**Strato di trasporto**

Lo strato di trasporto si colloca al livello 4 dello strato ISO-OSI e svolge il compito di mettere in comunicazione diversi processi software. Corrisponde al secondo livello TCP/IP.

La comunicazione tra applicazioni avviene con scambi di messaggi che vengono segmentati e trasformati in TPDU, Transport Protocol Data Unit

HTTP FTP SMTP

Livello di rete

TCP UDP

SAP

Livello di applicazione

Livello di trasporto

Entità di trasporto

SAP

Livello di rete

SAP= Service Access Point, interfaccia logica tra due entità una di livello N-1 e l’altra di livello N

I compiti del livello di trasporto sono:

* Aprire, mantenere e chiudere una connessione tra mittente e destinatario
* Suddividere i dati da spedire in tanti segmenti indipendenti
* Riassemblare i dati in arrivo e metterli nel gusto ordine
* Effettuare l’errore recovering di segmenti persi e danneggiati
* Effettuare il controllo del flusso
* Effettuare il multiplexing e de multiplexing
* Risolvere i problemi di efficienza legati all’uso delle risorse di rete

**Il livello di trasporto è caratterizzato da**:

* Servizi che possono essere
	+ Affidabili se eseguono le operazioni nel perfetto ordine
	+ Non affidabili se garantiscono solo l’indirizzamento
* Protocolli che si distinguono in:
	+ UDP User Datagram Protocol – protocollo asincrono che non richiede trasmissione di conferma di ricezione
	+ TCP Trasmission Control Protocol – protocollo sincrono dove è richiesto il messaggio di accettazione dei dati

I protocolli di trasporto sono implementati nei più diffusi sistemi operativi e forniscono ai programmatori le funzioni base dette primitive:

* LISTEN si mette in attesa di richiesta di connessione
* SEND DATA per trasmettere un contatto
* RECEIVE DATA per ricevere un contenuto
* T-CONNECT per aprire una connessione
* T-DISCONNECT per chiudere una connessione

Per ogni primitiva ci sono i seguenti metodi:

|  |  |
| --- | --- |
| **Metodo primitiva** | **Descrizione** |
| request() | Si chiede al servizio di compiere un’azione |
| indication() | Il servizio segnala un evento |
| response() | Si chiede al servizio di rispondere all’evento |
| confirm() | Il servizio segnala l’arrivo di una conferma |

Connessione asincrona o connection-less

Primitiva request

t

Primitiva indication

Connessione sincrona

t

Primitiva confirm

Primitiva request

Primitiva indication

Primitiva response

**L’indirizzamento di trasporto**

Per poter risolvere il problema della trasmissione dei dati tra applicazioni diverse sui medesimi host, il protocollo di trasporto utilizza il meccanismo delle porte.

Una porta è un valore numerico di due Byte che identifica un canale. Esse possono assumere valore da 0 a 65535. L’utilizzo di porte differenti permette più comunicazioni sulla stessa rete. Si viene allora a delineare il concetto di socket

Il socket è un indirizzo numerico formato dall’indirizzo IP e dal numero che individua la porta:

IP locale:porta locale

Una connessione tra due computer viene identificata dalle coppie:

1. Indirizzo IP mittente: porta mittente
2. Indirizzo IP destinatario: porta destinatario

I valori delle porte sono scritti nell’header

I numeri delle porte però non hanno tutti lo stesso significato

* 0-1023 sono porte per applicazioni particolari
* 1024 – 49151 sono porte riservate
* 49152 -65535 sono numeri liberi

Esempi di porte

 21/tcp FTP

22/tcp SSH secure shell

25/tcp SMTP

42 WINS Windows Internet Naming Service

53 DNS

80/tcp http

110/tcp POP3 Post Office Protocol, v3

**Servizio di multiplexing e demultiplexing**

I dati di tutti I processi in esecuzione sull’host vengono segmentati dal livello di trasporto e incalanati verso il livello di rete. Questo processo è detto di multiplexing e, ogni processo è caratterizzato univocamente dalla porta corrispondente. Il demultiplexing è la raccolta dei datagrammi che vengono riassemblati secondo l’ordine prestabilito

**Livelli applicativi**

Processo 4

Processo 3

Processo 2

Processo 1

Processo 4

Processo 3

Processo 2

Processo 1

**Porte**

UDP

TCP

UDP

TCP

IP

**Livello di rete**

**Livello trasporto**

IP

Server

Client

**QoS**

Il livello di trasporto si occupa anche della qualità del servizio Quality of Service

I parametri indicatori della qualità di servizio sono:

* ritardo massimo nell’attivazione della connessione
* numero di byte trasferiti nell’unità di tempo
* velocità di consegna
* probabilità di fallimento della connessione
* probabilità che la connessione non venga stabilita entro il massimo tempo di ritardo
* tasso di errore
* protezione contro le intercettazioni dati
* priorità della connessione

**UDP**

È stato concepito per tutte quelle applicazioni per le quali non è necessaria una completa gestione delle connessioni.

 Non è richiesto un messaggio di accettazione, acnowledgement e nemmeno di un messaggio di handshaking dove i dispositivi definiscono i protocolli e le velocità di trasmissione. Gestisce solo l’indirizzamento, la multiplazione ed il controllo dell’errore.

Le applicazioni sono:

telefonia voip, protocolli di instradamento RIP, risoluzione dei nomi DNS, amministrazione di rete SNMP, file server remoti NFS, network time protocol

 Un datagram UDP è fatto nel seguente modo:

IP Header (20 Bytes) UDP header (8 Bytes) UDP data

Es. Il server ha indirizzo 130.130.12.17 e punto di accesso 309. Il server manda in esecuzione l’applicazione e si mette in attesa fino a che il client non invia un datagram

Il server ha socket <130.130.12.17:309>

Quando il server riceve un messaggio:

* legge il numero di porta del mittente
* estrae il messaggio contenuto nel segmento
* invia il messaggio al socket specificato
* es: sia un destinatario A con socket <9.12.0.54:300> e destinatario B con socket <137.200.70.14:3010>

**IGMP Internet Group Management Protocol.** Protocollo che si trova su host e router per la partecipazione a un gruppo indicando il proprio indirizzo IP e impostandosi un indirizzo multi cast. I ruouter IGMP inviano pacchetti di aggiornamento per illustrare agli altri router i gruppi di appartenenza. IGMP è però un protocollo del livello 3

**Il servizio di trasferimento affidabile: connessione TCP**

Lo strato di trasporto attua meccanismi che permettono di eliminare eventuali problem.i Un servizio di trasferimento si dice affidabile se:

* tutti i messaggi sono consegnati a destinazione e giungono privi di errori
* ciascun messaggio è consegnato una e una sola volta
* i messaggi sono consegnati nello stesso ordine in cui sono inviati

L trasmissione deve essere:

* priva di errori
* senza perdita di dati
* senza duplicazioni nella consegna dei segmenti

 ***meccanismi impiegati***

I meccanismi impiegati per realizzare un trasferimento affidabile sono i seguenti:

* numerazione dei segmenti trasmessi e trasmissione dei segnali di riscontro con numero di sequenza
* impiego di temporizzazione
* impiego di finestre in trasmissione e ricezione

Numerazione dei segmenti trasmessi

* I dati sono una sequenza di bit e, sono una sequenza ordinata
* I dati vengono suddivisi in pacchetti formati da un certo numero di byte
* Ogni pacchetto ha un numero d’ordine che corrisponde al numero d’ordine del primo bit in esso contenuto ed è detto SN, sequence number
* Il pacchetto viene inizializzato da un numero detto ISN=0 Initial Sequence Number
* Il primo byte avrà sequence number SN=ISN+1
* Il pacchetto successivo avrà SN=SN+k, dove k è il numero di byte del pacchetto
* Nell’header viene scritto l’indirizzo Ip del mittente e della relativa porta mittente + l’indirizzo Ip del destinatario e della relativa porta

**Handshaking a tre vie**

La connessione avviene tramite handshaking cioè tramite lo scambio di messaggi di controllo. Questo tipo di connessione si chiama punto punto perché si crea un collegamento diretto tra client e server.

La prima fase della connessione è quello dell’invio di un pacchetto dati tra due computer per stabilire i criteri di connessione. Successivamente, avviene il trasferimento dati bidirezionale.

Ogni pacchetto viaggia attraverso un canale che si crea ogni volta tra client e server. Il server può creare diverse connessioni contemporaneamente ma ognuna, è identificata da un unico socket

Host B

Host A

1 Send SYN=1

ACK=0 ;SeqN=x

1 SYN received

2 Send SYN=1 SeqN=y

ACK =1; ACKN=x+1

 2 SYN received

3 Send SYN=0 SeqN=x+1

ACK =1; ACKN=y+1

3 Estabilished

Fase 1: il client invia un segmento TCP iniziale con:

* Source port (la propria porta)
* Destination port (la porta del server)
* SYN=1
* Capacità di pacchetto MSS (Maximum Segment Size)

Fase 2: in risposta, il server invia un segmento TCP:

* Source port(la propria porta)
* Destination port (la porta del server)
* SYN=1
* ACK=1
* Capacità di pacchetto MSS
* Delle due capacità di pacchetto, sarà scelta quella inferiore

Fase 3:

* Source port e destination port sono ormai stabilite
* Il numero di sequenza del client è x e il successivo è x+1
* Il numero di sequenza del client è x e il successivo è y+1

Es di trasmissione andata a buon fine

Host B

Host B

Host A

Host A

|  |  |
| --- | --- |
| 192.168.0.5:50000 | 10.2.3.7:80 |

Seq= 56,20 B

ACKn=156

ACKn=76

ACKn=76

Seq= 76,80 B

Seq= 56,20 B

|  |  |
| --- | --- |
| 10.2.3.7:80 | 192.168.0.5:50000 |

Seq= 56,20 B

Es di ricezione di dati mancanti

Host B

Host A

Seq=800,560B

Seq=4,560B

***Temporizzazione della trasmissione***

Per ciascun segmento inviato, TCP avvia un timer detto timer di ritrasmissione RTO Retrasmission Time Out indicato come time out; indica dopo quanto tempo deve considerarsi perso l’ultimo segmento trasmesso e quindi, bisogna ritrasmetterlo se nessun ACK viene pervenuta.

Host B

Host A

Host B

Host A

ACKn=529

ACKn=529

Seq=73,456B

Seq=73,456B

ACKn=529

RTO

RTO

ACKn=529

Seq=73,456B

Seq=73,456B

Gli esempi precedenti evidenziano che nel primo caso, il segnale ACK non giunge all’host mittente e, il segnale viene inviato di nuovo quando scade il tempo RTO; nel secondo caso, il segnale ACK giunge al mittente ma non nel tempo utile RTO e quindi, i dati vengono inviati di nuovo.

Il time di Keepalive è un ulteriore temporizzazione. Inizia il conteggio alla ricezione di ogni pacchetto e dichiara scaduta la connessione se c’è un tempo di inattività

Timed wait è il tempo che viene atteso prima di disconnettere realmente ed è pari al doppio del tempo di vita del pacchetto

**Finestra di trasmissione e ricezione**

Per effettuare la gestione dei segnali inviati e ricevuti il terminale mittente mette a disposizione una finestra di trasmissione.

La finestra di trasmissione viene gestita come una struttura a coda utilizzando due variabili: sendbase e nextseqnum

* Sendbase= il numero d’ordine del byte più vecchio tra quelli trasmessi ma di cui non si conosce l’esito. È l’estremo inferiore della finestra
* Nextseqnum= il numero d’ordine del prossimo byte che deve essere ancora trasmesso

Ws

nextseqnum

Send\_base

Ws(byte)=larghezza della finestra