Multiplexing e Demultiplexing

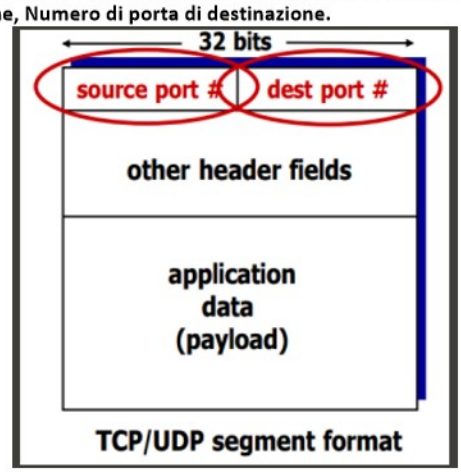
giovedì 10 agosto 2023 16:30

Il multiplexing e il demultiplexing rappresentano come il servizio di trasporto da host-to-host fornito al livello di rete, possa diventare un servizio di trasporto end-to-end per le applicazioni in esecuzione sugli host.

Multiplexing	Demultiplexing
È il compito di radunare frammenti di dati da diverse socket ed incapsularle aggiungendo un'**intestazione** *(usata successivamente nella fase di demultiplexing)*, creando così un **segmento** che verrà poi passato a livello di rete.	È il compito per cui quando riceviamo dei segmenti dal livello di rete, questi vengono ridirezionati verso la giusta porta socket.

A livello transport il multiplexing richiede:

1. Le socket abbiano identificatori univoci;
2. Ciascun segmento abbia campi che indichino come identificare la socket di destinazione: **Numero di porta di origine, Numero di porta di destinazione.**

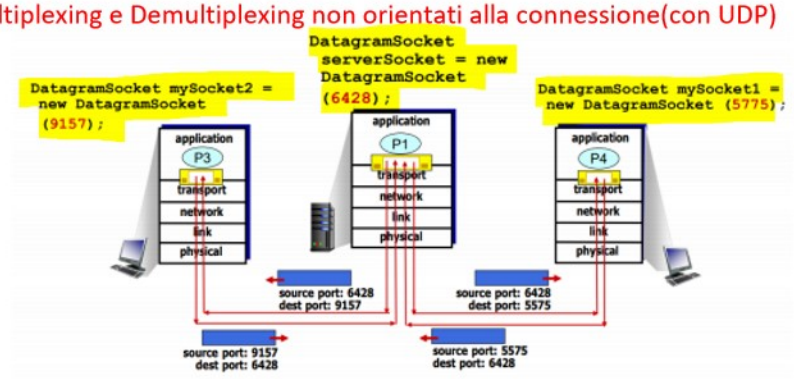


versione generale

Ricorda: l'host oltre che il numero di porta utilizza anche gli indirizzi IP per indirizzare il segmento. Gli indirizzi IP sono informazioni presenti nei datagrammi , quindi a livello di rete, le informazioni aggiunte a livello di trasporto sono le porte.

Riassumento ogni socket nell'host di origine deve avere un numero di porta e, quando un segmento arriva all'host ricevente, il livello transport esamina il numero della porta di destinazione e dirige il segmento verso la socket corrispondente.
Le socket TCP sono identificate da quattro parametri, due coppia di end-point *[IP - porta]*: **Indirizzo IP di origine, Porta di origine, Indirizzo IP di destinazione, Porta di destinazione**, invece le socket UDP sono identificate completamente da una coppia che consiste di un **indirizzo IP e di un numero di porta di destinazione.**

Multiplexing e Demultiplexing non orientati alla connessione(con UDP)



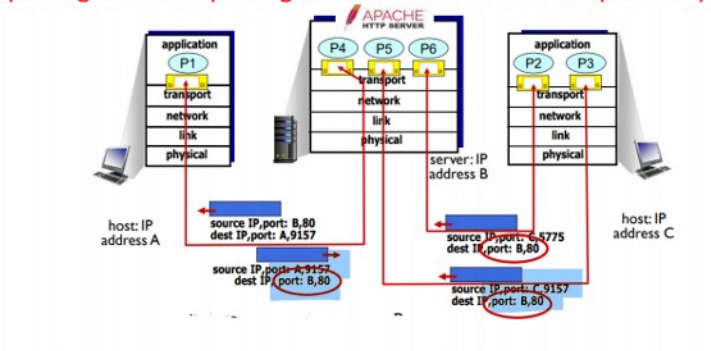
Assegnazione di una nuova porta alla socket

NOTA: < 1024 numero di porta riservate, WEB, FTP
1024 < p < 49151 registered
49152 < p < 65535 dynamic

Abbiamo due host che eseguono rispettivamente i processi applicativi P3 e P4 che inviano dati applicativi al server, che esegue il processo applicativo P1.
Una volta che sappiamo i rispettivi numeri di porta assegnati alle socket, questo è quello che succede:

1. Partiamo dall'host che esegue P3, il suo livello di trasporto crea un segmento che include **tutti i dati applicativi**, **il numero di porta di origine** *(servirà se il destinatario dovrà rispondergli)* e quello di **destinazione** (ed altri campi…);
2. Il segmento creato passa al livello di rete, che lo incapsula in un datagramma IP ed effettua un tentativo di consegna (ricordiamo che il protocollo IP è best-effort);
3. Se il segmento arriva al destinatario, nel nostro esempio al server P1, il livello di trasporto di quest'ultimo esamina il numero di porta di destinazione e lo consegna alla socket corrispondente. Il server potrebbe avere in esecuzione più di un processo (non solo P1), ciascun processo avrà una propria socket UDP e relativo numero di porta.
   (Avviene la stessa cosa tra P4 e P1)

Multiplexing e Demultiplexing orientati alla connessione(con TCP)



Notiamo che diverse porte in diversi client hanno lo stesso numero, non è comunque un problema per il server il demultiplexing perché gli indirizzi IP dei due diversi host sono differenti

Abbiamo 3 segmenti diretti verso l'host server (address) B che esegue i processi P4, P5, P6. Vediamo generalmente cosa succede fra un processo client ed host:

1. L'applicazione server TCP inizialmente, come visto in precendenza, dispone di una "socket di benvenuto" *(welcome socket)* che si pone in attesa di richieste di connessione da parte dei client, quindi tutte le richieste arriveranno sulla porta 12000;
2. Il client TCP crea una socket e genera un segmento per stabilire la connessione. Una richiesta di connessione è un segmento TCP con numero di porta di destinazione 12000 e uno speciale bit di richiesta di connessione posto a 1 nell'intestazione. Il segmento include anche il numero di porta di origine scelto dal client;
3. Il server host B quando riceve una richiesta sulla porta 12000, viene localizzato il processo server in attesa di accettare connessioni su quella porta. Una volta trovato, il processo accetta e crea una nuova connessione creando una nuova socket, identificata da i 4 valori previsti per le socket TCP. Una volta che la connessione è attiva, fino a che questa non viene chiusa, client--server comunicheranno sempre su questo canale;
4. Il server quindi comunica al client che la richiesta è stata accettata dicendogli il suo IP e numero di risposta dove lui dovrà indirizzare i segmenti.

Quindi ogni socket creata tramite accettazione di una connessione è associata ad un diverso client.