

**Corso di Reti di Calcolatori**  
**Soluzioni alla seconda prova scritta**

*Mauro Brunato, Elio Salvadori*

Venerdì 18 febbraio 2005

**Esercizio 1**

Descrivere le due famiglie di algoritmi distribuiti per la formazione e il mantenimento delle tabelle di instradamento (Distance Vector e Link-State).

**Soluzione** — Vedere il libro.

**Esercizio 2**

Con riferimento all'indirizzamento IP di tipo classless (senza classi) si consideri l'arco di indirizzi IP da 134.132.0.0 a 134.136.255.255.

**2.1)** Scrivere sia in notazione [indirizzo, subnet mask] che in notazione slash il blocco CIDR più piccolo in grado di contenere tale arco.

**2.2)** Quante reti di classe B sono contenute da tale blocco CIDR?

**2.3)** Suddividere il blocco CIDR in 8 sottoreti (che chiameremo nell'ordine *sub1*, *sub2*, ..., *sub8*) di uguale dimensione. Scrivere in notazione slash gli indirizzi delle 8 sottoreti.

**2.4)** Qual è l'indirizzo di broadcast della sottorete *sub3*?

**Soluzione** —

**2.1)** Per definire il blocco CIDR più piccolo dobbiamo individuare i bit che rimangono costanti in tutti gli indirizzi dell'arco richiesto. Il primo e l'ultimo indirizzo sono rispettivamente (traducendo in binario soltanto la parte interessante):

134.10000100.0.0  
...  
134.10001000.0.0

Ne segue che i bit costanti sono gli 8 del primo otteetto (134) e i primi quattro del secondo. Azzerando i bit non facenti parte della parte di rete, otteniamo l'indirizzo

134.128.0.0/255.240.0.0, oppure 134.128.0.0/12.

Da questo punto in poi, l'arco di indirizzi originario non è più utilizzato dall'esercizio, va sempre considerato il blocco CIDR appena determinato.

**2.2)** Una rete di classe B prevede 16 bit di rete, noi ne abbiamo 12 fissi, ne restano 4 da variare, per un totale di  $2^4 = 16$  sottoreti di classe B.

**2.3)** Per individuare 8 sottoreti sono necessari  $\lceil \log_2 8 \rceil = 3$  bit. Ciascuna delle sottoreti sarà quindi caratterizzata dai primi  $12 + 3$  bit:

sub1 = 134.128.0.0/15  
sub2 = 134.130.0.0/15  
sub3 = 134.132.0.0/15  
sub4 = 134.134.0.0/15  
sub5 = 134.136.0.0/15  
sub6 = 134.138.0.0/15  
sub7 = 134.140.0.0/15  
sub8 = 134.142.0.0/15

2.4) Vanno posti a 1 i bit di host (cioè i restanti 17) della rete sub3. Attenzione a porre a 1 anche il bit meno significativo del secondo otteetto, che fa parte dell'indirizzo di host:

134.133.255.255

### Esercizio 3

Descrivere brevemente il livello fisico e data-link di almeno tre standard di rete locale.

**Soluzione** — Vedere il libro.

### Esercizio 4

L'host A deve spedire all'host B 10 pacchetti da 1500 byte lungo una linea dedicata a 10Mbps con latenza pari a 10ms. Si considerino i due casi:

1. A e B utilizzano un protocollo stop and go.
2. A e B utilizzano il protocollo TCP a partenza lenta e finestra di congestione massima pari a 4 pacchetti. Si consideri già avvenuto l'handshake di connessione e si ignori quello di disconnessione.

In entrambi i casi, la trasmissione deve considerarsi terminata quando l'host A ha ricevuto conferma dell'ultimo pacchetto.

Si supponga che non vi siano errori di trasmissione e si trascurino le intestazioni aggiunte dai livelli inferiori.

4.1) Calcolare il tempo richiesto dal trasferimento nei due casi.

4.2) Nel caso del protocollo TCP, calcolare il tempo richiesto dal trasferimento se si considerano anche gli handshake di connessione e di disconnessione (scegliere liberamente se considerare o no eventuali piggybacking e chi inizia ciascuno degli handshake).

**Soluzione** — Si presenta una possibile interpretazione del testo. Le ipotesi sulle dimensioni dei pacchetti di servizio (ACK, SYN, FIN) e sulla presenza o meno di una soglia di linearità nella finestra di congestione TCP possono dare luogo a diversi possibili svolgimenti.

4.1) Il tempo di spedizione di un pacchetto è

$$T_s = \frac{1500 \cdot 8}{10 \cdot 10^6 \text{s}^{-1}} = 1.2 \cdot 10^{-3} \text{s}.$$

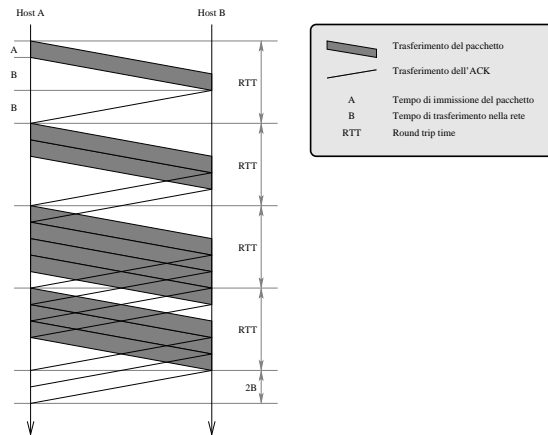
Nell'ipotesi che gli ACK siano di dimensione trascurabile, e detta  $T_1$  la latenza della rete, un RTT è pari a

$$T_{\text{RTT}} = T_s + 2T_1 = 1.2 \cdot 10^{-3} \text{s} + 2 \cdot 10 \cdot 10^{-3} \text{s} = 21.2 \cdot 10^{-3} \text{s}.$$

In base alle ipotesi dell'esercizio, la trasmissione è da considerarsi conclusa al termine del decimo RTT, quindi

$$T_{\text{stop\&go}} = 10T_{\text{RTT}} = 0.212\text{s}.$$

Per quanto riguarda TCP, la temporizzazione completa nelle ipotesi semplificate dell'esercizio è la seguente (per maggior visibilità dei dettagli, la scala fra il tempo di pacchetto e la latenza non è rispettata):



Si noti che la trasmissione di un burst di pacchetti è sostanzialmente avviata dalla ricezione del primo ACK in ritorno. Il tempo totale di completamento è dunque

$$T_{\text{TCP}} = 4T_{\text{RTT}} + 2T_1 = 4 \cdot 21.2 \cdot 10^{-3}\text{s} + 2 \cdot 1.2 \cdot 10^{-3}\text{s} = 87.2 \cdot 10^{-3}\text{s}.$$

**4.2)** Se non si considera alcun piggybacking nei pacchetti dati, servono tre pacchetti all'inizio e tre alla fine. Nell'ipotesi che l'handshake sia iniziato da A, servono due ulteriori  $T_1$  all'inizio (l'ultimo ACK dell'handshake viene mandato appena prima del primo pacchetto dati) e tre alla fine; il tempo di trasmissione arriva dunque a

$$T'_{\text{TCP}} = T_{\text{TCP}} + 5T_1 = 87.2 \cdot 10^{-3}\text{s} + 5 \cdot 10 \cdot 10^{-3}\text{s} = 0.1372\text{s}.$$

Il piggybacking può efficacemente influire sul tempo di chiusura. Il flag FIN può essere attivato sull'ultimo pacchetto dati inviato da A, e così l'ultima risposta di B. Resta soltanto l'ACK di chiusura da parte di A, per un totale di tre  $T_1$ , due all'inizio e uno alla fine. In tal caso,

$$T''_{\text{TCP}} = T_{\text{TCP}} + 3T_1 = 87.2 \cdot 10^{-3}\text{s} + 3 \cdot 10 \cdot 10^{-3}\text{s} = 0.1172\text{s}.$$