



Il livello trasporto

Introduzione

Multiplexing e demultiplexing

Il protocollo UDP

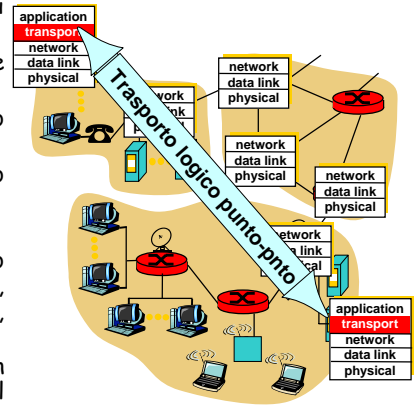
Ing. Nadia Ranaldo

1



Servizi e protocolli a livello trasporto

- Forniscono *comunicazione logica* tra processi applicativi su host differenti
 - Il livello rete invece fornisce comunicazione logica tra host
 - **Multiplexing** e **demultiplexing** a livello trasporto
- I protocolli a livello trasporto sono implementati nei sistemi terminali
 - Non nei router di rete
 - *send side*: converte i messaggi a livello applicativo in **segmenti a livello trasporto**, aggiungendo un'intestazione di trasporto, e li passa al livello rete
 - *receive side*: riasmonta i segmenti in messaggi a livello applicativo e li passa al livello applicativo
- Più protocolli a livello trasporto sono disponibili alle applicazioni
 - Internet: TCP e UDP



2



Protocolli a livello trasporto di Internet

- UDP: fornisce un servizio non affidabile e senza connessione
 - Semplice estensione del servizio "best-effort" del protocollo IP
 - Non assicura né la consegna dei segmenti, né il rispetto dell'ordine originario e non garantisce l'integrità dei dati all'interno dei segmenti
 - Controllo di integrità includendo campi di riconoscimento dell'errore nell'intestazione del segmento
- TCP: fornisce un servizio affidabile e orientato alla connessione
 - Controllo della congestione
 - Evita che le connessioni TCP sovraccarichino i router e i collegamenti con una eccessiva quantità di traffico
 - Controllo di flusso
 - Adatta la frequenza di invio del mittente con quella di lettura dell'applicazione ricevente
 - Controllo dell'integrità dei dati
- I servizi offerti dai protocolli a livello trasporto possono essere vincolati ai servizi offerti dal sottostante protocollo a livello di rete
 - Nel caso dei protocolli TCP e UDP i servizi non disponibili sono relativi alla garanzia sui ritardi e sulla banda di trasmissione

3



Multiplexing/demultiplexing (1)

segmento - unità di dati scambiata tra entità del livello di trasporto
- Anche **TPDU: transport protocol data unit**
- Un datagramma IP trasporta un segmento

Demultiplexing: consegna dei segmenti ricevuti al processo applicativo appropriato in esecuzione sull'host



Legenda:

○ Processo □ Socket

Il livello trasporto nell'host di ricezione trasferisce i dati ad una socket (non direttamente al processo)

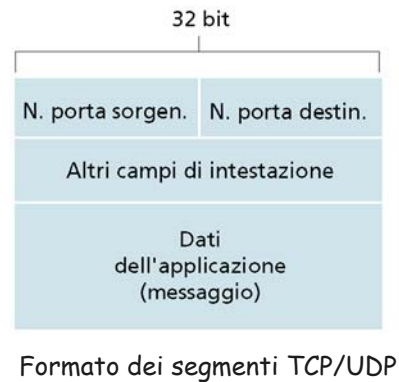
4

Multiplexing/demultiplexing (2)

Multiplexing: Recupero di dati da diversi processi applicativi, costruzione di segmenti ed invio attraverso un unico canale

multiplexing/demultiplexing:

- Basati sui numeri di porta e gli indirizzi IP dei mittenti e riceventi
 - In ogni segmento → num. di porta del mittente e del destinatario
 - Ogni socket deve avere un numero di porta
 - Quando un segmento arriva all'host, il livello trasporto esamina il numero di porta destinazione e dirige il segmento verso la socket corrispondente



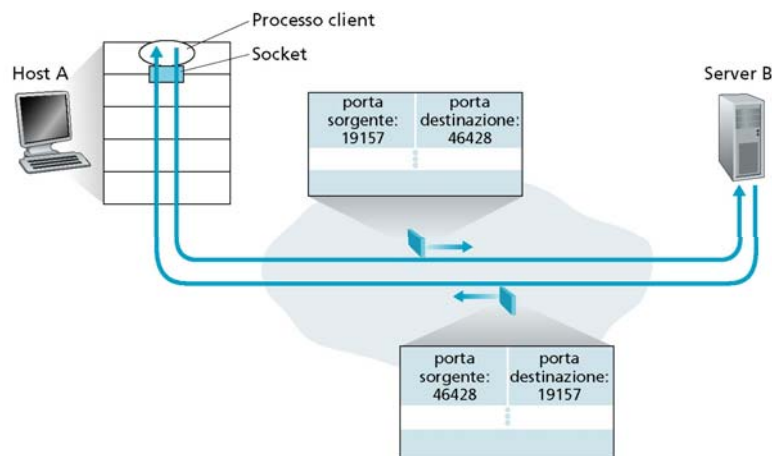
5

Multiplexing/demultiplexing senza connessione (1)

- Creazione di socket con numero di porta:
`DatagramSocket mySocket1 = new DatagramSocket(12534);`
- Creazione di socket con numero di porta assegnato automaticamente:
`DatagramSocket mySocket2 = new DatagramSocket();`
- Quando un processo invia un blocco di dati ad un altro processo il livello trasporto esegue le seguenti operazioni (multiplexing):
 - Crea un segmento che include i dati applicativi, i numeri di porta origine e destinazione
 - Passa il segmento al livello rete
 - Quest'ultimo lo incapsula in un datagramma IP ed effettua un tentativo best-effort di consegna del segmento
- Quando un host riceve un segmento UDP il livello trasporto esegue le seguenti operazioni (demultiplexing):
 - Controlla il numero di porta destinazione contenuto nel segmento
 - Dirige il segmento UDP alla socket con quel numero di porta

6

Multiplexing/demultiplexing senza connessione (2)



Inversione dei numeri di porta origine e destinazione per l'indirizzo di "ritorno"

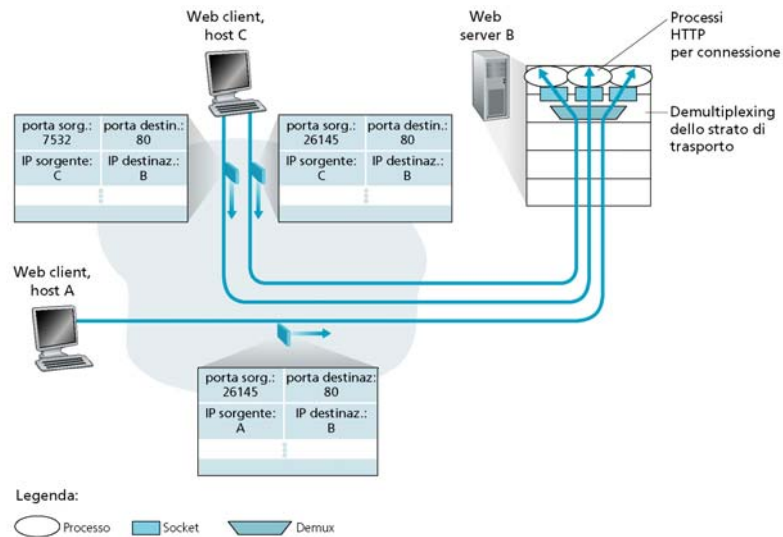
7

Multiplexing/demultiplexing con connessione (1)

- Mentre una socket UDP è identificata dalla coppia (indirizzo IP-num. di porta destinazione) una socket TCP è identificata da quattro valori:
 - Indirizzo IP mittente - Numero di porta mittente
 - Indirizzo IP destinatario - Numero di porta destinatario
- Il livello trasporto dell'host ricevente utilizza tutti e quattro i valori per dirigere i segmenti alla socket appropriata (ad eccezione dei segmenti TCP contenente la richiesta di connessione)
- Un host può supportare più socket TCP simultanee, collegate a processi differenti:
 - Ogni socket è identificata dai suoi quattro valori
- Es: i server Web solitamente generano un nuovo processo o creano un nuovo thread per ogni nuova connessione client
 - Se si utilizza HTTP persistente, client e server scambiano messaggi HTTP attraverso la stessa socket per tutta la durata della connessione persistente
 - Se si utilizza HTTP non persistente, allora viene creata e chiusa una nuova connessione TCP per ciascuna coppia richiesta /risposta

8

Multiplexing/demultiplexing con connessione (2)



9

UDP: User Datagram Protocol [RFC 768]

- Servizio "best effort" con aggiunta di multiplexing/demultiplexing
- I segmenti UDP potrebbero essere:
 - persi
 - consegnati fuori ordine all'applicazione
- **Servizio senza connessione:**
 - no protocollo di handshaking tra mittenti e riceventi UDP
 - ogni segmento UDP è gestito in maniera indipendente dagli altri

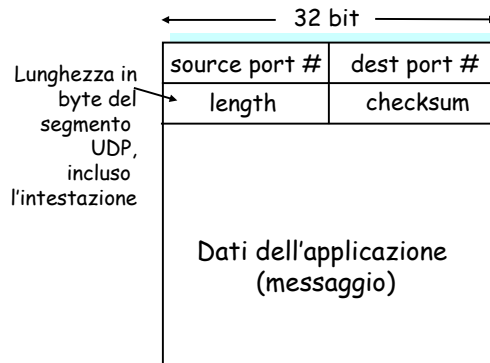
Perchè è utile il protocollo UDP?

- Nessuna connessione stabilita
 - Non introduce ritardo prima di trasferire dati come per il TCP
- Non viene gestito lo stato della connessione sul mittente e sul ricevente
 - Il TCP deve mantenere lo stato dei buffer di ricezione ed invio, parametri per il controllo della congestione, etc.
 - Un server può comunicare con più client contemporaneamente mediante UDP
- Intestazione del segmento piccolo (8 byte, mentre per il TCP 20 byte)
- No controllo della congestione: UDP può trasmettere alla velocità che si desidera
 - Più adatto ad applicazioni in tempo reale che tollerano una certa perdita di dati

10

UDP: Struttura dei segmenti UDP

- Spesso è utilizzato per applicazioni di streaming audio/video
- Altri usi del protocollo UDP
 - DNS
 - RIP
 - Aggiornamento delle tabelle di instradamento
 - SNMP
 - Gestione della rete
- Trasferimento affidabile con UDP: aggiungere affidabilità a livello applicazione
 - Con meccanismi di notifica e ritrasmissione
 - Recupero degli errori senza i vincoli sulla frequenza trasmissiva imposti dal TCP



11

Checksum UDP

- **Scopo:** rilevare "errori" nei segmenti trasmessi
 - Ad esempio a causa di disturbi nei collegamenti o in un router
 - Non c'è garanzia che tutti i collegamenti tra origine e destinazione controllino gli errori

Mittente:

- Considera il contenuto del segmento come una sequenza di interi a 16 bit
- Checksum (16 bit): esegue la somma del contenuto del segmento
- L'eventuale riporto viene sommato al primo bit
- Effettua il complemento ad uno della somma
- Inserisce tale valore nel campo checksum

Ricevente:

- Calcola la checksum del segmento ricevuto
- Controlla se la checksum calcolata è uguale al valore del campo checksum:
 - NO - segnalazione di errore
 - alcune implementazioni di UDP scartano il segmento, altre lo consegnano all'applicazione con un avvertimento
 - SI - non ci sono errori

12

Esempio di calcolo della checksum

- Somma di due interi a 16 bit
 - Somma dell'eventuale riporto
- Complemento bit a bit della somma
- Nota: per verificare la checksum è possibile anche sommare tutto il segmento (incluso il campo checksum) e verificare che il complemento a uno (negato) sia uguale a zero!

	1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
	1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
	1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1
sum	1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0
checksum	0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1

1