**ESERCIZI CAPITOLO 2 – LIVELLO APPLICAZIONE**

**ESERCIZIO 1**

1. gaia.cs.umass.edu/cds453/index.html
2. HTTP/1.1
3. Conessione persistente (300s)
4. Non presente nei messaggi HTTP
5. Messaggio GET, lo user-agent puo’ essere utile al server per inviare al client il file in un formato particolare, meglio supportato;

**ESERCIZIO 2**



GET index.html HTTP/1.1

Host: [www.unisa.it](http://www.unisa.it)

User-Agent: Mozilla/5.0 (Windows)

Accept: text/html, text/plain

Accept-Language: it-IT

Keep-Alive: 300

Connection: keep-alive

HTTP/1.1 200 OK

Date: Fri, 12 Apr 2019, 12:30GMT

Server: Apache/2.0.52 (Fedora)

Last Modified: Mon, 08 Apr 2019, 15:30GMT

Keep-Alive: timeout=max=100

Connection: keep-alive

Content-Type text/hmtl, charset=ISO-8859-1

HTTP/1.1 404 NOT FOUND

Date: Fri, 12 Apr 2019, 12:30GMT

Server: Apache/2.0.52 (Fedora)

Keep-Alive: timeout=max=100

Connection: keep-alive

HTTP/1.1 202 OK

Date: Fri, 12 Apr 2019, 12:30GMT

Server: Apache/2.0.52 (Fedora)

Connection: close

Last Modified: Mon, 08 Apr 2019, 15:30GMT

Content-Type text/hmtl, charset=ISO-8859-1

GET index.html HTTP/1.1

Host: [www.unisa.it](http://www.unisa.it)

User-Agent: Mozilla/5.0 (Windows)

Accept: text/html, text/plain

Accept-Language: it-IT

Keep-Alive: 300

Connection: keep-alive

If-Modified-Since: Mon, 08 Apr 2019, 15:30GMT

HTTP/1.1 304 NOT MODIFIED

Date: Fri, 12 Apr 2019, 12:30GMT

Server: Apache/2.0.52 (Fedora)

Keep-Alive: timeout=max=100

Connection: keep-alive

**ESERCIZIO 3**

|  |  |
| --- | --- |
| T1: NULL | T4: 2fhfw\* |
| T2: 2fhfw\* | T5: 2fhfw\*, 54h22\* |
| T3: 2fhfw\* |  |

1. **T1**

GET index.html HTTP/1.1

Host: unisa.it



HTTP/1.1 200 OK

Set-cookie: 2fhfw\*

1. **T2, T3**

GET new\_obj.html HTTP/1.1

Host: unisa.it

Cookie: 2fhfw\*



HTTP/1.1 200 OK

1. **T4**

GET index.html HTTP/1.1

Host: netflix.com



HTTP/1.1 200 OK

Set-cookie: 54h22\*

1. **T5**

GET index.html HTTP/1.1

Host: netflix.com

Cookie: 54h22\*



HTTP/1.1 200 OK

**ESERCIZIO 4**

**Cache proxy inizialmente vuota.**

1. **A**: **GET ordinaria**, il proxy inoltra la richiesta ad S, il quale risponde con l’oggetto richiesto a P lo salva il cache;
2. **B**: **GET ordinaria** (host diverso dalla GET precedente), il proxy inoltra a richiesta ad S, il quale risponde con l’oggetto richiesto a P lo salva il cache;
3. **A**: **GET condizionale**: il proxy risponde con la sua copia locale;
4. **C**: **GET ordinaria**... Non connesso al proxy, C contatta direttamente il server;
5. **C**: **GET ordinaria**... Non connesso al proxy, C contatta direttamente il server;
6. **B**: **GET ordinaria**... Nessun host precedente ha richiesto l’oggetto “login.html” ad ebay... Il proxy inoltra la richiesta ad S, il quale risponde con l’oggetto richiesto e lo salva in cache;
7. **B**: **GET condizionale**, il proxy risponde con la copia locale;
8. **A:** **GET condizionale**, il proxy risponde con la copia locale;

**ESERCIZIO 5**

**Cache proxy inizialmente vuota.**

1. **A**: **GET ordinaria**... Il proxy P inoltra la richiesta al server S, il quale risponde con l’oggetto richiesto e il proxy ne salva una copia;
2. **B**: **GET condizionale**... Proxy P risponde con copia locale;
3. **A**: **GET ordinaria**... Il proxy P inoltra la richiesta al server S, il quale risponde con l’oggetto richiesto e il proxy ne salva una copia;
4. **C**: **GET ordinaria**... Il proxy Q inoltra la richiesta al server S, il quale risponde con l’oggetto richiesto e il proxy ne salva una copia;
5. **D**: **GET ordinaria**... Il proxy Q inoltra la richiesta al server S, il quale risponde con l’oggetto richiesto e il proxy ne salva una copia;
6. **B**: **GET ordinaria**... Il proxy P inoltra la richiesta al server S, il quale risponde con l’oggetto richiesto e il proxy ne salva una copia;
7. **C**: **GET ordinaria**... Il proxy Q inoltra la richiesta al server S, il quale risponde con l’oggetto richiesto e il proxy ne salva una copia;
8. **A**: **GET ordinaria**... Il proxy P inoltra la richiesta al server S, il quale risponde con l’oggetto richiesto e il proxy ne salva una copia;
9. **B**: **GET condizionale**... Proxy P risponde con copia locale;

HTTP/1.1 **200 OK**

Date: **Tue 10 Apr 2022 14:30:45GMT**

Server: Apache/2.0.52 (Fedora)

Last Modified: **Mon 09 Apr 2022 14:30:45GMT**

Keep-Alive: timeout=max=100

Connection: keep-alive

Content-Type: text/html; charset=ISO-8859-1

**ESERCIZIO 10**

* + 220 unisa.it, **SERVER**
  + HELO unisa.it, **CLIENT**
  + 250 Stop with greetings, and let us do our job, **CLIENT**

C: MAIL FROM: <dferraioli@unisasa.it>

**S: 220** [**dferraioli@unisa.it**](http://dferraioli@unisa.it) **... Sender ok**

C: RCPT TO: <auletta@unisa.it>

S: 250 auletta@unisa.it ... Recipient ok

C: DATA

S: 354 Please enter the email content

C: Ho caricato su Google Drive

C: la traccia dell’esame

**C: .**

**S: 250 message accepted for delivery**

C: QUIT

S: 221 unisa.it closing connection



**From: dferraioli@unisasa.it**

**To: auletta@unisa.it**

Subject: esame Reti

Date: Fri, 17 Mar 2023 10:28:25 +0100

Mime-Version: 1.0

**Content-Type: text/plain**

Ho caricato su Google Drive la traccia dell’esame.

**ESERCIZI CAPITOLO 3 – LIVELLO DI TRASPORTO**

**ESERCIZIO 8**

Il controllo di flusso implementato da TCP serve sostanzialmente a prevenire il buffer-overflow del destinatario ed evitare quindi perdita di informazione. Ciò si verifica tenendo costantemente monitorati la velocità di invio del mittente e il tasso di lettura del destinatario. Dato che i buffer sono limitati, se il mittente invia più velocemente del tasso di lettura del destinatario, prima o poi si verificherà overflow.

Tale confronto si base sulle variabili allocate da TCP, in particolar modo (finestra di ricezione) che segnala al mittente lo spazio disponibile nel buffer del destinatario. Deve essere costantemente verificata la disuguaglianza:

Il numero di byte privi di acknowledgment deve essere sempre inferiore allo spazio libero nel buffer del ricevente. Quando , il mittente è costretto ad interrompere l’invio ed aspettare che il buffer si svuoti.

In questo esempio, dato che la velocità di invio dell’host A (120Mbps, limitati a 100Mbps dal canale di comunicazione usato) è maggiore di quella di lettura dell’host B (50Mbps), prima o poi si verificherà overflow nel buffer del destinatario, se TCP non implementasse questo controllo.

Quando il buffer sarà pieno, rwnd sarà pari a 0 e il mittente è costretto a fermare l’invio di dati finchè non si libera spazio nel buffer del destinatario.

**ESERCIZIO 9**

La ritrasmissione veloce consente ad un mittente TCP di re-inviare un pacchetto mancante, senza dover aspettare la scadenza del timer, ossia il verificarsi di un evento di timeout. TCP, per definizione, effettua ritrasmissione rapida nel momento in cui rileva 3 ACK duplicati. Rilevati 3 ACK duplicati, il mittente invia immediatamente il segmento mancante.

Sono necessari 3 ACK duplicati per una questione di sicurezza ed affidabilità, per “essere certi che il pacchetto sia andato effettivamente perso”.

Può capitare per esempio che il segmento subisca semplicemente un leggero ritardo e che quindi arrivi prima un pacchetto con un numero di sequenza maggiore di quello atteso (in questa situazione TCP invia ACK duplicato)... In tale frangente basta attendere “un altro po” affinchè il segmento in ritardo possa arrivare a destinazione e pertanto una ritrasmissione sarebbe del tutto inutile.

**ESERCIZI CAPITOLO 4 – LIVELLO DI RETE**

**ESERCIZIO 9**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Num.** | **IP** | **IP - bin** | **Netmask** | **Next-Hop** |
| 1 | 0.0.0.0 | 00000000.00000000.00000000.00000000 | 0 | A |
| 2 | 40.0.0.0 | 00101000.00000000.00000000.00000000 | 6 | B |
| 3 | 42.0.0.0 | 00101010.00000000.00000000.00000000 | 7 | C |
| 4 | 42.128.0.0 | 00101010.10000000.00000000.00000000 | 10 | D |
| 5 | 42.32.0.0 | 00101010.00100000.00000000.00000000 | 12 | E |
| 6 | 40.48.0.0 | 00101000.00110000.00000000.00000000 | 12 | F |
| 7 | 42.32.128.0 | 00101010.00100000.10000000.00000000 | 19 | G |
| 8 | 42.160.64.0 | 00101010.10100000.01000000.00000000 | 20 | H |
| 9 | 42.168.128.0 | 00101010.10101000.10000000.00000000 | 22 | I |
| 10 | 42.168.224.0 | 00101010.10101000.11100000.00000000 | 26 | J |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PKG** | **IP** | **IP - bin** | **Rotte valide** | **Next - Hop** |
| 1 | 42.160.128.64 | 00101010.10100000.10000000.01000000 | 1, 2, 3, 4 | D |
| 2 | 40.61.1.24 | 00101000.00111101.00000001.00011000 | 1, 2, 6 | F |
| 3 | 42.0.6.32 | 00101010.00000000.00000110.00100000 | 1, 2, 3 | C |
| 4 | 40.168.1.80 | 00101000.10101000.00000001.01010000 | 1, 2 | B |
| 5 | 42.32.7.4 | 00101010.00100000.00000111.00000100 | 1, 2, 3, 5 | E |
| 6 | 41.32.27.43 | 00101001.10100000.00011011.00101011 | 1, 2 | B |
| 7 | 42.48.128.7 | 00101010.00110000.10000000.00000111 | 1, 2, 3 | C |
| 8 | 42.32.156.0 | 00101010.00100000.10011100.00000000 | 1, 2, 3, 5, 7 | G |
| 9 | 40.168.130.24 | 00101000.10101000.10000010.00011000 | 1, 2 | B |
| 10 | 44.48.0.32 | 00101100.00110000.00000000.00100000 | 1 | A |

**ESERCIZIO 10**

* **ETH0 = 192.170.123.4**
* **ETH1 = 192.170.124.4**

Forwarding table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#Entry** | **IP** | **IP - bin** | **Netmask** | **Next-Hop** |
| 1 | 192.170.123 | 11000000.10101010.01111011.00000000 | 255.255.255.0 | 0.0.0.0 |
| 2 | 192.170.124.0 | 11000000.10101010.01111100.00000000 | 255.255.255.0 | 0.0.0.0 |
| 3 | 191.138.112.0 | 10111111.10001010.01110000.00000000 | 255.255.240.0 | 192.170.123.1 |
| 4 | 191.138.96.0 | 10111111.10001010.01100000.00000000 | 255.255.224.0 | 192.170.124.1 |
| 5 | 191.138.0.0 | 10111111.10001010.00000000.00000000 | 255.255.0.0 | 192.170.123.2 |
| 6 | 191.138.160.0 | 10111111.10001010.10100000.00000000 | 255.255.224.0 | 192.170.124.2 |
| 7 | 0.0.0.0 | 00000000.00000000.00000000.00000000 | 0.0.0.0 | 192.170.123.3 |

**IP1: 192.138.163.13 -** 10111111.10001010.10100011.00000000

Matches: 5, 6

Inoltro indiretto tramite forwarding table: entry: 6, next-hop: 192.170.124.2

**IP2: 191.138.113.32 -** 10111111.10001010.01110001.00100000

Matches: 3, 4, 5, 7

Inoltro indiretto tramite forwarding table: entry: 3, next-hop: 192.170.123.1

**IP3: 131.175.123.244 -** 10000011.10101111.01111011.11110100

Matches: 7

Inoltro indiretto tramite forwarding table: entry: 7, next-hop: 192.170.123.3

**IP4: 255.255.255.255 -**

Matches: 7

I router per default scartano le richieste broadcast per evitare una concatenazione di richieste che avrebbe effetto in tutte le sottoreti. Le richieste broadcast pertanto sono limitate alla sottorete dell’host che effettua la richiesta pertanto non si ha nessun inoltro.

**IP5: 192.170.123.255 da ETH1 - 11000000.10101010.01111011.11111111**

Matches: 1

Indirizzo broadcast della subnet incidente su ETH0 **192.170.123.x.** Il pacchetto viene inoltrato su ETH0 e successivamente in broadcast a tutta la subnet.

**ESERCIZIO 12**

* **ETH0 = 125.170.123.76, MTU = 500B**
* **ETH1 = 125.174.124.169, MTU = 600B**

Forwarding table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Dest.** | **IP dest.** | **IP - bin** | **Netmask** | **Next-Hop** | **Int** |
| A | 125.170.123.0 | 01111101.10101010.01111011.00000000 | 255.255.255.128 | 0.0.0.0 | ETH0 |
| B | 131.170.123.128 | 10000011.10101010.01111011.10000000 | 255.255.255.128 | 0.0.0.0 | ETH1 |
| C | 125.170.122.128 | 01111101.10101010.01111010.10000000 | 255.255.255.128 | 125.174.124.254 | ETH1 |
| D | 125.170.122.0 | 01111101.10101010.01111010.00000000 | 255.255.254.0 | 125.170.123.2 | ETH0 |
| E | 125.174.124.0 | 01111101.10101110.01111100.00000000 | 255.255.255.128 | 125.170.123.2 | ETH0 |
| F | 125.174.122.0 | 01111101.10101110.01111010.00000000 | 255.255.254.0 | 125.170.123.3 | ETH0 |
| G | 0.0.0.0 | 00000000.00000000.00000000.00000000 | 0.0.0.0 | 125.174.124.253 | ETH1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#Entry** | **IP dest.** | **IP - bin** | **Dimensione** | **Flag DNF** | **TTL** |
| 1 | 125.170.123.127 | 01111101.10101010.01111011.01111111 | 600B | 1 | 18 |
| 2 | 125.174.123.12 | 01111101.10101110.01111011.00001100 | 500B | 1 | 2 |
| 3 | 125.174.123.6 | 01111101.10101110.01111011.00000110 | 400B | 1 | 16 |
| 4 | 125.170.122.66 | 01111101.10101010.01111010.01000010 | 700B | 0 | 16 |
| 5 | 128.174.124.136 | 10000000.10101110.01111100.10001000 | 400B | 0 | 1 |

1. **IP1:**

Matches: **A**, D

Inoltro diretto sull’interfaccia ETH0 del router. Entry A.

Il router decrementa il TTL che diventa 17. Il pacchetto e’ valido.

Il pacchetto viene scartato in quanto flag DNF = 1 e MTU IP1 = 600 > MTU ETH0. Il router quindi scarta il pacchetto ed invia un messaggio di errore tramite ICMP.

1. **IP2:**

**Matches: F**

TTL = 1, pacchetto ancora valido per l’inoltro. MTU coincide... Pacchetto inoltrato dal router.

Inoltro indiretto: entry F, next-hop 125.170.123.3, Interfaccia ETH0.

1. **IP3**

**Matches: F**

TTL = 15, pacchetto ancora valido per l’inoltro. MTU IP3 < MTU ETH0...

Inoltro indiretto: entry F, next-hop 125.170.123.3, Interfaccia ETH0.

1. **IP4**

**Matches: D**

**TTL = 15,** pacchetto ancora valido per l’inoltro. MTU IP4 < MTU ETH0 ma flag DNF = 0. Il router frammenta il pacchetto in 2 pacchetti di dimensioni 500B e 320B e lo inoltra.

Inoltro indiretto: entry F, next-hop 125.170.123.2, Interfaccia ETH0.

1. **IP5**

**Matches: G**

TTL = 0, pacchetto non valido per l’inoltro! Il router scarta direttamente il pacchetto ed invia un messaggio di errore ICMP.

**ESERCIZIO 14**

* **ETH0: 192.168.1.254 (locale)**
* **ETH1: 131.175.22.13**
* **ETH2: 123.17.4.5**
* **ETH3: 17.7.4.27**

Forwarding table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Dest.** | **IP dest.** | **IP - bin** | **Netmask** | **Next-Hop** | **Int** |
| A | 192.168.1.0 | 11000000.10101000.00000001.00000000 | 255.255.255.0 | 0.0.0.0 | ETH0 |
| B | 131.175.23.0 | 10000011.10101111.00010111.00000000 | 255.255.255.128 | 0.0.0.0 | ETH1 |
| C | 123.17.4.0 | 01111011.00010001.00000100.00000000 | 255.255.255.0 | 0.0.0.0 | ETH2 |
| D | 17.7.4.0 | 00010001.00000111.00000100.00000000 | 255.255.255.128 | 0.0.0.0 | ETH3 |
| E | 13.14.190.0 | 00001101.00001110.10111110.00000000 | 255.255.255.128 | 123.17.4.34 | ETH2 |
| F | 12.13.0.0 | 00001100.00001101.00000000.00000000 | 255.255.128.0 | 131.175.23.27 | ETH1 |
| G | 12.13.192.0 | 00001100.00001101.11000000.00000000 | 255.255.192.0 | 123.17.4.34 | ETH2 |
| H | 0.0.0.0 | 00000000.00000000.00000000.00000000 | 0.0.0.0 | 17.7.4.93 | ETH3 |

Pacchetti da inoltrare

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#Entry** | **IP sorg.** | **Porta sorg.** | **IP dest.** | **IP dest - bin** | **Porta dest.** | **Interfaccia** |
| 1 | 192.168.1.5 | 2345 | 192.168.1.8 | 11000000.10101000.00000001.00001000 | 2346 | ETH0 |
| 2 | 192.168.1.6 | 4356 | 12.13.205.7 | 00001100.00001101.11001101.00000111 | 1234 | ETH0 |
| 3 | 137.12.5.3 | 1234 | 12.13.129.11 | 00001100.00001101.10000001.00001011 | 80 | ETH2 |
| 4 | 137.15.7.2 | 2345 | 123.17.4.5 | 01111011.00010001.00000100.00000101 | 80 | ETH2 |
| 5 | 137.23.8.1 | 25 | 123.17.4.7 | 01111011.00010001.00000100.00000111 | 1026 | ETH1 |
| 6 | 192.168.1.17 | 115 | 131.175.23.195 | 10000011.10101111.00010111.11000011 | 6534 | ETH0 |

**NAT: ETH0 192.168.1.254 ETH2 123.17.4.5**

**PORT FORWARDING: ETH2 123.17.4.5:80 192.168.1.3:80 ETH0**

1. **IP1: SRC: 192.168.1.5:2345 , DST: 192.168.1.8:2346, ETH0 – ETH0**

Matches: **A**

Inoltro diretto all’interfaccia ETH0. I due host appartengono alla stessa sottorete, il passaggio per il router non e’ necessario e pertanto non agisce NAT.

1. **IP2: SRC: 192.168.1.6: 4356 , DST: 12.13.205.7:1234, ETH0 – ETH2**

Matches: **H**

Inoltro indiretto al next-hop 17.7.4.93 tramite l’interfaccia ETH0. Presente NAT!

SRC: 192.168.1.6:4356 123.17.4.5: X

1. **IP3: SRC: 137.12.5.3: 1234 , DST: 12.13.129.11: 80, ETH2 – ETH3**

Matches H

Inoltro indiretto al next-hop: 17.7.4.93 tramite interfaccia ETH3.

1. **IP4: SRC: 137.15.7.2: 2345 , DST: 123.17.4.5:80, ETH2 – ETH2**

Matches: C

Inoltro nella stessa sottorete, passaggio per il router non necessario. Inoltro diretto all’interfaccia ETH2.

DST: 123.17.4.5:80 192.168.1.3:80 ETH0 port forwarding

1. **IP5: SRC: 137.23.8.1: 25 , DST: 123.17.4.7:1026, ETH1 – ETH2**

Matches: C

Inoltro diretto all’interfaccia ETH2.

1. **IP6: SRC: 192.168.1.17:115 , 131.175.23.195:6534, ETH0 – ETH3**

Matches: H

Inoltro indiretto al next-hop 17.7.4.93 tramite l’interfaccia ETH3. Presente NAT!

SRC: 192.168.1.17:115 123.17.4.5: X

**ESERCIZI CAPITOLO 6 – LIVELLO DI COLLEGAMENTO**

**ESERCIZIO 12**

**Frame inviati sulla rete:**

* F1: A-broadcast
* F2: H-A
* F3: A-H

Tabelle di commutazione gia’ compilate.

1. **INDICARE COME VIENE INOLTRATO CIASCUN FRAME:**
   1. Il primo frame viene inviato in broadcast a tutti i nodi della rete. Gli hub e gli switch si comportano allo stesso modo: inoltrano il nodo a tutte le interfacce tranne quella che riceve il pacchetto in ingresso ovvero l’interfaccia collegata al mittente. Il frame F1 parte quindi da A, raggiunge l’HUB H1 che lo inoltra in broadcast sulle interfacce 2 e 3. Raggiunge S1 che lo inoltra su 3 e 2 verso S4 ed H2. Da H2 il frame raggiunge gli host direttamente connessi tramite le interfacce 1 e 2... Su S4, F1, viene inoltrato ad S2 ed S3 tramite le interfacce 2 ed 1. S2 ed S3 inoltrano il frame agli host connessi.
   2. F2 parte da H, raggiunge S3 in broadcast il quale (considerando che le tabelle di commutazioni sono gia’ totalmente compilate e le entry tutte valide) lo inoltra verso S4 tramite l’interfaccia 3. In questo caso G non riceve il frame! S4 inoltra il frame verso S1 e successivamente verso H1. H1 trattandosi di un hub, sfrutta la comunicazione in broadcast, inoltrando il frame di H verso A e verso B. B scarta il frame in quanto non diretto a lui, A accetta il frame.

Quindi: H – S3 – S4 – S1 – H1 - A/B

* 1. F3 parte da A, raggiunge l’hub H1 il quale inoltra il frame verso B e verso S1. B scarta il frame. S1 lo inoltra ad S4, S4 lo inoltra ad S3 ed S3 lo consegna ad H.

Quindi: A – H1 - B/S1 – S4 – S3 - H

1. **INDICARE GLI INDIRIZZI SORGENTE E DESTINAZIONE DEI PKGs CHE VIAGGIANO TRA S1 ED S4**
   1. Switch e hub non modificano il contenuto dei frame inoltrati quindi le intestazioni rimangono le stesse di quelle inserite dal mittente. Dato che le tabelle di commutazione sono gia’ compilate, l’host A gia’ conosce l’indirizzo IP dell’host e pertanto lo puo’ utilizzare direttamente senza dover effettuare alcuna richiesta ARP. Essendo gli switch trasparenti a livello di rete e, essendo presente una sola sotterete, gli indirizzi mittente e destinazione rimangono invariati.
2. **QUALI PACCHETTI SONO RICEVUTI DAI NODI B e G?**
   1. B riceve i frames F1, F2, F3, tutti i frame.
   2. G riceve solo F1 trasmesso in broadcast

**ESERCIZIO 13**

2 ROUTER, 1 SWITCH. 3 SOTTORETI.

1. **ASSEGNARE IP A TUTTE LE INTERFACCE UTILIZZANDO I SEGUENTI IP BASE E MAC ADDRESS A TUTTI GLI ADATTATORI DI RETE:**

SUB1: 192.168.1.0/24 , SUB2: 192.168.2.0/24 , SUB3: 192.168.3.0/24

1. **SUBNET1:**
   * 1. 192.168.1.1, 00-00-00-00-00 (HOST B)
     2. 192.168.1.2, 00-00-00-00-01 (HOST A)
     3. 192.168.1.3, 00-00-00-00-02
   1. **SUBNET2:**
      1. 192.168.2.1, 00-00-00-00-03
      2. 192.168.2.2, 00-00-00-00-04 (HOST C)
      3. 192.168.2.3, 00-00-00-00-05 (HOST D)
      4. 192.168.2.4, 00-00-00-00-06
   2. **SUBNET3:**
      1. 192.168.3.1, 00-00-00-00-07
      2. 192.168.3.2, 00-00-00-00-08 (HOST E)
      3. 192.168.3.3, 00-00-00-00-09 (HOST F)
2. **CONSIDERATE DI DOVER INVIARE UN DATAGRAM IP DALL’HOST E ALL’HOST B E CHE LE TABELLE ARP SIANO AGGIORNATE. ELENCARE I PACCHETTI TRASMESSI NELL’INTER-RETE;**
   1. Dato che le tabelle ARP sono gia’ aggiornate non e’ necessario effettuare una richiesta ARP. Il nodo “E” infatti conosce gia’ l’indirizzo IP dell’host “B” ossia 192.168.1.1. La tabella di inoltro di E ci dice che il datagramma deve arrivare all’interfaccia destra del router, con IP 192.168.3.1.
   2. E consulta la propria tabella ARP per trovare il MAC ADDRESS del router di destra, prepara un frame ETHERNET destinato all’interfaccia destra, con MAC: 00-00-00-00-07.
   3. Il frame raggiunge lo switch intermedio, il quale legge gli headers, consulta la propria tabella di commutazione ed inoltra il frame sull’interfaccia 1, facendolo arrivare al router.
   4. Il frame raggiunge l’interfaccia 1 del router di destra, il quale estrae il datagramma dal frame, consulta l’intestazione e la sua tabella di inoltro per ottenere l’indirizzo IP del nodo successivo. Ottenuto l’indirizzo del prossimo router 192.168.2.1, consulta la tabella ARP per ricavarne il MAC address. Successivamente inoltra il frame al MAC: 00-00-00-00-03 utilizzando l’interfaccia 4 con indirizzo fisico: 00-00-00-00-06. Sorgente: 00-00-00-00-06, Destinazione: 00-00-00-00-03
   5. Il frame passa per lo switch che esamina l’intestazione e la sua tabella di commutazione ed inoltra il frame sull’interfaccia 1;
   6. Il frame arriva al router 2 (sull’interfaccia 1), quello di sinistra. Il router consulta la tabella di inoltro per identintificare il nodo a cui inoltrare il datagramma estratto ossia il nodo B di indirizzo IP: 192.168.1.1. Per l’instradamento occorre l’indirizzo MAC per cui viene consultata la tabella ARP ottenendo in questo modo il MAC address del nodo B: 00-00-00-00-0. Sorgente: 00-00-00-00-02 , Destinazione: 00-00-00-00-00.
3. **RIPETERE IL PUNTO PRECEDENTE CONSIDERANDO LA TABELLA ARP DEL MITTENTE VUOTA:**
4. Il nodo E esamina la propria tabella di inoltro scoprendo di dover inviare il datagram all’interfaccia destra del router di destra, IP: 192.168.3.1.
5. La tabella ARP di “E” e’ vuota pertanto effettua una ARP request in broadcast per conoscere il MAC-ADDRESS al quale inoltrare il frame. SORGENTE: 00-00-00-00-08, DESTINATARIO: FF-FF-FF-FF-FF. Quando il router riceve la richiesta, risponde con il suo MAC-ADDRESS e l’host “E” salva la corrispondenza nella tabella ARP.
6. “E” prepara il frame ETHERNET utilizzando come indirizzo destinatario il MAC-ADDRESS del router che ha risposto alla richiesta ARP: 00-00-00-00-07
7. Gli step da questo punto saranno uguali a quelli precedenti in quanto le tabelle ARP sono tutte aggiornate

**ESERCIZIO 18**

Mostrare gli aggiornamenti delle tabelle di commutazione dopo lo scambio dei messaggi.

1. **A - F**

**A invia ARP request in broadcast. Tutti i router salvano il suo MAC address.**

|  |  |  |
| --- | --- | --- |
| **Switch** | **MAC ADDR. (indico nodo)** | **Interfaccia (indico next-hop)** |
| S0 | A | S1 |
| S1 | A | A |
| S2 | A | S1 |
| S3 | A | S0 |
| S4 | A | S0 |

1. **H-A**

Tutti gli switch gia’ sanno come inoltrare verso A. La comunicazione avviene in unicast e il pacchetto percorre S3-S0-S1.

|  |  |  |
| --- | --- | --- |
| **Switch** | **MAC ADDR. (indico nodo)** | **Interfaccia (indico next-hop)** |
| S0 | A | S1 |
| S0 | H | S3 |
| S1 | A | A |
| S1 | H | S0 |
| S2 | A | S1 |
| S3 | A | S0 |
| S3 | H | H |
| S4 | A | S0 |

1. **L-G**

L invia ARP request in broadcast. Tutti i router salvano il suo MAC address.

|  |  |  |
| --- | --- | --- |
| **Switch** | **MAC ADDR. (indico nodo)** | **Interfaccia (indico next-hop)** |
| S0 | A | S1 |
| S0 | H | S3 |
| S0 | L | S4 |
| S1 | A | A |
| S1 | H | S0 |
| S1 | L | S0 |
| S2 | A | S1 |
| S2 | L | S1 |
| S3 | A | S0 |
| S3 | H | H |
| S3 | L | S0 |
| S4 | A | S0 |
| S4 | L | L |

1. **B-L**

Dalla richiesta ARP precedente tutti gli switch hanno imparato i percorsi verso A pertanto la comunicazione avviene in unicast e il pacchetto percorre S1-S0-S4.

|  |  |  |
| --- | --- | --- |
| **Switch** | **MAC ADDR. (indico nodo)** | **Interfaccia (indico next-hop)** |
| S0 | A | S1 |
| S0 | H | S3 |
| S0 | L | S4 |
| S0 | B | S1 |
| S1 | A | A |
| S1 | H | S0 |
| S1 | L | S0 |
| S1 | B | B |
| S2 | A | S1 |
| S2 | L | S1 |
| S3 | A | S0 |
| S3 | L | S0 |
| S3 | H | H |
| S4 | A | S0 |
| S4 | L | L |
| S4 | B | S0 |

**ESERCIZIO 26**

1. IP SORGENTE: 128.119.210.238 (IP-C) , IP DESTINAZIONE: 128.119.28.37 (IP-A)
2. MAC SORGENTE: 63-31-3A-1E-2C-B2 (MAC-4), MAC DESTINAZIONE: FF-FF-FF-FF-FF (broadcast)
3. MAC SORGENTE: 1B-17-58-4D-CD-93 (MAC-A), MAC DESTINAZIONE: 89-1E-90-92-30-B5 (MAC-2)
4. MAC SORGENTE: 89-1E-90-92-30-B5 (MAC-2), MAC DESTINAZIONE: 1B-17-58-4D-CD-93 (MAC-A)

A crea un datagramma IP con IP SORGENTE: 128.119.28.37 (IP-A) e DESTINAZIONE: 128.119.210.238 (IP-C). A esamina la propria routing-table e determina che il datagramma deve essere consegnato all’interfaccia sinistra del router (2).

A effettua quindi un’ARP request, al fine di conosce il MAC-ADDRESS dell’interfaccia 2 del router e il router salva nella sua ARP-TABLE la corrispondenza IP-MAC di A.

Il router risponde con il suo MAC e A (e tutti gli altri host della subnet) lo salvano nella propria ARP-TABLE.

A incapsula il datagramma in un frame con destinazione MAC(2), ottenuto tramite ARP-reply dal router.

Il datagram è ora arrivato al router, il quale esaminando le proprie tabelle di routing lo inoltra alla sua interfaccia 4, quella di destra.

A questo punto viene invocato nuovamente ARP per conoscere il MAC address di C, sempre in broadcast. C risponde al router con il proprio MAC-ADDRESS e salva la corrispondenza IP-4 MAC-4 nella propria ARP-TABLE mentre il router salva la corrispondenza IP-C MAC-C nella propria tabella ARP.

L’interfaccia 4 del router incapsula quindi il datagramma che era stato precedentemente estratto dallo stesso per esaminare gli header, in un frame con destinazione MAC-C. Il frame arriva a C.

C risponde con un ACK. In questo caso non è necessario aggiornare ulteriormente le tabelle ARP in quanto tutti i nodi coinvolti conoscono il percorso per far arrivare il frame ad A.

**ARP-TABLE ROUTER**

|  |  |
| --- | --- |
| IP | MAC |
| 128.119.28.37 (IP-A) | 1B-17-58-4D-CD-93 (MAC-A) |
| 128.119.210.238 (IP-C) | 0B-3E-17-22-AC-39 (MAC-C) |



**TABELLE DI INOLTRO SWITCH**

A prepara datagram con IP SORGENTE IP-A e DESTINAZIONE IP-C. Usa ARP per determinare MAC-2, spedendo un messaggio in broadcast all’indirizzo FF-FF-FF-FF-FF. Lo switch S1 salva già la corrispondenza nella propria tabella. A ottiene MAC-2 e incapsula il datagram in un frame con MAC-SORGENTE MAC-A e DESTINAZIONE MAC-2, appena ottenuto mediante ARP. S1 salva nella propria tabella di commutazione la corrispondenza MAC-2 ETH2, dovuta alla risposta ARP del router.

Inoltra il frame verso l’interfaccia sinistra del router, passando per lo switch S1 che inoltra il frame in broadcast dal momento che non è presente nella sua tabella di commutazione il MAC-ADDRESS dell’interfaccia 2 del router.

Il frame arriva al router il quale estrae il datagramma, legge le intestazioni e consulta la propria tabella di inoltro. Inoltra il datagramma sull’interfaccia di destra e usa ARP per ricavare il MAC di C. La richiesta ARP passa per 2 switch intermedi i quali salvano le corrispondenze MAC-4 ETH4, MAC-7 ETH7. C risponde con il suo MAC ADDRESS e i router intermedi salvano le corrispondenze: MAC-C ETH6, MAC-C ETH7.

Il router ora conosce il MAC di C e Incapsula il datagram in un frame con destinazione MAC-C. Il datagram è arrivato a C.

C ora deve inviare l’ACK ad A e prepara quindi un datagram con IP SORGENTE IP-C e DESTINAZIONE IP-A. Lo incapsula in un frame e lo inoltra al MAC-4, l’interfaccia destra del router. Considerando le condizioni dell’esercizio precedente (ossia che le tabelle ARP vengono aggiornate anche alla richiesta ARP e non solo durante la risposta, l’host C non ha necessità di usare ARP in quanto le tabelle sono già compilate). Anche le tabelle di commutazione degli switch sono già piene e l’ACK può arrivare a destinazione senza nessun aggiornamento.

|  |  |  |
| --- | --- | --- |
| **SWITCH** | **MAC** | **ETH** |
| **S1** | **1B-17-58-4D-CD-93 (MAC-A)** | **1** |
| **S1** | **89-1E-90-92-30-B5 (MAC-2)** | **2** |
|  |  |  |
| **S2** |  |  |
|  |  |  |
| **S3** | **63.31.3A.1E.2C.B2 (MAC-4)** | **4** |
| **S3** | **0B-3E-17-22-AC-39 (MAC-C)** | **7** |
|  |  |  |
| **S4** | **63.31.3A.1E.2C.B2 (MAC-4)** | **7** |
| **S4** | **0B-3E-17-22-AC-39 (MAC-C)** | **6** |