Il livello di trasporto

Liceo G.B. Brocchi – Bassano del Grappa (VI)
Liceo Scientifico – opzione scienze applicate
Giovanni Mazzocchin

Il livello di trasporto

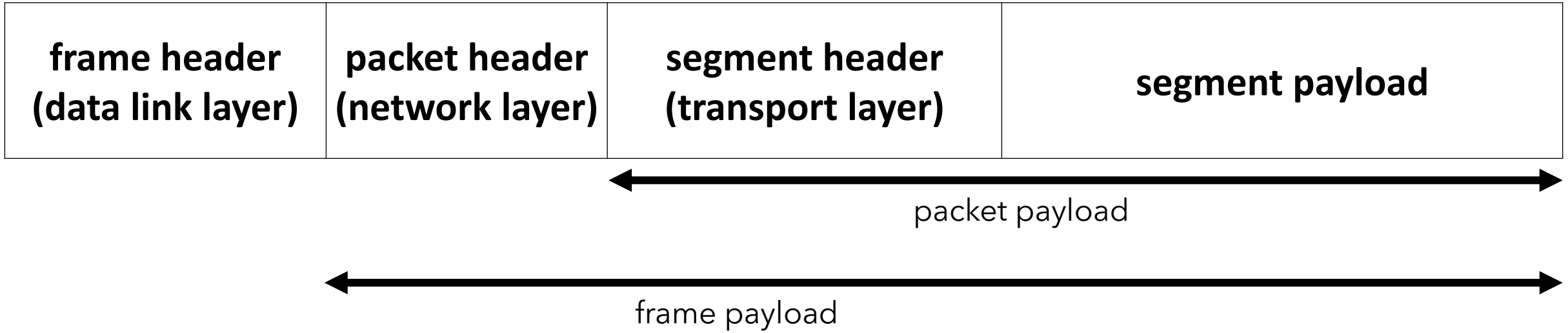
- Il **livello di trasporto** (*transport layer*) fornisce un servizio di trasmissione dei dati da un processo su una macchina sorgente ad un processo su una macchina destinazione con un livello di affidabilità indipendente dalla rete su cui opera
- È il livello su cui si basano le applicazioni: questo livello fornisce infatti un'interfaccia di procedure (funzioni) utilizzate dai programmi per colloquiare in rete in modo (opzionalmente) affidabile e orientato alla connessione. Non opera nella rete, ma negli *end host* (macchina sorgente e macchina destinataria)
- Lo scopo del livello di trasporto è fornire un servizio (opzionalmente) affidabile su reti inaffidabili: una sorta di canale perfetto simulato dal protocollo, che permette ai byte inviati dal processo sorgente di arrivare intatti e ordinati al processo destinatario

Il livello di trasporto

primitiva	pacchetto inviato	significato
LISTEN		attendi che un processo si connetta
CONNECT	CONNECTION REQUEST	prova a stabilire una connessione
SEND	DATA	invia dati
RECEIVE		attendi che arrivi un pacchetto DATA
DISCONNECT	DISCONNECTION REQUEST	richiedi il rilascio della connessione

- è evidente che il servizio fornito è **connection-oriented**
- i vari passaggi ricordano a grandi linee una chiamata telefonica
- IP era connection-oriented o connectionless?

Il livello di trasporto



TCP, UDP

- I protocolli di trasporto più importanti dello stack TCP/IP sono:
  - **TCP** (***T****ransmission* ***C****ontrol* ***P****rotocol*): fornisce un byte stream *reliable* (con error correction e flow control end-to-end) su una rete *unreliable* (a discapito della velocità del trasferimento dati). Utilizzato nel trasferimento di file e in molte altre applicazioni. Il servizio è connection-oriented
  - **UDP** (***U****ser* ***D****atagram* ***P****rotocol*): protocollo *leggero* sopra IP. Aggiunge solo le **porte** utilizzate per connettere le applicazioni. Caratteristiche: connectionless, no congestion control, no flow control, no ritrasmissioni, massimizzazione della velocità di trasferimento. Utilizzato in: DNS, live streaming, online gaming

TCP, UDP

- TCP e UDP risiedono nei sistemi operativi degli host
- Offrono servizi a diversi protocolli dello strato superiore (*Application*), associando a ciascuno di essi un **port number** di 16 bit
- *Source port* e *Destination port* sono inserite nell'header del segmento TCP (verificare con Wireshark)
- I numeri di porta sono 2^{16}, suddivisi in 3 intervalli:
  - `0 – 1023`: *well known ports*, riservate da IANA alle applicazioni più comuni
  - `1024 – 49151`: *registered ports*
  - `49152 – 65535`: non sono soggette ad alcun vincolo
- Su Linux: `cat /etc/services`

Porte e applicazioni

port	protocol	use
20, 21	FTP	file transfer
22	SSH	secure remote login
23	telnet	remote login
25	SMTP	email
80	HTTP	www
143	IMAP	email access
443	HTTPS	secure web

Socket

- **Socket**: `<IP address:port number>`
- Una connessione logica TCP è identificata dalla coppia **(socket sorgente – socket destinazione)**
- Stream sockets (**TCP**) vs Datagram sockets (**UDP**)
- I socket sui sistemi Unix si comportano un po' come i file: sono caratterizzati da un descrittore, è possibile aprirli, chiuderli, scriverci sopra, leggerli etc…
- Facciamo qualche esperimento con i **Berkeley Sockets**, una API (***A****pplication* ***P****rogramming* ***I****nterface*) per la programmazione di rete, disponibile su tutti i sistemi UNIX-like e non solo

Socket

- Su Linux:
  - `man socket`
  - `man listen`
  - `man bind`
  - `man accept`
  - `man send`
  - `man recv`

- Il comando `netstat -a` permette di visualizzare i socket attivi e gli stati delle connessioni
- Il tool `nmap` permette di effettuare il *port scanning*, ossia l'individuazione delle porte aperte su un computer bersaglio