

Modello di servizio TCP – 1

- 2 tipi di servizio a livello trasporto
 - TCP (**Transmission Control Protocol**)
 - Garantisce comunicazione affidabile agli utenti (*end point*) anche in presenza di una rete inaffidabile
 - **Connection-oriented**
 - UDP (**User Datagram Protocol**)
 - Fornisce agli utenti esclusivamente la qualità di comunicazione offerta dal livello IP sottostante
 - **Connectionless**
 - Migliore velocità di trasporto per minore affidabilità

Livello trasporto (TCP/IP) - parte 2

Architettura degli Elaboratori 2 - T. Vardanega

Pagina 429

Modello di servizio TCP – 2

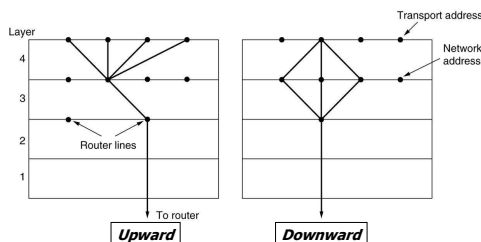
- Il servizio TCP richiede la creazione di **socket** ai 2 estremi della connessione
 - La coppia di **socket** denota la connessione che può convogliare **più** conversazioni simultanee tra i suoi punti terminali
 - Un singolo **socket** può essere punto terminale di più connessioni (*server*)
 - Ogni **socket** ha **identità unica** determinata dall'indirizzo IP del nodo ospite (32 bit) e dal numero di **porta locale** (16 bit)

Livello trasporto (TCP/IP) - parte 2

Architettura degli Elaboratori 2 - T. Vardanega

Pagina 430

Multiplexing



- Più connessioni di livello 4 (Trasporto) possono viaggiare su una stessa connessione di rete (*upward multiplexing*)
- Una connessione di livello 4 può aggirare limiti di traffico imposti da circuiti virtuali di livello 3 (Rete) attivando più connessioni di rete (*downward*)

Livello trasporto (TCP/IP) - parte 2

Architettura degli Elaboratori 2 - T. Vardanega

Pagina 431

Modello di servizio TCP – 3

- I numeri di porta nell'intervallo 0..255 (**well-known ports**) sono preassegnati ad alcuni **servizi standard**, alcuni dei quali usano TCP altri UDP
 - Alcuni servizi standard che usano TCP
 - 25 SMTP Simple Mail Transfer Protocol
 - 80 HTTP HyperText Transfer Protocol
 - 110 POP3 Post Office Protocol versione 3

Livello trasporto (TCP/IP) - parte 2

Architettura degli Elaboratori 2 - T. Vardanega

Pagina 432

Modello di servizio TCP – 4

- La connessione TCP è **bidirezionale (full-duplex)** e **punto a punto**
 - Non fornisce supporto per comunicazioni a diffusione (**broadcast**) e neppure per gruppo (**multicast**)
- La connessione TCP è a **flusso di dati (byte stream)**
 - Le unità di trasporto **non** corrispondono alle unità dati riconosciute dall'utente applicativo
 - Inviati x B → Trasportati $(x/2 + x/2)$ B → Ricevuti x B
- Vi è **memorizzazione** (e dunque **ritardo**)
 - Tra ricezione in agente TCP da **M** ed emissione su rete
 - Tra ricezione da rete ed emissione verso **D**
- Le prime versioni di TCP prevedevano un'opzione per effettuare **consegna immediata** (modalità "PUSH")

Livello trasporto (TCP/IP) - parte 2

Architettura degli Elaboratori 2 - T. Vardanega

Pagina 433

Il protocollo TCP – 1

- Unità di trasporto TCP (**segmento**) composta da un **prefisso** obbligatorio e da un campo dati utente di ampiezza **variabile**
- Ampiezza massima di segmento determinata dall'esigenza di
 - Contenere l'intero segmento entro il campo dati di 1 **datagram** = $(64k - 1) - 20$ B
 - Limite determinato dal livello IP
 - Tener conto della massima dimensione effettiva delle unità di trasporto su rete per evitare costose frammentazioni
 - Limite determinato dalle caratteristiche (di rami) della rete
 - Spesso limitata dalla dimensione del campo dati della trama **Ethernet** (= 1500 B)

Livello trasporto (TCP/IP) - parte 2

Architettura degli Elaboratori 2 - T. Vardanega

Pagina 434

Il protocollo TCP – 2

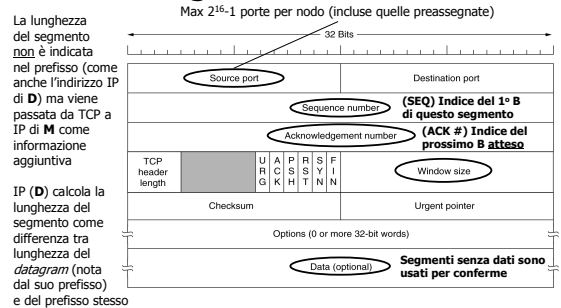
- Un segmento TCP troppo largo per le capacità di un tratto della rete viene frammentato in più **datagram** ciascuno con proprio prefisso e indice di frammento
 - Maggior costo trasmissivo
 - Onere di ricostituzione del segmento con maggiore complessità nella gestione delle conferme
 - I frammenti vengono confermati nel corretto ordine di sequenza e non nel loro ordine di arrivo (come in SWP)
 - Arrivo: 1 | 3 | 0 | 4 | 2
 - Conferma: - | - | 1 | - | 4

Livello trasporto (TCP/IP) - parte 2

Architettura degli Elaboratori 2 - T. Vardanega

Pagina 435

Il segmento TCP – 1



Livello trasporto (TCP/IP) - parte 2

Architettura degli Elaboratori 2 - T. Vardanega

Pagina 436

Il segmento TCP – 2

- Così come il **datagram** anche il **prefisso** di segmento deve indicare la propria lunghezza (in unità di 32 *bit*)
 - A causa della presenza di un campo opzionale
- Il successivo campo da 6 *bit* era previsto per sviluppi futuri ma al momento è ancora inutilizzato
- Seguono 6 *flag* da 1 *bit* ciascuno
 - **URG** : se 1, il campo **Urgent Pointer** indica la posizione in *byte* del segmento corrente alla quale si trovano dati **urgenti**
 - Nel caso a **D** viene inviata notifica immediata di interruzione con invio dei dati corrispondenti
 - **ACK** : se 1 il campo **ACK #** è valido, altrimenti va ignorato

Livello trasporto (TCP/IP) - parte 2

Architettura degli Elaboratori 2 - T. Vardanega

Pagina 437

Il segmento TCP – 3

- Gli altri *flag* da 1 *bit* ciascuno ...
 - **PSH** : se 1 richiede all'entità **TCP** di lato **D** di consegnare il segmento a **D** senza introdurre ritardo (= memorizzazione al livello **TCP**)
 - **RST** : se 1 richiede *reset* di connessione
 - **SYN** : usato per stabilire una connessione
 - SYN=1 ACK=0 : **M** → **D** : richiesta di connessione
 - SYN=1 ACK=1 : **D** → **M** : richiesta accettata
 - **FIN** : se 1 notifica la richiesta di fine connessione
 - **Rilascio simmetrico**
 - Un processo può continuare a ricevere dati ben oltre il proprio invio di richiesta di fine connessione

Livello trasporto (TCP/IP) - parte 2

Architettura degli Elaboratori 2 - T. Vardanega

Pagina 438

Il segmento TCP – 4

- Nel campo **window size** (da 16 *bit*) l'entità **TCP** di lato **D** indica l'ampiezza della propria finestra di ricezione per la connessione
 - Il valore 0 richiede a **M** di sospendere ogni nuovo invio confermando però ricezione fino al B di indice (**ACK#** – 1)
 - **M** potrà riprendere l'invio solo dopo l'emissione da parte di **D** di un segmento con lo stesso ACK # precedente ma indicante ampiezza di finestra > 0
 - La finestra minima deve avere capienza sufficiente per un segmento di massima ampiezza (< 64 kB)

Livello trasporto (TCP/IP) - parte 2

Architettura degli Elaboratori 2 - T. Vardanega

Pagina 439

Il segmento TCP – 5

- Il campo **checksum** consente a **D** di effettuare controllo di integrità di segmento
 - Complemento a 1 della somma in complemento a 1 di tutte le parole a 16 *bit* del segmento
 - Aggiungendo al segmento un **pseudo-prefisso** che include: (i) gli indirizzi **IP** di **M** e **D**, (ii) 1 B a 0, (iii) 1 B identificatore di protocollo (TCP=6), (iv) 2 B che indicano l'ampiezza in B dell'intero segmento (non presente nel prefisso reale)
 - **Violazione di astrazione!** TCP conosce gli indirizzi **IP**
 - Ampiezza massima di segmento < $(2^{16} - 1 - 20)$ B
 - Rendendo pari l'ampiezza del campo dati con l'aggiunta di 1 B di 0 (se necessario)
 - Considerando 0 il campo **checksum** ricevuto

Livello trasporto (TCP/IP) - parte 2

Architettura degli Elaboratori 2 - T. Vardanega

Pagina 440

Il segmento TCP – 6

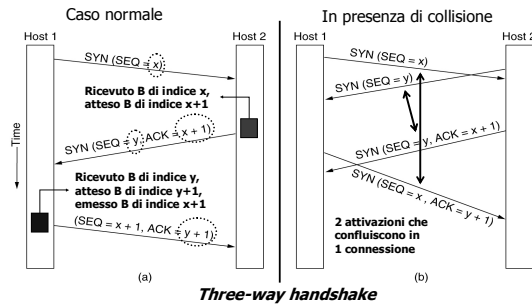
- **D** può usare il campo opzionale per indicare la massima dimensione accettabile dell'area dati di segmento
- **M** può usarlo per fare la sua proposta
 - Il valore minore tra i 2 viene selezionato
 - Altrimenti si assume semplicemente il valore 536 B
 - Dunque segmenti di dimensione massima $536 + 20 = 556$ B
- Il campo opzionale può anche servire per fissare una dimensione di finestra di ricezione su **32 bit**
 - Più ampia di quella standard
 - Quando la rete permette emissione veloce di unità ampie 64 kB (segmento di massima dimensione) consente a **M** di evitare frequenti periodi di attesa forzata

Livello trasporto (TCP/IP) - parte 2

Architettura degli Elaboratori 2 - T. Vardanega

Pagina 441

Attivazione di connessione – 1



Three-way handshake

Livello trasporto (TCP/IP) - parte 2

Architettura degli Elaboratori 2 - T. Vardanega

Pagina 442

Attivazione di connessione – 2

- Caso normale (singola attivazione)
 - M** → **D** (SEQ = x, SYN = 1, ACK = 0)
 - D** → **M** (SEQ = y, SYN = 1, ACK = 1, Ack# = x+1)
 - M** → **D** (SEQ = x+1, SYN = 1, ACK = 1, Ack# = y+1)
- 2 attivazioni sulla stessa connessione
 - M**₁ → **D**₂ (SEQ = x, SYN = 1, ACK = 0)
 - M**₂ → **D**₁ (SEQ = z, SYN = 1, ACK = 0)
 - D**₂ → **M**₁ (SEQ = y, SYN = 1, ACK = 1, Ack# = x+1)
 - D**₁ → **M**₂ (SEQ = k, SYN = 1, ACK = 1, Ack# = z+1)
 - Se entrambe le attivazioni hanno successo esse danno luogo a 1 sola connessione con indici iniziali (**x, y**)

Livello trasporto (TCP/IP) - parte 2

Architettura degli Elaboratori 2 - T. Vardanega

Pagina 443

Politica di trasmissione TCP – 1

- **Controllo di flusso** concerne la disponibilità di memoria in **D** e non l'emissione di conferme di ricezione
 - Con ogni conferma emessa **D** aggiorna la propria disponibilità residua (**window size**)
 - **M** usa questo dato per disciplinare i propri invii
- Lo stile di comunicazione tra **M** e **D** può dar luogo a situazioni di estrema inefficienza
 - **M** potrebbe inviare 1 B di dati alla volta
 - 1 segmento da 21 B → 1 **datagram** da 41 B
 - **D** potrebbe consegnare al suo processo utente 1 B alla volta
 - **M** alterna attese a invii di ampiezza 1 B

Livello trasporto (TCP/IP) - parte 2

Architettura degli Elaboratori 2 - T. Vardanega

Pagina 444

Politica di trasmissione TCP – 2

- Ove possibile conviene raggruppare la conferma e l'aggiornamento di disponibilità con la 1ª emissione di dati nella stessa direzione sulla stessa connessione
 - Per ridurre il numero di segmenti inviati senza carico utile
 - **Piggybacking**
- Quando **TCP** di lato **M** riceve dal suo processo utente 1 B alla volta
 - Invia il 1º B memorizzando ogni altro B **ritardandone l'invio** sino all'arrivo della 1ª conferma
 - A quel punto invia tutti i B memorizzati in 1 singolo segmento memorizzando ogni altro B fino alla conferma di ricezione del segmento
 - **Algoritmo di Nagle**
 - **Inadatto** per sessioni grafiche interattive su rete perché corrompe la tempistica degli eventi mostrati all'utente

Livello trasporto (TCP/IP) - parte 2

Architettura degli Elaboratori 2 - T. Vardanega

Pagina 445

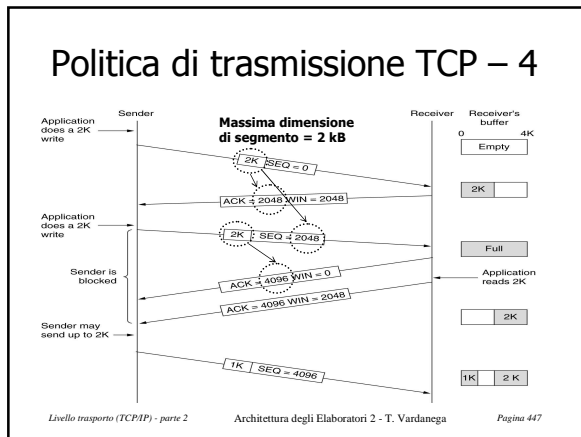
Politica di trasmissione TCP – 3

- Quando **TCP** di lato **M** ha molti dati da inviare ma **TCP** di lato **D** può inviargli al suo processo destinatario solo 1 B alla volta
 - **TCP** di lato **D** ritarda l'emissione verso **M** della propria disponibilità di ricezione fino a quando non ne abbia accumulata una quantità decente
 - Tipicamente almeno metà del proprio **buffer** oppure la dimensione massima di segmento dichiarata in fase di inizializzazione di connessione
 - **Algoritmo di Clark (silly window problem)**

Livello trasporto (TCP/IP) - parte 2

Architettura degli Elaboratori 2 - T. Vardanega

Pagina 446



Politica di trasmissione TCP – 5

- TCP di lato **M** adatta il suo ritmo alla capacità della finestra di ricezione di TCP di lato **D**
 - M** → **D** (SEQ = 0, dati = 2 kB)
 - D** → **M** (Ack# = 2048, WIN = 2048)
 - M** → **D** (SEQ = 2048, dati = 2 kB)
 - **M** deve sospendere ogni nuovo invio perché ha per ora esaurito la capacità ricettiva di **D**
 - D** → **M** (Ack# = 4096, WIN = 0)
 - D** → **M** (Ack# = 4096, WIN = 2048)
 - **M** può riprendere gli invii

Livello trasporto (TCP/IP) - parte 2 Architettura degli Elaboratori 2 - T. Vardanega Pagina 448

Controllo di congestione – 1

- In presenza di reti fortemente inaffidabili è difficile determinare la causa della perdita di segmento segnalata da un *time out* di ricezione
 - **Congestione di rete**
 - Ritardo sulla linea, con o senza perdita di pacchetto
 - Errori trasmissivi
 - Perdita di pacchetto
- Non conviene fare assunzioni se non si conosce la topologia del tratto di rete da attraversare
 - Le connessioni che usano linee fisiche hanno buona affidabilità
 - L'arrivo del *time out* è sintomo di congestione
 - Le connessioni *wireless* sono invece molto inaffidabili
 - L'arrivo del *time out* può essere sintomo di interferenza

Livello trasporto (TCP/IP) - parte 2 Architettura degli Elaboratori 2 - T. Vardanega Pagina 449

Controllo di congestione – 2

- Il **controllo di flusso** lega l'emissione di segmenti da parte di **M** alla capacità ricettiva di **D**
 - Assumendo una rete con sufficiente capacità trasmissiva
- Il **controllo di congestione** aiuta ad alleviare la perdita di pacchetti in presenza di reti inaffidabili
 - Può essere necessario indipendentemente dalla capacità di ricezione di **D**

Livello trasporto (TCP/IP) - parte 2 Architettura degli Elaboratori 2 - T. Vardanega Pagina 450

Controllo di congestione – 3

- **M** mantiene 2 finestre concettuali d'invio
 - **rw** = capacità di ricezione dichiarata da **D**
 - **cw** = dimensione di segmento che si ritiene non incorra in congestione della rete
 - Congestion window
- **M** invia segmenti di dimensione pari al minimo tra **rw** e **cw**
- **M** usa un'euristica per approssimare l'ampiezza migliore di **cw**

Livello trasporto (TCP/IP) - parte 2 Architettura degli Elaboratori 2 - T. Vardanega Pagina 451

Controllo di congestione – 4

- **M** inizializza **cw** alla massima dimensione di segmento permesso sulla connessione (**cs**)
- **M** emette una sequenza (*burst*) di **cw/cs** segmenti ampi **cs** verificando per ciascuno l'arrivo della conferma di ricezione prima del *time out*
- Per ogni singola conferma ricevuta **M** incrementa l'ampiezza di **cw** di 1 **cs** e ripete l'operazione finché non si verifichi un *time out* o **cw** diventi > **rw**
 - In questo modo l'ampiezza di **cw** raddoppia a ogni conferma completa di un'intera sequenza
- **Algoritmo slow start**
 - Adottato come base da tutte le realizzazioni concrete di TCP

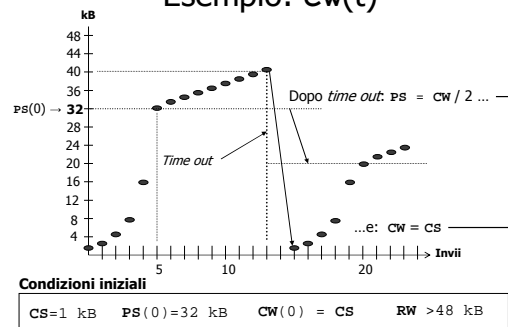
Livello trasporto (TCP/IP) - parte 2 Architettura degli Elaboratori 2 - T. Vardanega Pagina 452

Controllo di congestione – 5

- L'algoritmo di **controllo di congestione** usa un ulteriore parametro soglia ps inizializzato alla massima dimensione teorica di segmento (= 64 kB)
- A ogni *time out* si aggiornano i valori
 - $ps(i+1) = cw(i) / 2$
 - $cw(i+1) = cs$
- Fino al successivo *time out* si usa l'algoritmo **slow start** per riportare cw a un valore ottimale

Livello trasporto (TCP/IP) - parte 2 Architettura degli Elaboratori 2 - T. Vardanega Pagina 453

Esempio: $cw(t)$



Livello trasporto (TCP/IP) - parte 2 Architettura degli Elaboratori 2 - T. Vardanega Pagina 454

Intervallo di *time out* – 1

- TCP usa un'euristica per determinare la durata dell'intervallo di *time out* adatta alle prestazioni effettive della rete
 - La rete viene sottoposta a monitoraggio continuo
 - La variabile **RTT** (*round-trip time*) denota la miglior stima corrente dell'intervallo tra un invio e la sua conferma di ricezione
 - Un orologio misura l'intervallo T per ogni invio
 - Il valore di **RTT** viene aggiornato a
 - $RTT = \alpha RTT + (1 - \alpha) T$
 - dove α ha valore prevalente (p.es. 7/8)

Livello trasporto (TCP/IP) - parte 2 Architettura degli Elaboratori 2 - T. Vardanega Pagina 455

Intervallo di *time out* – 2

- L'idea iniziale era fissare l'intervallo di *time out* come βRTT (p.es. $\beta = 2$)
- Si è però osservato sperimentalmente che non conviene usare un fattore β costante
- Attualmente l'intervallo di *time out* viene calcolato come $RTT + 4 \times D$
 - Dove D viene costantemente aggiornato al valore $D = \alpha D + (1 - \alpha) |RTT - T|$

Livello trasporto (TCP/IP) - parte 2 Architettura degli Elaboratori 2 - T. Vardanega Pagina 456

Intervallo di *time out* – 3

- In presenza di reinvio a seguito di *time out* è difficile dire se una successiva conferma di ricezione concerne il 1° invio o la sua ritrasmissione!
 - L'attribuzione erronea di T corrompe **RTT**
- **Algoritmo di Karn** prevede di non utilizzare per il calcolo di **RTT** i valori di T relativi a segmenti ritrasmessi

Livello trasporto (TCP/IP) - parte 2 Architettura degli Elaboratori 2 - T. Vardanega Pagina 457

Altre misurazioni

- L'entità TCP usa altri 2 valori temporali oltre a **RTT**
 - Un orologio misura il tempo trascorso da quando D abbia notificato disponibilità 0 senza più aggiornarla
 - Contro un **limite di persistenza** prefissato
 - Trascorso tale limite, M chiede a D di aggiornare la sua disponibilità di ricezione
 - Un altro orologio misura il tempo dall'ultimo invio o ricezione sulla connessione
 - Contro un **limite di attività vitale** prefissato
 - Trascorso tale limite il lato dove questo è avvenuto invia un **segmento di controllo** per verificare la presenza della controparte
 - In assenza di risposta lo stesso lato **rilascia** la connessione

Livello trasporto (TCP/IP) - parte 2 Architettura degli Elaboratori 2 - T. Vardanega Pagina 458

Modello di servizio UDP

- UDP fornisce all'utente semplicemente accesso diretto al servizio datagram di IP
- L'entità di trasporto UDP non effettua alcun controllo di correttezza di trasmissione
- L'unità dati UDP è un normale datagram prefissato da
 - 2 parole da 16 *bit* indicanti le porte di **M** e **D**
 - 1 parola da 16 *bit* indicante l'ampiezza del datagram
 - 1 parola da 16 *bit* con checksum calcolato con la stessa tecnica usata per il segmento TCP