektora **A**, **B** és **D** fogadása után

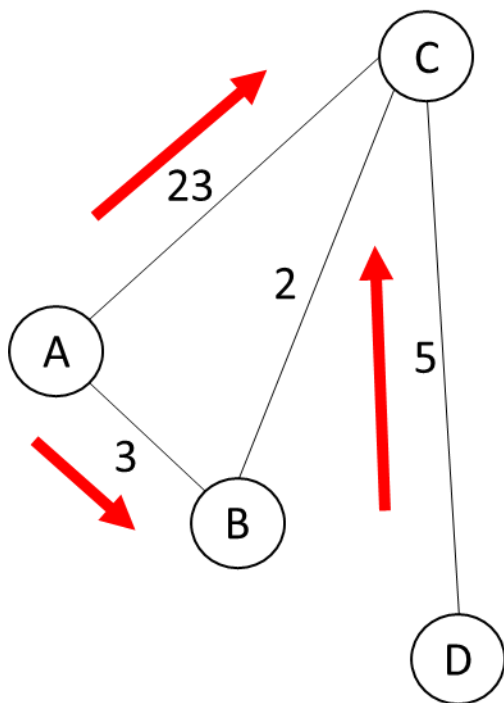
C -ből	költség	keresztül
A -hoz	5	B -n
B -hez	2	B -n
D -hez	5	D -n

D vektora **C** fogadása után

D -ből	költség	keresztül
A -hoz	10	C -n
B -hez	7	C -n
C -hez	5	C -n

- Most viszont már csak **A** és **D** csúcsnak változott a vektora, ezért csak ezek fogják elküldeni vektoraikat szomszédjaiknak.

Elosztott Bellman-Ford algoritmus – példa



A végső vektora		
A-ból	költség	keresztül
B-hez	3	B-n
C-hez	5	B-n
D-hez	10	B-n

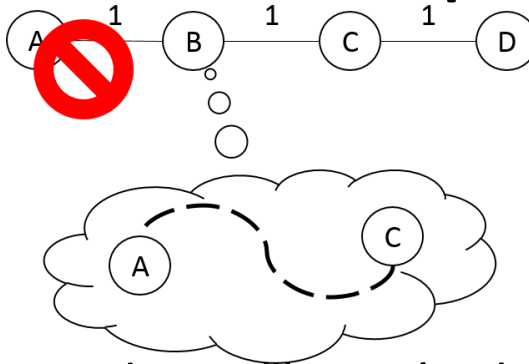
B vektora A fogadása után		
B-ből	költség	keresztül
A-hoz	3	A-n
C-hez	2	C-n
D-hez	7	C-n

C vektora A és D fogadása után		
C-ből	költség	keresztül
A-hoz	5	B-n
B-hez	2	B-n
D-hez	5	D-n

D végső vektora		
D-ből	költség	keresztül
A-hoz	10	C-n
B-hez	7	C-n
C-hez	5	C-n

- Mivel egyik vektor sem hordoz információt legrövidebb utakról, így nem változnak a táblázatok.
- Nincs üzenetszóró csúcs → az algoritmus leáll...
- ... legalábbis amíg meg nem változik valami

Elosztott Bellman-Ford algoritmus – Végtelenig számolás problémája



- Tegyük fel, hogy a fenti rendszerről A csúcs lekapcsolódik.
- B érzékeli, hogy A felé már nincs meg az él
- A probléma, hogy C-től fog kapni egy vektort, amely tévesen azt az információt hordozza, hogy C A-tól 2 távolságra van (C-B-A), **mivel C nem tudja, hogy A nincs már bent.**
- Ezért B frissíti az A-hoz vezető utat $2 + 1$ -gyel a táblázatban, mivel nem tudja, hogy a C-féle 2 költségű úton **ő is rajta van.**
- B-nek változott a vektora, amit C felé továbbít, aki továbbra is úgy tudja, hogy A B-n keresztül elérhető, emiatt C az A-hoz vezető utat $3 + 1$ -re frissíti → ez így fog menni a „végtelenig”

Elosztott Bellman-Ford algoritmus – Végtelenig számolás problémája

- Lehetséges megoldások:
- **„split horizon”**: olyan utakat nem küld vissza a csomópont a szomszédjának, amit tőle „tanult”. (A-ról nem küld információt C B -nek)
- **„split horizon with poison reverse”**: negatív információt küld vissza arról a szomszédjának, amit tőle „tanult”. (C a $D(C,A) = \infty$ információt küldi B -nek, így B -nek nem lesz útvonala A -hoz C keresztül.)

Feladat 1

- Tegyük fel, hogy egy távolság alapú forgalomirányítási protokollban az *A* és *B* routerek távolság vektora az alábbi ábrán található. A protokoll elosztott Bellman-Ford algoritmust használ az útvonalak meghatározására. A költségek szimmetrikusak, azaz minden élen mindkét irányban azonosak.

A vektora			B vektora		
A-ból	költség	keresztül	B-ből	költség	keresztül
B-hez	4	B-n	A-hoz	4	A-n
C-hez	6	C-n	C-hez	10	A-n
D-hez	11	B-n	D-hez	7	D-n
E-hez	10	C-n	E-hez	14	A-n

- Tegyük fel, hogy a csomópontok a „split horizon” szabályt használják a távolságvektorok átadására. Adja meg azt a távolságvektort, amit *B* elküld *A*-nak, miután *E* és *B* közötti közvetlen kapcsolat költsége 5-re változik.
- Adja meg azt a távolságvektort, amit *B* az előző pontban küldene *A*-nak, ha „split horizon with poison reverse” szabályt használna.

Feladat 1 megoldása

- „split horizon” szabály esetén:

B vektora		
B-ből	költség	keresztül
D-hez	7	D-n
E-hez	5	E-n

- „split horizon with poison reverse” szabály esetén:

B vektora		
B-ből	költség	keresztül
A-hoz	∞	-
C-hez	∞	-
D-hez	7	D-n
E-hez	5	E-n

Alhálózati maszk

- Az alhálózat egy logikai felosztása egy IP hálózatnak. Az IP cím ezért két részből áll: hálózatszámából és hoszt azonosítóból.
- A szétválasztás a 32 bites alhálózati maszk segítségével történik, amellyel bitenkénti ÉS-t alkalmazva az IP címre megkapjuk a hálózat-, komplementerével pedig a hoszt azonosítót.
- Ez arra jó, hogy meg tudjuk határozni, hogy a címzett állomás a helyi alhálózaton van-e, vagy sem.
- Az utóbbi esetben az alapértelmezett router felé továbbítják a csomagot (default gateway).

Alhálózati maszk

- CIDR jelölés: kompakt reprezentációja egy IP címnek és a hozzá tartozó hálózatszámnak
- → IP cím + '/' + decimális szám.
- Pl.: 135.46.57.14/24 esetben 135.46.57.14 az IP cím,
- 255.255.255.0 a hálózati maszk (24 db. 1-es bit az elejétől),
- így 135.46.57.0 a hálózat azonosító.

Alhálózati maszk – példa

	10000111	00101110	00111001	00001110	135.46.57.14
AND	11111111	11111111	11111111	00000000	255.255.255.0
<hr/>					
	10000111	00101110	00111001	00000000	135.46.57.0

135.46.57.14/24 → 135.46.57.0

Feladat 3

- Hány cím érhető el a következő alhálózati maszkokkal? Adjuk meg a minimális és maximális címet is!
- 188.100.22.12/32
- 188.100.22.12/20
- 188.100.22.12/10

Feladat 3 megoldása

- 188.100.22.12/32 : egy darab a 188.100.22.12
- 188.100.22.12/20 : $2^{32-20} = 2^{12} = 4096$ darab lenne, de valójában ebből még kettőt le kell vonni, mert speciális jelentéssel bírnak:
 - csupa 0: az alhálózat hálózati címe (magára az alhálózatra vonatkozik)
 - csupa 1-es: broadcast a helyi hálózaton
- 4094 lesz, így a min. cím: 188.100.16.1, a max. cím: 188.100.31.254
- 188.100.22.12/10 : $2^{32-10} - 2 = 4194302$, min. cím: 188.64.0.1, a max. cím: 188.127.255.254.

ifconfig (Linux)

- Hálózati interfészek konfigurálására/lekérdezésére
- Pl.: segítségével megtudhatjuk az IP címünket (inet addr) és MAC címünket (HWaddr):
 - (forrás: <https://en.wikipedia.org/wiki/Ifconfig>)

```
eth0      Link encap:Ethernet HWaddr 00:0F:20:CF:8B:42
          inet addr:217.149.127.10 Bcast:217.149.127.63 Mask:255.255.255.192
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:2472694671 errors:1 dropped:0 overruns:0 frame:0
          TX packets:44641779 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1761467179 (1679.7 Mb) TX bytes:2870928587 (2737.9 Mb)
          Interrupt:28
```

- Ehhez hasonló Windows-nál az ipconfig

További hasznos hálózati segédeszközök

- netstat (Linux és Windows) : kilistázza az aktív hálózati kapcsolatokat
- Pl. kideríthető vele, hogy a programok melyik portokat használják
- tcpdump (Linux) : forgalom figyelő eszköz, a hálózati interfészeiről jövő csomagokat tudja olvasni

Hálózati címfordítás (NAT)

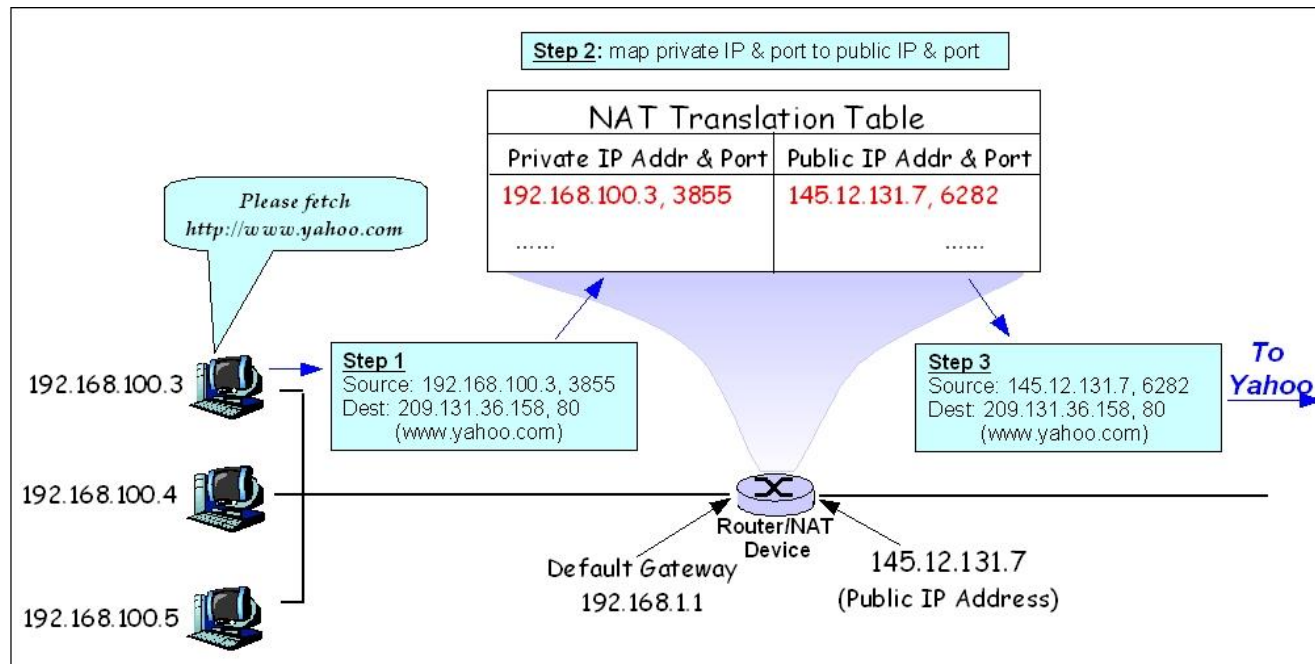
- Gyors javítás az IP címek elfogyásának problémájára.
- Az internet forgalomhoz minden cégnek egy vagy legalábbis kevés IP címet adnak (publikus IP cím(ek))
- A publikus IP cím hozzá van rendelve egy router-hez, a helyi hálózaton (LAN) belül, - amely mögötte van, - minden eszközhöz egy privát IP cím van rendelve
- A privát IP címek csak a LAN-on belül érvényesek (vannak IP cím tartományok erre a célra foglalva)

Hálózati címfordítás (NAT)

- Ha a helyi hálózaton lévő másik géppel akarunk kapcsolatot létesíteni → közvetlenül el tudjuk érni
- Amikor helyi eszkösről akarunk egy külső eszközt elérni, mi történik?
- Szükségünk van port mezők használatára, ami TCP-nél vagy UDP-nél van

Hálózati címfordítás (NAT)

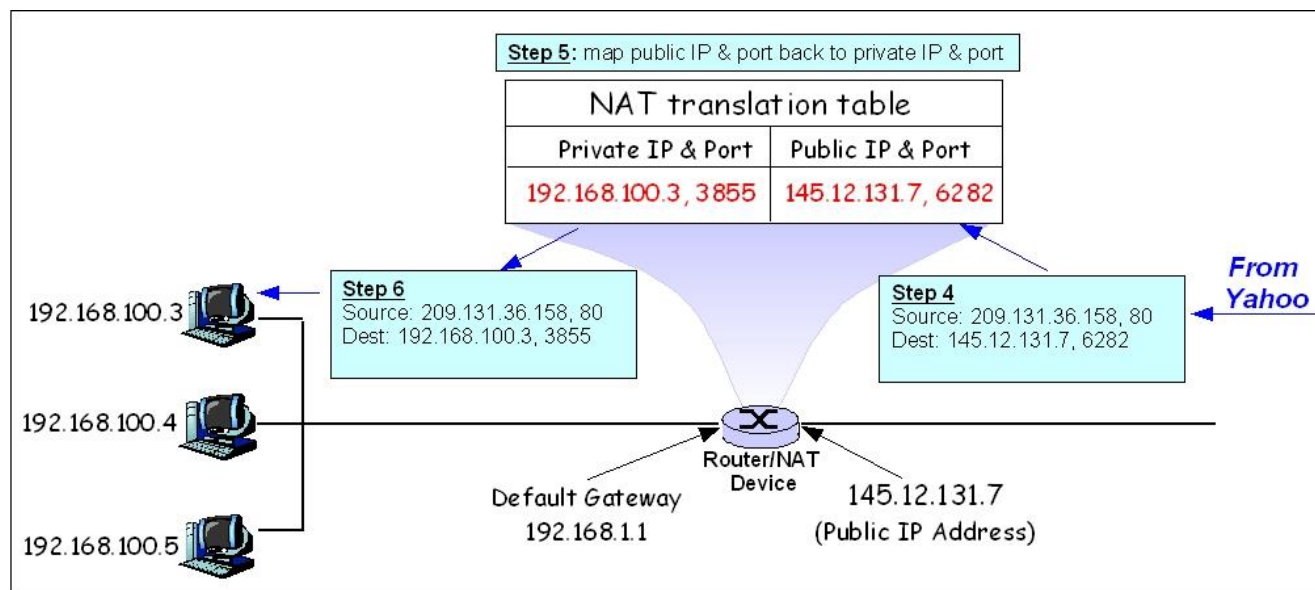
Forrás: https://en.wikibooks.org/wiki/Communication_Networks/NAT_and_PAT_Protocols



- 192.168.100.3 privát IP című gépről HTTP kérés, 3855 porton → Default gateway (192.168.1.1): megnézi a transzlációs tábláját:
 - Ha létezik már a (192.168.100.3, 3855) párhoz (publikus IP cím, port) bejegyzés → lecseréli a küldő forrását arra
 - Ha nincs létrehoz egy új bejegyzést (egyedi lesz!), és azt használja fel a cseréhez

Hálózati címfordítás (NAT)

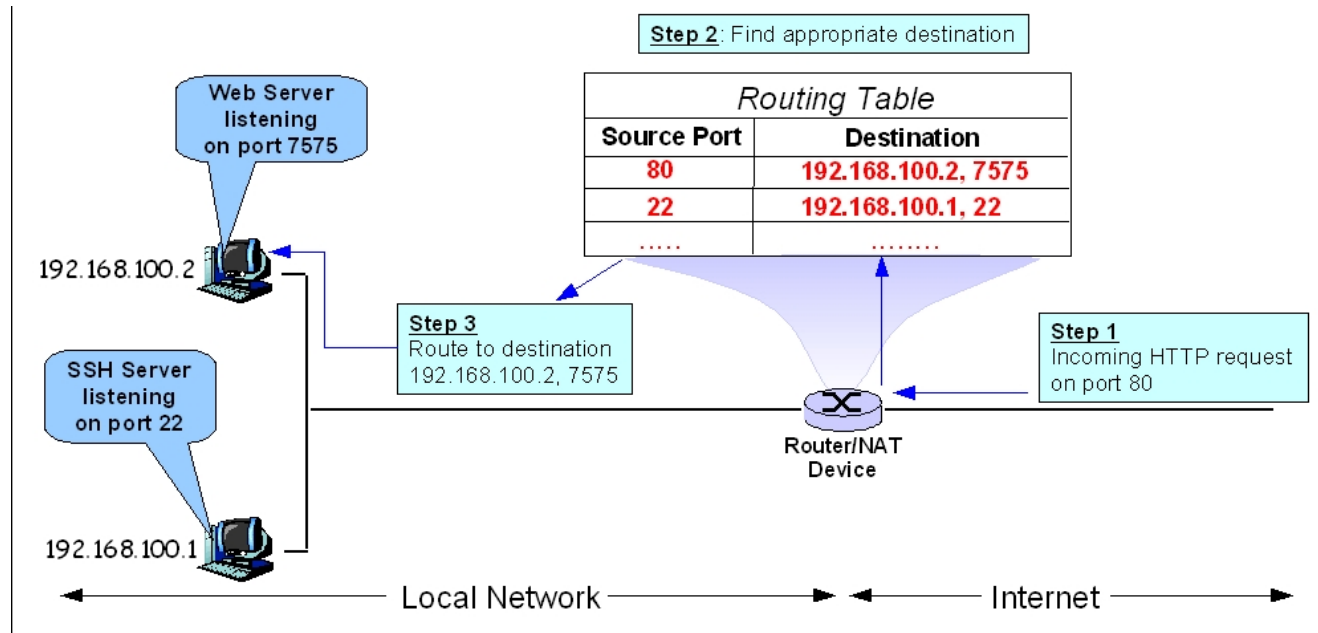
Forrás: https://en.wikibooks.org/wiki/Communication_Networks/NAT_and_PAT_Protocols



- A HTTP válasz a yahoo-tól ugyanúgy a router transzlációs tábláján keresztül megy végbe, csak fordított irányban
- Egy különbség: hiányzó bejegyzés esetén a csomagot eldobja a router

Porttovábbítás (port forwarding)

Forrás: https://en.wikibooks.org/wiki/Communication_Networks/NAT_and_PAT_Protocols



- Az előző példánál a címfordítás transzparens volt (csak a router tudott arról, hogy IP konverzió zajlik). Mit lehet tenni, ha pl. egy belső hálózaton lévő HTTP szervert akarunk elérni kívülről?
- **Porttovábbítás** lehetővé teszi adott lokális hálózaton (LAN) lévő privát IP címek külső elérését egy megadott porton keresztül
- Gyakorlatilag ez a *statikus* NAT alkalmazása

iptables (Linux)

- Egy Linux program csomagszűrési/csomagtovábbítási szabályok, NAT módosítása/lekérdezése
- Például szeretnénk a 192.168.1.10 IP címhez és 80-as porthoz jövő csomagot a 192.168.1.20 IP című géphez küldeni a 80-as portjához, akkor az alábbi parancsok (is) kelleni fognak:

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 192.168.1.20:80
```

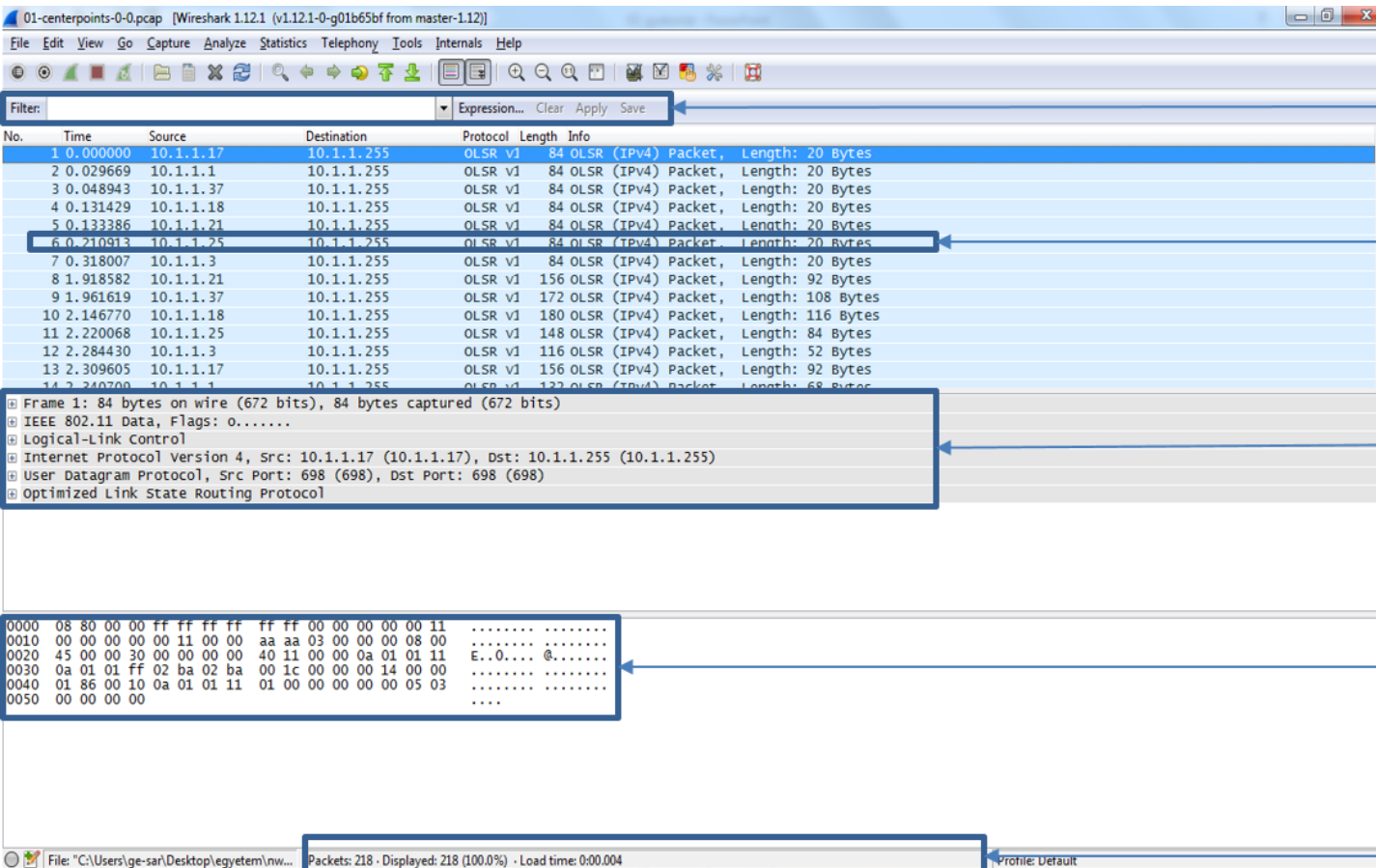
```
iptables -t nat -A POSTROUTING -p tcp -d 192.168.12.20 --dport 80 -j SNAT --to-source 192.168.12.10
```

- (-t kapcsolóval a táblát határozzuk meg, -A PREROUTING : a szabályt a PREROUTING lánc végére szűrja be, -j a csomagcél megadására (SNAT: Source NAT, DNAT: Destination NAT))

Wireshark

- Forgalomelemző eszköz: korábban rögzített adatok elemzésére szolgál
- Windows-on és Linux-on is elérhető
- www.wireshark.org

Wireshark ablakok



Szűrők definiálására
alkalmas input eszközök

Csomag összefoglaló
nézete

Kiválasztott csomag
hierarchikus nézet

Kiválasztott csomag bájtt-
alapú nézet

Szűrés statisztikái

Wireshark szűrők



- Operátorok: or, and, xor, not
- protokollok: ip, tcp, http... (teljes listát lásd → Expression gomb)
- Példa: `tcp.flags.ack==1 and tcp.dstport==80` (tcp nyugta flag és fogadó port beállítva)

Wireshark példa

- A `http_out.pcapng` állomány felhasználásával válaszoljuk meg az alábbi kérdéseket:
 1. Milyen oldalakat kértek le a szűrés alapján HTTP GET metódussal? Milyen böngészőt használtak hozzá?
 2. Hány darab képet érintett a böngészés?
 3. Volt-e olyan kérés, amely titkosított kommunikációt takar?

Wireshark példa megoldás I.

Wireshark network traffic capture showing an HTTP GET request for /Lapok/kezdolap.aspx. The packet list, packet details, and packet bytes panes are visible. The packet details pane shows the Hypertext Transfer Protocol section with fields like Host, Connection, Accept, and User-Agent.

Filter: `http.request.method=="GET"`

No.	Time	Source	Destination	Protocol	Length	Info
25	0.545934	192.168.1.100	173.194.39.104	HTTP	621	GET / HTTP/1.1
28	0.575342	192.168.1.100	173.194.116.143	HTTP	757	GET /?gfe_rd=cr&ei=1l4dVImtCufb8geysIHoCa HTTP/1.1
145	5.585338	192.168.1.100	157.181.161.79	HTTP	412	GET / HTTP/1.1
155	5.863160	192.168.1.100	157.181.161.79	HTTP	485	GET /Lapok/kezdolap.aspx HTTP/1.1
178	6.015329	192.168.1.100	157.181.161.79	HTTP	568	GET /Style%20Library/hu-HU/Core%20Styles/Band.css HTTP/1.1
182	6.016702	192.168.1.100	157.181.161.79	HTTP	571	GET /Style%20Library/hu-HU/Core%20Styles/controls.css HTTP/1.1
183	6.017730	192.168.1.100	157.181.161.79	HTTP	550	GET /Style%20Library/elteik.css HTTP/1.1
184	6.018474	192.168.1.100	157.181.161.79	HTTP	557	GET /_styles/HtmlEditorCustomStyles.css HTTP/1.1
215	6.073612	192.168.1.100	157.181.161.79	HTTP	565	GET /Style%20Library/ik/ikonok/oldalalterkep.png HTTP/1.1
216	6.075335	192.168.1.100	157.181.161.79	HTTP	561	GET /Style%20Library/ik/ikonok/kereses.png HTTP/1.1
217	6.076699	192.168.1.100	157.181.161.79	HTTP	559	GET /Style%20Library/ik/ikonok/email.png HTTP/1.1
221	6.185875	192.168.1.100	157.181.161.79	HTTP	557	GET /Style%20Library/ik/ikonok/rss.png HTTP/1.1
222	6.191369	192.168.1.100	157.181.161.79	HTTP	561	GET /Style%20Library/ik/ikonok/english.png HTTP/1.1
223	6.195866	192.168.1.100	157.181.161.79	HTTP	569	GET /SiteCollectionImages/terkep_reszlet.png HTTP/1.1
224	6.196355	192.168.1.100	157.181.161.79	HTTP	562	GET /SiteCollectionImages/kari_galeria.png HTTP/1.1
225	6.196788	192.168.1.100	157.181.161.79	HTTP	567	GET /Style%20Library/ik/ik_bg_lighter.gif HTTP/1.1

Frame 155: 485 bytes on wire (3880 bits), 485 bytes captured (3880 bits) on interface 0

Ethernet II, Src: HonHaiPr_6d:48:8b (cc:af:78:6d:48:8b), Dst: Tp-LinkT_1c:6b:9a (00:27:19:1c:6b:9a)

Internet Protocol Version 4, Src: 192.168.1.100, Dst: 157.181.161.79

Transmission Control Protocol, Src Port: 36766, Dst Port: 80, Seq: 1, Ack: 1, Len: 431

Hypertext Transfer Protocol

GET /Lapok/kezdolap.aspx HTTP/1.1\r\n

Host: www.inf.elte.hu\r\n

Connection: keep-alive\r\n

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.120 Safari/537.36\r\n

0000 00 27 19 1c 6b 9a cc af 78 6d 48 8b 08 00 45 00 .'.k... xmH...E.
0010 01 d7 0f 0b 40 00 80 06 e9 04 c0 a8 01 64 9d b5@... ..d..
0020 a1 4f 8f 9e 00 50 65 03 aa 42 ea 40 8a 0a 50 18 .O...Pe. .B...P.
0030 44 70 f2 e7 00 00 47 45 54 20 2f 4c 61 70 6f 6b Op...GE T /Lapok
0040 2f 6b 65 7a 64 6f 6c 61 70 2e 61 73 70 78 20 48 /kezdola p.aspx H
0050 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 77 TTP/1.1. .Host: w
0060 77 77 2e 69 6e 66 2e 65 6c 74 65 2e 68 75 0d 0a ww.inf.e lte.hu..
0070 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 6b 65 65 70 Connecti on: keep
0080 2d 61 6c 69 76 65 0d 0a 41 63 63 65 70 74 3a 20 -alive.. Accept:
0090 74 65 78 74 2f 68 74 6d 6c 2c 61 70 70 6c 69 63 text/htm l,applic
00a0 61 74 69 6f 6e 2f 78 68 74 6d 6c 2b 78 6d 6c 2c ation/xh tml+xml,

Frame (frame), 485 bytes

Packets: 1453 · Displayed: 70 (4.8%) · Load time: 0:0.68

Profile: Default

Wireshark példa megoldás I.

- Milyen oldalakat kértek le a szűrés alapján HTTP GET metódussal? Milyen böngészőt használtak hozzá?
- Szűrés: `http.request.method=="GET"`
- User-agent header-ből lehet következtetni a böngésző típusára: User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.120 Safari/537.36
- Ehhez segítségünkre lehet ez a link:
http://www.zytrax.com/tech/web/browser_ids.htm

Wireshark példa megoldás II.

http_out.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Filter: http.accept == "image/webp,*/*;q=0.8"

No.	Time	Source	Destination	Protocol	Length	Info
215	6.073612	192.168.1.100	157.181.161.79	HTTP	565	GET /Style%20Library/ik/ikonok/oldalterkep.png HTTP/1.1
216	6.075335	192.168.1.100	157.181.161.79	HTTP	561	GET /Style%20Library/ik/ikonok/kereses.png HTTP/1.1
217	6.076699	192.168.1.100	157.181.161.79	HTTP	559	GET /Style%20Library/ik/ikonok/email.png HTTP/1.1
221	6.185875	192.168.1.100	157.181.161.79	HTTP	557	GET /Style%20Library/ik/ikonok/rss.png HTTP/1.1

Frame 215: 565 bytes on wire (4520 bits), 565 bytes captured (4520 bits) on interface 0

Ethernet II, Src: HonHaiPr_6d:48:8b (cc:af:78:6d:48:8b), Dst: Tp-LinkT_1c:6b:9a (00:27:19:1c:6b:9a)

Internet Protocol Version 4, Src: 192.168.1.100, Dst: 157.181.161.79

Transmission Control Protocol, Src Port: 36766, Dst Port: 80, Seq: 432, Ack: 33346, Len: 511

Hypertext Transfer Protocol

GET /Style%20Library/ik/ikonok/oldalterkep.png HTTP/1.1\r\n

Host: www.inf.elte.hu\r\n

Connection: keep-alive\r\n

Accept: image/webp,*/*;q=0.8\r\n

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.120 Safari/537.36\r\n

Referer: http://www.inf.elte.hu/Lapok/kezdolap.aspx\r\n

Accept-Encoding: gzip,deflate,sdch\r\n

Accept-Language: hu-HU,hu;q=0.8,en-US;q=0.6,en;q=0.4\r\n

If-None-Match: "{02084E23-E4E5-4B60-A388-F77D8FAD8892},2"\r\n

If-Modified-Since: Wed, 16 Jan 2008 11:58:25 GMT\r\n

0090 3a 20 6b 65 65 70 2d 61 6c 69 76 65 0d 0a 41 63 : keep-a live..Ac

00a0 63 65 70 74 3a 20 69 6d 61 67 65 2f 77 65 62 70 cept: im age/webp

00b0 2c 2a 2f 2a 3b 71 3d 30 2e 38 0d 0a 55 73 65 72 ,*/*;q=0.8..User

00c0 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f -Agent: Mozilla/

00d0 35 2e 30 20 28 57 69 6e 64 6f 77 73 20 4e 54 20 5.0 (Win dows NT

00e0 36 2e 31 3b 20 57 4f 57 36 34 29 20 41 70 70 6c 6.1; WOW 64) Appl

00f0 65 57 65 62 4b 69 74 2f 35 33 37 2e 33 36 20 28 eWebKit/ 537.36 (

0100 4b 48 54 4d 4c 2c 20 6c 69 6b 65 20 47 65 63 6b KHTML, l ike Geck

0110 6f 29 20 43 68 72 6f 6d 65 2f 33 37 2e 30 2e 32 o) Chrom e/37.0.2

0120 30 36 32 2e 31 32 30 20 53 61 66 61 72 69 2f 35 062.120 Safari/5

0130 33 37 2e 33 36 0d 0a 52 65 66 65 72 65 72 3a 20 37.36..R eferer:

0140 68 74 74 70 3a 2f 2f 77 77 72 2e 69 6e 66 2e 65 http://w ww.inf.e

0150 6c 74 65 2e 68 75 2f 4c 61 70 6f 6b 2f 6b 65 7a lte.hu/L apok/kez

0160 64 6f 6c 61 70 2e 61 73 70 78 0d 0a 41 63 63 65 dolap.as px..Acce

0170 70 74 2d 45 6e 63 6f 64 69 6e 67 3a 20 67 7a 69 pt-Encod ing: gzi

0180 70 2c 64 65 66 6c 61 74 65 2c 73 64 63 68 0d 0a p,deflat e,sdch..

0190 41 63 63 65 70 74 2d 4c 61 6e 67 75 61 67 65 3a Accept-L anguage:

01a0 20 68 75 2d 48 55 2c 68 75 3b 71 3d 30 2e 38 2c hu-HU,h u;q=0.8,

HTTP Accept (http.accept), 30 bytes

Packets: 1453 Displayed: 26 (1.8%) Load time: 0:0.63 Profile: Default

Wireshark példa megoldás II.

- Hány darab képet érintett a böngészés?
- Szűrés: `http.accept == "image/webp,*/*;q=0.8"`
- Accept header: a kérésre adott válasz tartalmának elfogadható típusa
- WebP: új, veszteséges tömörítést alkalmazó képformátum, amelyet a Google fejlesztett ki a web hálózati forgalmának csökkentésére

Wireshark példa megoldás III.

http_out.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.dstport==443

No.	Time	Source	Destination	Protocol	Length	Info
4	0.007666	192.168.1.100	173.194.116.143	TCP	66	36763→443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
10	0.043654	192.168.1.100	173.194.116.143	TCP	54	36763→443 [ACK] Seq=1 Ack=1 Win=17160 Len=0
13	0.045430	192.168.1.100	173.194.116.143	TLSv1.2	267	Client Hello
17	0.082913	192.168.1.100	173.194.116.143	TCP	54	36763→443 [ACK] Seq=214 Ack=2861 Win=17160 Len=0
19	0.099481	192.168.1.100	173.194.116.143	TLSv1.2	308	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
22	0.133310	192.168.1.100	173.194.116.143	TCP	54	36763→443 [ACK] Seq=468 Ack=4215 Win=15804 Len=0
24	0.328037	192.168.1.100	173.194.116.143	TCP	54	36763→443 [ACK] Seq=468 Ack=4252 Win=15768 Len=0
31	0.656301	192.168.1.100	173.194.116.143	TLSv1.2	103	Application Data
32	0.656739	192.168.1.100	173.194.116.143	TLSv1.2	91	Application Data
33	0.657355	192.168.1.100	173.194.116.143	TLSv1.2	111	Application Data
34	0.659378	192.168.1.100	173.194.116.143	TLSv1.2	693	Application Data
35	0.671176	192.168.1.100	173.194.39.120	TCP	66	36764→443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
37	0.692679	192.168.1.100	173.194.39.120	TCP	54	36764→443 [ACK] Seq=1 Ack=1 Win=17160 Len=0
39	0.693518	192.168.1.100	173.194.39.120	TLSv1.2	269	Client Hello
43	0.710588	192.168.1.100	173.194.39.120	TCP	54	36764→443 [ACK] Seq=216 Ack=2861 Win=17160 Len=0
45	0.727492	192.168.1.100	173.194.39.120	TLSv1.2	308	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
48	0.745754	192.168.1.100	173.194.39.120	TCP	54	36764→443 [ACK] Seq=470 Ack=4216 Win=15804 Len=0
53	0.794975	192.168.1.100	173.194.116.143	TCP	54	36763→443 [ACK] Seq=1250 Ack=5893 Win=17160 Len=0

Frame 13: 267 bytes on wire (2136 bits), 267 bytes captured (2136 bits) on interface 0

Ethernet II, Src: HonHaiPr_6d:48:8b (cc:af:78:6d:48:8b), Dst: Tp-LinkT_1c:6b:9a (00:27:19:1c:6b:9a)

Internet Protocol Version 4, Src: 192.168.1.100, Dst: 173.194.116.143

Transmission Control Protocol, Src Port: 36763, Dst Port: 443, Seq: 1, Ack: 1, Len: 213

Secure Sockets Layer

0030 10 c2 4c d5 00 00 16 03 01 00 d0 01 00 00 cc 03 ..L... ..
0040 03 79 3e 23 0a c0 08 98 16 d7 3a 0f 4c 50 0b 3d .y>#... .LP.=
0050 35 01 f1 22 2f 12 50 7a ad 5c 7b 13 2d a7 5c cd 5.."/.Pz .\{.-.\.
0060 ea 00 00 28 cc 14 cc 13 c0 2b c0 2f 00 9e c0 0a ...(. ... +./...
0070 c0 09 c0 13 c0 14 c0 07 c0 11 00 33 00 32 00 39 3.2.9
0080 00 9c 00 2f 00 35 00 0a 00 05 00 04 01 00 00 7b .../.5.{
0090 00 00 00 12 00 10 00 00 0d 77 77 77 2e 67 6f 6fwww.goo
00a0 67 6c 65 2e 68 75 ff 01 00 01 00 00 0a 00 08 00 gle.hu... ..
00b0 06 00 17 00 18 00 19 00 0b 00 02 01 00 00 23 00#.,

Secure Sockets Layer (ssl), 213 bytes

Packets: 1453 · Displayed: 428 (29.5%) · Load time: 0:0.67 · Profile: Default

Wireshark példa megoldás III.

- Volt-e olyan kérés, amely titkosított kommunikációt takar?
- Szűrés: `tcp.dstport==443`
- Transport Layer Security (TLS) titkosító protokoll feletti HTTP kommunikációra utal a 443-as port
- Elektronikus levelezéshez, banki szolgáltatásokhoz stb. elengedhetetlen
- Nélküle le lehet hallgatni a kommunikációt, lásd a következő diát (sample3.pcapng felhasználásával)

Wireshark – „leleplezés”

sample3.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/> Expression... +

No.	Time	Source	Destination	Protocol	Length	Info
12	4.106306	192.168.2.101	84.2.36.197	HTTP	131	POST /pages/user/login.jsp?method=Login HTTP/1.1 (application/x-www-form-urlencoded)
13	4.121234	84.2.36.197	192.168.2.101	TCP	60	80->65533 [ACK] Seq=1 Ack=1393 Win=7890 Len=0
14	6.160352	84.2.36.197	192.168.2.101	HTTP	594	HTTP/1.1 302 Moved Temporarily

Internet Protocol Version 4, Src: 192.168.2.101, Dst: 84.2.36.197

Transmission Control Protocol, Src Port: 65533, Dst Port: 80, Seq: 1316, Ack: 1, Len: 77

[2 Reassembled TCP Segments (1392 bytes): #11(1315), #12(77)]

Hypertext Transfer Protocol

POST /pages/user/login.jsp?method=Login HTTP/1.1\r\n

Host: **iiw.hu\r\n**

Connection: keep-alive\r\n

Referer: http://iiw.hu/pages/user/login.jsp\r\n

Content-Length: 77\r\n

Cache-Control: max-age=0\r\n

Origin: http://iiw.hu\r\n

Content-Type: application/x-www-form-urlencoded\r\n

Accept: application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5\r\n

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US) AppleWebKit/534.3 (KHTML, like Gecko) Chrome/6.0.472.59 Safari/534.3\r\n

Accept-Encoding: gzip,deflate,sdch\r\n

Accept-Language: hu-HU,hu;q=0.8,en-US;q=0.6,en;q=0.4\r\n

Accept-Charset: ISO-8859-2,utf-8;q=0.7,*;q=0.3\r\n

[truncated]Cookie: ajaxable=1; forgetEmail=0; email=bGFraXNAaW5mLmVsdGUuaHU\$; httpslogin=0; __utma=114476831.2067882217.1284887869.1284887869.1284887869.1; __utmc=114476831;\r\n

[Full request URI: <http://iiw.hu/pages/user/login.jsp?method=Login>]

[HTTP request 1/1]

[Response in frame: 14]

File Data: 77 bytes

HTML Form URL Encoded: application/x-www-form-urlencoded

Form item: "email" = "kulimasz@perec.hu"

Form item: "password" = "kistraktor53"

0020 24 c5 ff fd 00 50 e6 41 31 0f 9a 6f 83 0a 50 18 \$....P.A 1..o..P.

0030 fa f0 7c ef 00 00 65 6d 61 69 6c 3d 6b 75 6c 69 ..|...em ail=kuli

0040 6d 61 73 7a 25 34 30 70 65 72 65 63 2e 68 75 26 masz%40p erec.hu&

0050 70 61 73 73 77 6f 72 64 3d 6b 69 73 74 72 61 6b password =kistrak

0060 74 6f 72 35 33 26 68 74 74 70 73 6c 6f 67 69 6e tor53&ht tpslogin

Frame (131 bytes) Reassembled TCP (1392 bytes)

sample3 Packets: 381 · Displayed: 381 (100.0%) · Load time: 0:0.16 Profile: Default

Emlékeztető (fizikai réteg)

A fogadónak szinkronizálva kell lennie a küldővel. Főprobléma: az óra eltolódás.

Megoldások:

1. Egy **explicit órajel** adása:

- párhuzamos átviteli csatornák használata,
- szinkronizált adatok,
- rövid átvitel esetén alkalmas.

2. A fogadó szinkronizálás **kritikus időpontokban**:

- szinkronizáljunk például egy szimbólum vagy blokk kezdetén,
- a kritikus időpontokon kívül szabadon futnak az órák,
- az órajel generátorok rövidtávú stabilitásán nyugszik ez a megközelítés (feltesszük, hogy nem divergálnak el egymástól nagyon gyorsan)

3. **Önütemező jel**:

- külön órajel szinkronizáció nélkül dekódolható jel,
- a (kódolt) adatjel tartalmazzon elég információt úgy, hogy a fogadó azonnal tudja, amikor egy bit indul/befejeződik

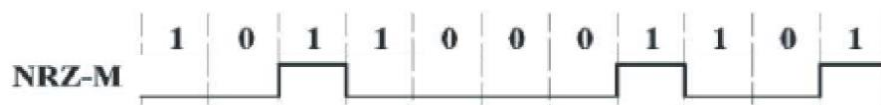
Bináris kódolások I.

- Non-Return to Zero-Level (NRZ-L)

➤ 1: magas feszültség, 0: alacsony (nem önütemező: pl. 0111111110)

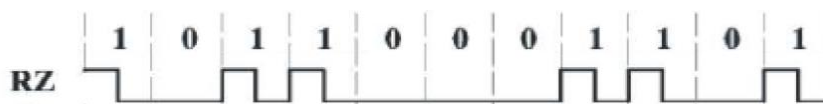


- Non-Return to Zero-Mark (NRZ-M)



➤ 1: váltás az intervallum elején, 0: nincs váltás (nem önütemező)

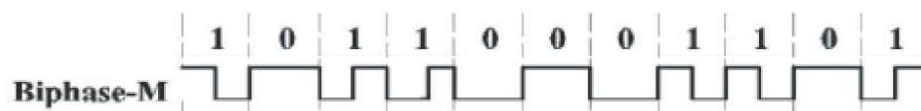
- Return to Zero (RZ)



➤ 1: magas feszültség → alacsony, 0: alacsony (nincs váltás), (nem önü.)

Bináris kódolások II.

- Biphase-Mark



- minden intervallum elején váltás
- 1: még egy váltás az intervallum közepén, 0: nincs váltás az intervallum közepén (önütemező)

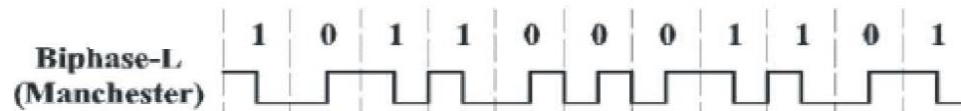
- Biphase-Space



- minden intervallum elején váltás
- minden intervallum elején váltás 1: nincs váltás az intervallum közepén, 0: még egy váltás az intervallum közepén (önütemező)

Bináris kódolások III.

- Manchester (Biphase-L)



- 1: magas feszültség → alacsony, 0: alacsony feszültség → magas
 - önütemező: ha egy minta párban megegyeznek a feszültség jelszintek, akkor azt eldobjuk (a digitális szignál folyamatosan ismétlődik)
 - dekódolás: ha a mintapárban magas feszültségről alacsonyra vált, akkor 1, ha alacsonyról magasra, akkor 0
- Mit lát a fogadó oldal, ha egy hosszú szünet (0-s jelszint) után az 10 bitsorozatot próbáljuk átvinni Manchester kódolással?
 - Hogyan orvosolható ez a probléma?

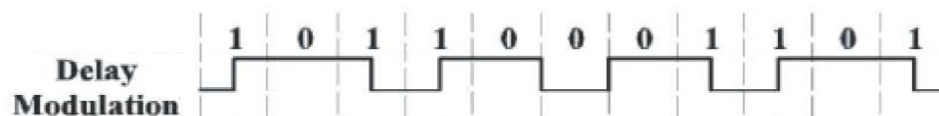
Bináris kódolások IV.

- Differential Manchester



- minden intervallum közepén váltás
- 1: nincs váltás az intervallum elején, 0: váltás az intervallum elején (önütemező)

- Delay Modulation (Miller)



- 1: váltás az intervallum közepén, 0: váltás az intervallum végén, ha 0 következik, nincs váltás, ha 1 következik (önütemező)

Órai feladat (4 pont)

- Készíts olyan server-kliens socket alkalmazást, ahol a kliens elküld egy bitsorozatot (max 16) és egy kódolást ('NRZ-L','RZ','Manchester','DiffManchester'), amire a szerver kiszámolja a kódolt jelerősséget! A jelerősség párok: az intervallum eleje és vége.
- pl:
 - kliens küldi: („NRZ-L”, 1100)
 - server válasza: (1,1),(1,1),(0,0),(0,0)
- segítség:
 - x00-k eltüntetése egy string végéről:

```
szoveg.rstrip('\x00')
```

```
python haziFizikaiClient.py DiffManchester 110011
```

Coding: DiffManchester

Input: 110011

Output: (01),(10),(10),(10),(01),(10),

```
python haziFizikaiClient.py Manchester 110011
```

Coding: Manchester

Input: 110011

Output: (10),(10),(01),(01),(10),(10),

```
python haziFizikaiClient.py NRZ-L 110011
```

Coding: NRZ-L

Input: 110011

Output: (11),(11),(00),(00),(11),(11),

```
python haziFizikaiClient.py RZ 110011
```

Coding: RZ

Input: 110011

Output: (10),(10),(00),(00),(10),(10),

VÉGE
KÖSZÖNÖM A FIGYELMET!