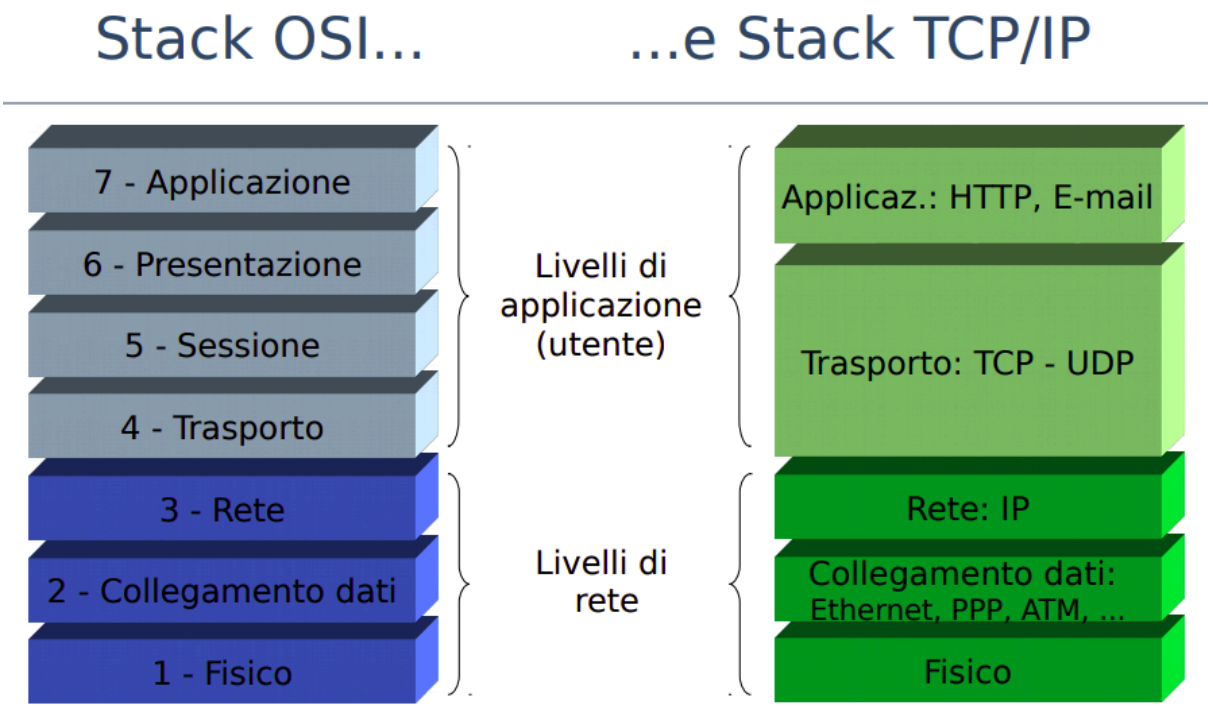
Introduzione ---> Ripasso delle Reti



Livelli TCP/IP:

- **Fisico**: trasmissione fisica tramite cavi e bit interpretati da segnali elettrici;
- **Data Link**: scendendo un gradino nella pila protocollare, troviamo il livello data link, il responsabile della comunicazione efficiente e ordinata all'interno della stessa rete fisica. Immaginate questo livello come il sistema che gestisce le conversazioni e gli scambi tra dispositivi che si trovano nella stessa "stanza" o segmento di rete, che sia una LAN tradizionale via cavo (Ethernet) o una rete senza fili (Wi-Fi, Bluetooth). In questo contesto, il modo in cui i dati vengono trasmessi e gestiti è cruciale. Ethernet, ad esempio, un tempo si affidava a meccanismi come il CSMA/CD (Carrier Sense Multiple Access with Collision Detection) per evitare "scontri" di dati sul cavo condiviso. Oggi, con l'avvento degli switch, le collisioni sono quasi un ricordo, poiché ogni dispositivo ha una connessione dedicata al suo interno, rendendo la comunicazione più diretta ed efficiente. Il Wi-Fi, invece, operando via etere, non può "sentire" le collisioni mentre trasmette e continua a usare un approccio come il CSMA/CA (Collision Avoidance), dove i dispositivi "ascoltano" e aspettano il loro turno per parlare, cercando di prevenire le collisioni anziché rilevarle. A questo livello, i dati viaggiano raggruppati in unità chiamate PDU (Protocol Data Unit), che sono fondamentalmente "pacchetti" o "raggruppamenti di bit/byte". Per assicurarsi che ogni pacchetto arrivi al destinatario giusto all'interno di questa stessa rete locale, si lavora

principalmente con gli indirizzi MAC. Ogni dispositivo di rete ha il suo indirizzo MAC, un identificatore unico di 48 bit, solitamente rappresentato come 6 coppie di caratteri esadecimali separati da due punti (es. 00:1A:2B:3C:4D:5E). Possiamo pensare al router come alla "porta" che connette la nostra "stanza" (la rete locale) al mondo esterno. All'interno di questa porta, ci sono delle "maniglie", le interfacce gateway (in particolare quella di default), che sono i punti di ingresso e uscita dalla stanza stessa. Quando un pacchetto viene inviato all'interno della rete locale, il suo primo campo nell'intestazione è proprio l'indirizzo MAC del destinatario. Questo è un meccanismo estremamente efficiente: le schede di rete di tutte le macchine presenti sulla stessa LAN "leggono" solo questo primo campo. Se l'indirizzo MAC non corrisponde al proprio, possono ignorare il pacchetto immediatamente e senza alcuno sforzo ulteriore di lettura o elaborazione da parte del sistema operativo (SO). Questo evita inutili sprechi di tempo e risorse computazionali, garantendo che solo la macchina destinata debba "aprire" e processare quel pacchetto specifico;

- **Rete**: connessione tra diverse reti locali, tramite instradamento e indirizzi IP (protocollo v4 32 bit, v6 128 bit);
- **Trasporto**: esegue un enumerazione per il trasporto di PDU che devono essere divisi in sotto porzioni gestendo, tramite protocolli come TCP (non dire che è sicuro, bensì affidabile) e UDP, la perdita di PDU e il loro recupero;
- **Applicazione**: questo livello ne ingloba 3 dell'OSI perché troppi livelli introducono complessità ed inefficienza inutile (più header, più costi e più tempi). Attraverso il sistema di porte introdotte dal livello di trasporto, siamo in grado di far comunicare i precisi processi (il DNS è un protocollo di questo livello e non del livello di rete, ed è un esempio di tecnologia che utilizza UDP).

N.B: un protocollo è una convenzione; più un pacchetto scende tra i livelli più si gonfia per via degli header.

STRUMENTI DI ANALISI DELLA RETE

Esistono diversi strumenti SW che consentono di analizzare ed eseguire diagnostica di PDU che arrivano sulla propria interfaccia di rete:

- TCPDUMP, storico tool da linea di comando (per OS Linux);
- WinDump, storico tool da linea di comando (per OS Windows);
- Wireshark, moderno tool con GUI disponibile per Linux, Windows e Mac.

Noi chiaramente useremo il terzo. Le principali funzionalità di questa libreria sono la possibilità di cercare e trovare interfacce di rete, la gestione avanzata di filtri di cattura e la gestione degli errori e statistiche di cattura.

Catturano fisicamente i PDU e li interpretano. Le due funzionalità (cattura e interpretazione) sono distinte, ossia si può catturare il traffico e analizzare in un secondo momento, allo stesso modo analizzare del traffico precedentemente catturato.

Mentre l'interpretazione avviene ad un livello più astratto e che permette all'utente una visualizzazione e una personalizzazione migliore, la parte di cattura coinvolge elementi di basso livello come le interfacce di rete (ricordarsi che sono molteplici wifi-bluetooth, eth., NFC, USB-C 3.0, …). In particolare la 'lo' (loopback) è una rete fittizia sempre presente che rende possibile la comunicazione tra processi della stessa macchina come fossero su diverse reti.

In particolare la parte che esegue la cattura (o sniffing) dei PDU è svolta da una libreria a parte scritta in C chiamata libpcap, che scavalca il SO prendendosi una copia di tutte le PDU in arrivo su una determinata interfaccia di rete prima che il SO ignori ed elimini i PDU che normalmente non sarebbero destinati alla macchina in uso. Pertanto necessita i privilegi di root (sudo), dato che deve lavorare a stretto contatto con l'hw (attivazione flag di mod. promiscua -- vedi primo punto). Esistono due macro-casi per questa operazione:

- sniffing all'interno di reti non-switched: in questa tipologia di reti il mezzo trasmissivo è condiviso e, quindi, tutte le schede di rete dei PC ricevono tutti i PDU, anche quelli destinati ad altri. I propri, invece, sono selezionati a seconda del MAC. Lo sniffing, in questo caso, consiste nell'impostare sull'interfaccia di rete la cosiddetta **modalità promiscua** che disattiva il "filtro hardware" basato sul MAC. Così facendo, si permette al sistema l'ascolto di tutto il traffico passante sul cavo. Un esempio di rete non-switched è la rete WiFi;
- sniffing all'interno di reti Ethernet switched: in questo caso, invece, l'apparato centrale della rete (definito switch), si preoccupa di inoltrare su ciascuna porta solo il traffico

destinato ai dispositivi collegati a quella porta. Quindi, ciascuna interfaccia di rete, riceve solo i PDU destinati al proprio indirizzo, i PDU multicast e quelli broadcast. L'impostazione della modalità promiscua è, pertanto, insufficiente per poter intercettare il traffico in una rete gestita da switch.

In WSH ci sono due tipi di filtri:

- filtro di cattura: si utilizza associando un espressione booleana sulla base dei campi dei PDU; avviene prima della cattura effettiva, in quanto lavora ancora una volta insieme alla modalità promiscua volta ad alleggerire il costo della cattura. In questo modo, i PDU che non rispettano l'espressione richiesta vengono completamente ignorati. Per iniziare una cattura basta andare sul menù a tendina 'cattura' e selezionare 'opzioni'; da qui selezionare l'interfaccia desiderata e sulla parte bassa un filtro di cattura per scremare eventuali pacchetti indesiderati;
- filtro di visualizzazione: a differenza del precedente, i PDU ormai sono stati catturati (post cattura), ma serve solo nella mera visualizzazione/ricerca di determinati PDU (non eliminando quindi i PDU che non rispecchiano l'espressione di questo filtro); legato a questo, è possibile anche colorare i PDU per una migliore visualizzazione sulla base di determinate caratteristiche del PDU stesso;

Comandi di utilizzo generale da terminale:

- **ping**: è un semplice strumento per verificare la raggiungibilità di un computer connesso alla rete e il relativo Round Trip Time (RTT), ossia il tempo che intercorre dalla partenza del pacchetto inviato al ritorno della risposta. Per questa operazione viene utilizzato il protocollo ICMP (Internet Control Message Protocol è un protocollo di servizio per trasmettere informazioni riguardanti malfunzionamenti, informazioni di controllo o messaggi tra i vari componenti di una rete di calcolatori);
- **traceroute**: il comando traceroute è, invece, un semplice strumento per tracciare il percorso che un pacchetto segue dalla sorgente alla destinazione. Il comando mostra un elenco di tutte le interfacce dei router che il pacchetto attraversa finché non raggiunge la destinazione. Si noti la presenza di alcuni asterischi in corrispondenza di determinate tappe. Questi sono dovuti al fatto che, certe interfacce di specifici router, non forniscono alcuna informazione. Questa scelta viene presa dagli amministratori di rete per evitare di svelare la topologia di rete a possibili malware. In tal caso traceroute non può mostrare tali passi del percorso (il protocollo è sempre ICMP). Come per le telefonate spam che cercano di capire se il nostro numero di telefono è ancora attivo per venderlo e fare pubblicità, il protocollo ICMP è spesso bloccato dai

network administrator per motivi di sicurezza (in particolare riguardo l'intasamento della rete);

- **nslookup**: il comando nslookup consente di effettuare una interrogazione ai server DNS per poter ottenere da un hostname il relativo indirizzo IP, o viceversa. Si può utilizzare in due modalità interattivo e non interattivo. DNS (Domain Name System) è un sistema di server organizzato gerarchicamente, per la gestione del namespace (Domain Name Space). Il compito principale di questo servizio è quello di rispondere alle richieste della risoluzione del nome di dominio, ovvero la conversione dei nomi di dominio in indirizzi IP. Nello specifico la modalità interattiva permette di effettuare più interrogazioni e visualizza i singoli risultati; viene abilitata in maniera automatica quando il comando non è seguito da alcun argomento. La modalità non interattiva invece permette di effettuare una sola interrogazione visualizzandone il risultato, e si abilita ogni qualvolta si specifichi l'host-to-find;
- **ifconfig**: il comando ifconfig è un utilizzato per configurare e controllare un'interfaccia di rete TCP/IP da riga di comando. L'esecuzione del comando con l'opzione -a mostra a video le informazioni di tutte le interfacce di rete;
- **route**: il comando route è utilizzato per visualizzare e modificare le tabelle di routing;
- **whois**: il comando whois consente, mediante l'interrogazione di appositi database server da parte di un client, di stabilire il nome del privato, azienda o ente al quale è intestato un determinato indirizzo IP o uno specifico dominio DNS. Nel Whois vengono solitamente mostrate anche informazioni riguardanti l'intestatario, data di registrazione e la data di scadenza. Whois si può consultare tradizionalmente da riga di comando, anche se ora esistono numerosi siti web che permettono di consultare gli archivi dove sono contenute tali informazioni (non incappare nella somiglianza con nslookup, qui si inserisce solo il sito senza www, mentre nell'altro si).

ESERCITAZIONE

PARTE 1 (file capture.cap):

1. Che tipo di protocollo di livello Data-link è utilizzato? Come fa Wireshark a capirlo?

RISPOSTA:

Il protocollo utilizzato è Ethernet. Lo possiamo vedere dal primo campo della sezione 'Frame' del (in generale dal livello inferiore a quello interessato è contenuto il campo che specifica il protocollo utilizzato dal livello superiore; in questo modo il campo header-length diventa inutile in quanto è standard dato un determinato protocollo);

2. Disegnare la PDU di livello Data-link indicando il valore dei vari campi.

RISPOSTA:

```
--------+----------------------+----------------------+---------------+----------
| MAC Destinazione (6B)   | MAC Sorgente (6B)    | Type (2B) |
--------+----------------------+----------------------+---------------+----------
```

Destinazione: broadcast (ff:ff:ff:ff:ff:ff);
Mittente: 00:e0:81:24:dd:64;
Type: IPv4 //indica il protocollo del livello superiore;

NB: quando apri un pacchetto visualizzi per righe i vari livelli TCP/IP del pacchetto partendo da quello fisico e andando in giù.

3. Qual è il MAC sorgente? Di che tipo è: unicast o broadcast?

RISPOSTA:

Lo ho già scritto prima, è di tipo unicast;

4. Qual è il MAC destinazione? Di che tipo è: unicast o broadcast?

RISPOSTA:

Lo ho già scritto prima, è di tipo broadcast;

5. Che tipo di protocollo di livello Network è utilizzato? Come fa Wireshark a capirlo?

RISPOSTA:

Il protocollo utilizzato è IPv4. Lo possiamo capire da uno dei campi indicati nel livello inferiore, in questo caso il campo Type del Data-Link;

6. Qual è la lunghezza dell'header IP?
RISPOSTA:
20 bytes;

7. Quali sono gli indirizzi IP sorgente e destinazione?
RISPOSTA:
L’indirizzo mittente è 157.27.252.223 mentre l’indirizzo destinatario è 157.27.252.255;

8. Che tipo di protocollo di livello trasporto è contenuto in IP? Come fa Wireshark a capirlo?
RISPOSTA:
Il protocollo del livello di trasporto è UDP, ed è capibile dal relativo campo nel livello di rete;

9. Quali sono le porte sorgente e destinazione a livello di trasporto?
RISPOSTA:
Porta 631 per entrambe le domande;

10. Creare un filtro per visualizzare solo i pacchetti che hanno ARP come protocollo (suggerimento: basta scrivere arp nella barra Filter sotto la toolbar; si ricordi di premere su Apply dopo aver scritto arp).
RISPOSTA:
Fatto;

11. Dopo aver applicato il filtro precedente qual è la percentuale di pacchetti che rimangono visualizzati rispetto al totale? (suggerimento: vedere entrambi i valori nella barra di stato in basso).
RISPOSTA:
Pacchetti totali 272, visualizzato dopo il filtro sul protocollo ARP 173 (63,6%);

12. Creare un filtro per visualizzare solo i pacchetti che hanno destinazione MAC 00:01:e6:57:4b:e0. (suggerimento: usare l'editor di espressioni; la categoria da selezionare è Ethernet; per 13 l'indirizzo MAC usare la notazione esadecimale con i due punti come separatori; si ricordi di premere su Apply dopo aver creato l'espressione).
RISPOSTA:
filtro: eth.dst == 00:01:e6:57:4b:e0

13. Dopo aver applicato il filtro precedente qual è la percentuale di pacchetti che rimangono visualizzati rispetto al totale? (suggerimento: vedere entrambi i valori nella barra di stato in basso).
RISPOSTA:
1 su 272, ossia il 0.4%;

14. Creare un filtro per visualizzare solo i pacchetti che hanno destinazione MAC broadcast. (suggerimento: nell'editor di espressioni la categoria da usare è Ethernet; per l'indirizzo MAC usare la notazione esadecimale con i due punti come separatori; si ricordi di premere su Apply dopo aver creato l'espressione).
RISPOSTA:
filtro: eth.dst == ff:ff:ff:ff:ff:ff

15. Dopo aver applicato il filtro precedente qual è la percentuale di pacchetti che rimangono visualizzati rispetto al totale? Sono molti? Perché?
RISPOSTA:
Sono 228 su 272, ossia il 83,8% del totale. Sono molti perché la maggior parte delle richieste sono volte a scoprire chi ha un determinato indirizzo IP (ARP request);

PARTE 2 (file simpleHTTP.cap):

1. Colorare di rosso tutti i pacchetti che contengono UDP e di verde tutti i pacchetti che contengono TCP. (suggerimento: nell'editor delle regole di colorazione è sufficiente portare in alto due regole già esistenti e modificarle per cambiarne i colori di sfondo).
RISPOSTA:
Su ‘Visualizza’ trovi la dicitura ‘Regole di colorazione’, selezioni il tipo di protocollo da colorare e poi in fondo premi su ‘sfondo e scegli il colore, poi ok.

2. Cosa contengono i primi due pacchetti della sessione di cattura? IP sorgente, IP destinazione. Tipo di protocollo di trasporto. Tipo di protocollo di livello Applicazione. Come fa Wireshark a capirlo? Messaggio contenuto nel Payload di livello applicazione.
RISPOSTA:

- Pacchetto 1:
    1. Source Address: 157.27.252.202
    2. Destination Address: 157.27.10.10
    3. UDP;
    4. DNS;

5. c’è scritto;
6. www.polito.it;

- Pacchetto 2:
    a. Source Address: 157.27.10.10
    b. Destination Address: 157.27.252.202
    c. UDP;
    d. DNS;
    e. c’è scritto;
    f. 130.192.73.1

3. Prendere in considerazione il pacchetto n.3. per IP sorgente, IP destinazione. Tipo di protocollo di trasporto. IP sorgente e destinazione sono in qualche modo collegati con i messaggi scambiati a livello applicazione nei primi due pacchetti? È possibile fare delle ipotesi su cosa serve il protocollo di livello applicazione dei primi due pacchetti?
RISPOSTA:
Il protocollo del livello di trasporto è TCP, e non sono collegati con i pacchetti precedenti in quanto erano di una connessione con protocollo UDP. L’unico collegamento che si può fare è semantico e a livello di applicazione, in quanto i primi due pacchetti erano una richiesta DNS per l’IP del sito www.polito.it, mentre questo terzo pacchetto è una richiesta di connessione TCP proprio a quel sito (ossia indirizzo 130.192.73.1).

4. Creare un filtro per visualizzare solo i pacchetti TCP (compresi i pacchetti HTTP) e determinare il numero.
RISPOSTA:
Filtro: tcp && http ---> 134 su 823 (16,3%) con condizione AND, oppure Filtro: tcp || http con OR abbiamo 807 su 823 (98,1%);

5. Prendere in considerazione il pacchetto n. 6. IP sorgente, IP destinazione. Tipo di protocollo di trasporto. Tipo di protocollo di livello Applicazione. Perché prima della trasmissione del primo messaggio HTTP c'è lo scambio di tre pacchetti puramente TCP? Quali sono i flag settati nell'header TCP di questi tre pacchetti?
RISPOSTA:
TCP trasporto, HTTP applicazione. Lo scambio di pacchetti è il three-hands-shake per avviare la comunicazione affidabile. Le flag sono SYN, SYN-ACK e ACK.

5. Creare un filtro per visualizzare solo i pacchetti TCP (esclusi i pacchetti HTTP) e determinarne il numero. Qual è la percentuale sul totale dei pacchetti TCP trovata al punto 5? A cosa servono tali pacchetti? Se il protocollo DNS dei pacchetti 1 e 2 avesse usato il protocollo TCP, quanti pacchetti IP sarebbero stati generati? Sarebbe stato utile?
RISPOSTA:
Filtro: tcp && !http ---> 81,8% sul totale, sono tutti three-hands-shake. Se anche la richiesta DNS avesse usato TCP, i pacchetti totali sarebbero stati 5 (3 per l'apertura connessione e i 2 già presenti); no, non sarebbe servito a nulla in quanto la richiesta al DNS di un dominio non richiede l'utilizzo di un protocollo affidabile come TCP;

6. Selezionare il pacchetto 3 e seguire lo stream TCP col comando da menu Analyze/Follow TCP Stream. Cosa si può leggere? Qual è il messaggio contenuto nel payload della PDU di livello Applicazione?
RISPOSTA:
Analizza - Segui - TCP stream, permette di vedere le richieste HTTP (GET) più nel dettaglio. In particolare nel payload è possibile vedere tutto il codice HTML della pagina polito.it richiesta, compresa di immagini e quant'altro.

PARTE 3 (file busyNetwork.cap):

1. Elencare i protocolli di livello Applicazione che entrano in azione in questa cattura classificandoli in base al livello Trasporto utilizzato.
RISPOSTA:
HTTP (usa TCP), DNS (usa UDP), FTP, SSH;

2. Provare ad analizzare diversi stream TCP con sopra diversi protocolli di livello applicazione. Che differenza c'è tra il contenuto trasmesso in una connessione TCP per il protocollo FTP e quello trasmesso per il protocollo SSH?
RISPOSTA:
Per FTP (file transport) la porta usata è la 21 e la maggior parte dei dati trasmessi è facilmente visibile una volta catturato il traffico, mentre per SSH (secure shell) la porta è la 22, e i dati sono tutti crittografati, quindi illeggibili senza chiave di decriptazione.

PARTE 4 (file pingCapture.cap):

1. Individuare le richieste ping (pong) inviate e le relative risposte. Quante sono?
RISPOSTA:

basta mettere nel filtro ‘icmp’ e sono 22;

2. Quali sono IP sorgente e destinazione della richiesta ICMP? A quale ente o azienda sono intestati?
RISPOSTA:
univr;

3. Provare a invocare il comando ping dal proprio PC verso www.google.com e verso il proprio Default Gateway (come faccio a sapere il suo IP?) e osservare il RTT medio e la sua variazione. Chi mostra la media più grande? Perché? Chi mostra la variazione più grande? Perché?
RISPOSTA:
--- www.google.com ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9014ms
Per trovare invece il default Gateway immettiamo il comando ‘ip route’ e vediamo ‘default via 192.168.1.1’. Eseguendo ping --->
--- 192.168.1.1 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10019ms

PARTE 5 ():
Entrare nel sistema Linux presente in cloud e digitare il comando traceroute www.google.com. Individuare le interfacce dei router attraversati. Individuare i nomi delle organizzazioni a cui sono intestati gli IP delle interfacce dei router attraversati.
RISPOSTA:
Dopo esserci connessi alla vpn dell’uni, ho avviato ‘traceroute www.google.com’ , e come risultato ho ottenuto:
1 * * *
2 10.252.10.1 (10.252.10.1) 48.837 ms 48.778 ms 48.657 ms
3 198.51.100.129 (198.51.100.129) 48.730 ms 51.017 ms 48.602 ms
4 ru-univr-l1-rl1-vr00.vr00.garr.net (193.204.218.109) 48.478 ms 48.434 ms 48.397 ms
5 * * *
6 rs1-mi01-re1-mi02.mi02.garr.net (185.191.180.158) 48.167 ms 48.282 ms 48.174 ms
7 142.250.164.230 (142.250.164.230) 48.143 ms 48.071 ms 142.250.174.46 (142.250.174.46) 48.041 ms
8 192.178.104.191 (192.178.104.191) 48.976 ms 192.178.104.103 (192.178.104.103) 48.537 ms *

9 142.251.235.177 (142.251.235.177) 48.405 ms 192.178.82.62 (192.178.82.62) 48.461 ms 142.251.235.177 (142.251.235.177) 48.308 ms 10 142.251.235.177 (142.251.235.177) 48.263 ms mil04s51-in-f4.1e100.net (142.251.209.36) 48.184 ms 142.251.235.179 (142.251.235.179) 49.466 ms di cui le interfacce sono quelle esplicitate nelle parentesi dopo l'IP di riferimento del router attraversato. Per le aziende associate basta lanciare l'ip del gestore con whois;

PARTE 6():

Cercare quali interfacce sono attualmente attive sul proprio PC. Qual è l'indirizzo IP dell'interfaccia che state utilizzando sul vostro host? E la netmask corrispondente? Qual è l'indirizzo IP di www.univr.it?

RISPOSTA:

Lancio nel terminale 'ifconfig' e vediamo subito le mie interfacce attive, ossia:

- wlo0 ---> connessione wifi;
- lo ---> loopback;
- enx527ac5ca44bc ---> non ne ho idea;
- eno1 ---> connessione ethernet;

Al momento sto utilizzando wlo0 e il mio indirizzo IP è 172.20.10.4, mentre la netmask corrispondente è 255.255.255.240. Infine con nslookup www.univr.it è possibile risalire a 157.27.3.60, ossia l'indirizzo IP dell'uni.

N.B: in WSH vediamo sulla dicitura protocol il nome del protocollo di livello più alto che WSH stesso è riuscito a trovare e individuare. Quindi se vediamo per esempio TCP, significa che il pacchetto non ha (da solo magari unito ad altri sì) una parte di livello applicativo interpretabile allo stato attuale. Questa nota è particolarmente importante per esempio nel caso in cui leggessimo la flag FYN, comunemente associata al protocollo TCP, in un pacchetto interpretato come HTTP- Web Socket. In questo caso non è la flag FYN di chiusura connessione (ossia, lo è del Web Socket ma non della connessione intera TCP), ma potrebbe proprio significare completamente un altra cosa in base al protocollo a cui fa riferimento.

Interfaccia Socket

Le applicazioni di rete sono insiemi di processi su host diversi che si scambiano messaggi attraverso la rete. Esistono degli schemi base che regolano lo scambio di messaggi:

1. Client/server;
2. Publisher/Subscriber (Pub/Sub).

Client/Server:

E' la situazione più frequente. Il client fa sempre il primo passo con una richiesta, mentre il server fa il secondo passo e manda la risposta, rimettendosi poi in attesa di altre richieste. La richiesta del client può essere una richiesta di un dato oppure la trasmissione di un dato (cioè la richiesta di prendere in consegna un certo dato).

Quello che determina il ruolo di client e server è l'ordine dei messaggi e non il contenuto. In un'applicazione client/server, il client è quindi colui che fa il primo passo indipendentemente dal fatto che chieda o trasmetta un dato.

Dobbiamo anche fare attenzione ai nomi: client e server sono processi e non host; l'insieme di almeno un client e un server costituisce l'applicazione di rete. In particolare si può dire che un certo processo gioca/interpreta il ruolo di client o server all'interno dell'applicazione.

Durante l'esercitazione non useremo direttamente l'interfaccia socket, bensì una libreria sviluppata da un ex studente durante la sua tesi che incapsula l'utilizzo diretto della socket per non andare eccessivamente a basso livello (function wrapped).

Il programma, prima di usare la rete, deve creare un oggetto di tipo socket. Il socket è l'insieme di chiamate a funzione (punto di comunicazione) identificato da 3 parametri, ossia indirizzo IP locale, porta locale, modalità di trasmissione UDP oppure TCP. Funziona come end-point, ossia in una connessione c'è una socket sia in un host che nell'altro. Il programma server deve decidere esplicitamente il numero di porta locale affinché i client possano saperlo (la porta serve a indicare quale processo, l'IP quale host); i numeri da 0 a 1023 sono riservati a protocolli applicativi noti. Il programma che tuttavia crea un oggetto socket su queste porte deve avere i privilegi di root. Il programma client può deciderlo esplicitamente oppure lasciarlo decidere al proprio sistema operativo.

UDP:

Ogni pacchetto scambiato tra gli host è logicamente indipendente dai precedenti e successivi. Le perdite di pacchetti non vengono compensate in automatico né dalla rete né dal sistema operativo. Per questo motivo l'UDP è più leggero del TCP come uso di risorse di calcolo e di rete, ma il tipo di applicazione deve essere compatibile con questo tipo di comportamento (esempio, richiesta DNS). Se il server non ha nome occorre chiedere ad un utente sul server di scoprire e comunicarci il suo indirizzo IP, mentre lato server, siccome una risposta parte sempre dopo la relativa richiesta, l'indirizzo IP del client, così come la porta (notare che nel file infatti non viene specificato 20.000 nel server) è contenuto nelle informazioni di mittente della richiesta.

TCP:

E' byte oriented ossia non c'è correlazione a livello di programmazione tra il taglio di lettura e scrittura dei byte e l'effettivo pacchetto trasmesso. In altre parole, il sender e la sua specifica di quanti byte mandare il pacchetto, è scorrelato dal come il receiver decide di bufferizzarli e riceverli. E' utilizzata quando tra i pacchetti trasmessi c'è una relazione: sono parte di un "messaggio più grande" (ad es. un'immagine o un PDF). L'oggetto socket si preoccupa di numerare i pacchetti appartenenti alla stessa connessione per rilevare eventuali pacchetti persi e poterli ritrasmettere. Come vantaggio, l'utente scrive/legge su un archivio remoto con la stessa naturalezza di quando scrive/legge su un archivio locale come se la rete in mezzo non ci fosse. Come svantaggio invece, gli host trasmettitore e ricevitore devono "lavorare" di più dentro il sistema operativo, e i pacchetti persi e ritrasmessi arrivano in ritardo e questo può essere compatibile oppure no con il tipo di applicazione.

ESERCITAZIONE

Dipartimento di INFORMATICA

Esercizi basati su clientUDP e serverUDP

1) Lanciare prima il server e poi il client. Cosa si osserva? Invertire la sequenza di lancio. Cosa si osserva?
2) Modificare i sorgenti per mettere il server che **riceve** sulla porta 10000 e il client che **trasmette** dalla propria porta 30000 (ogni modifica dei sorgenti richiede una loro ricompilazione)
3) Mettere il server in ascolto sulla porta 100 e osservare cosa succede
   - Bisogna modificare anche il client? Dove?
   - Per chi usa il proprio PC con Linux o una virtual machine Linux, lanciare il server con il comando "`sudo ./serverUDP`" e osservare cosa cambia
4) Sostituire "127.0.0.1" prima con "localhost" e poi con "pippo" e osservare cosa succede
5) [Da fare solo se in Lab Delta] Accordarsi per lavorare su coppie di macchine in modo che server e client siano su macchine diverse. Come bisogna modificare i sorgenti?
6) Lanciare due volte il server usando due terminali. Cosa si osserva? Funzionano entrambi?
7) Modificare il server in maniera che soddisfi 5 richieste prima di terminare
   - E se volessi che non terminasse mai?

RISPOSTA:

N.B: durante il processo, è possibile visualizzare per bene lo scambio di pacchetti desiderato attraverso WireShark e il filtro "*ip.src === 127.0.0.1 && ip.dst === 127.0.0.1*";

1. se il server è aperto la richiesta viene soddisfatta; nel caso opposto, la richiesta client rimane in attesa e non verrà mai soddisfatta, nemmeno facendo partire il server effettivamente;
2. non avviene connessione dato che la porta è diversa;
3. bisogna modificare il client per far combaciare le porte;
4. localhost è la stessa cosa di quell'IP, pippo non funziona perchè non è una connessione;
5. bisogna mettere stessa porta e l'ip della connessione
6. il secondo server termina subito probabilmente perchè vede che è già aperto;
7. fatto, basta aggiungere un while(true) prima della ricezione UDP;

Esercizi basati su clientUDP_inc.c e serverUDP_inc.c:

1) Compilare ed eseguire il secondo esempio;
2) Modificarlo per costruire una semplice sommatrice – Il client acquisisce ripetutamente da tastiera un numero intero e lo manda al server finché l'utente digita zero – Il server accumula in una variabile "somma" i valori mandati dal client finché il client manda zero – Quando il client manda zero il server risponde al client con la somma ottenuta;

RISPOSTA:
In sostanza i file originali hanno un client che chiede un numero all'utente, lo manda al server, che come sempre va eseguito prima, che lo incrementa di uno e lo rimanda indietro.

Per modificare il file è stato sufficiente aggiungere un while response != 0, creare una somma cumulativa nel server, o continuare a chiedere input all'utente nel client.

3) impossibile da fare senza ethernet;
4) Invocare il server della sommatrice con due client diversi (tutti e tre possono anche essere sulla stessa macchina ovviamente su finestre terminali diverse). Che somma leggo da ciascun client? E' la somma che ciascun client da solo si aspetterebbe?

RISPOSTA:
Il primo client che termina riceve la somma cumulativa ottenuta da entrambi i client, mentre il secondo rimane in attesa infinita.

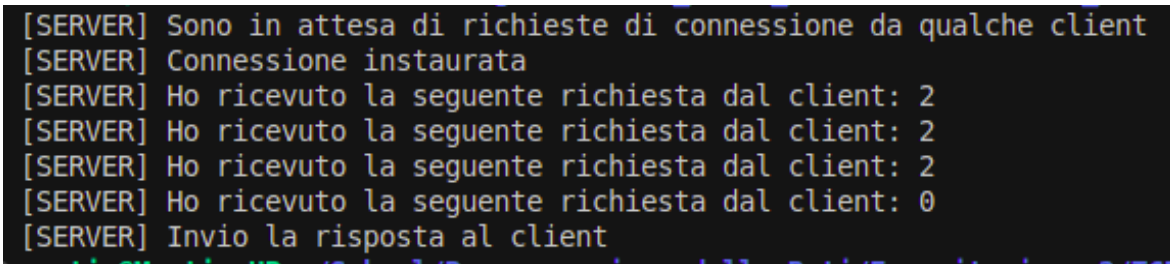
Esercizi basati su clientTCP.c e serverTCP.c

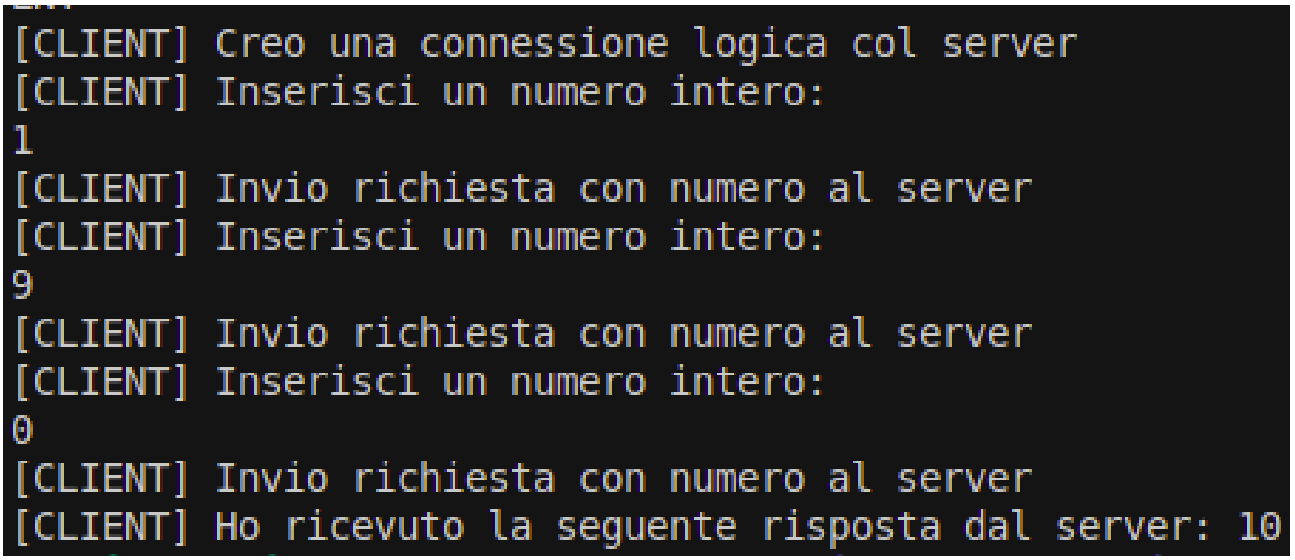
1) Lanciare due volte il server usando due terminali. Cosa si osserva? Funzionano entrambi?
2) Scrivere la sommatrice usando TCP, compilare ed eseguire
3) Provare a rifare il caso dell'Esercizio 10 ma con questa nuova versione della sommatoria. Cosa si può osservare? Che soluzione si può trovare? C'è influenza reciproca tra i due client?

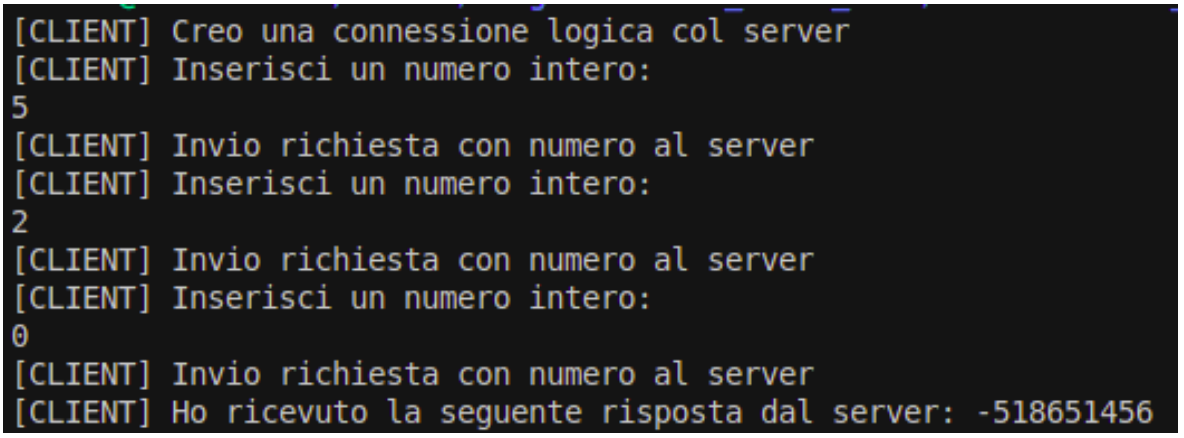
Risposta 1:
Il secondo server lanciato si chiude subito probabilmente perché la funzione di creazione del socket per prima cosa fa un controllo per la porta richiesta, e se già aperta killa il processo;

Risposta 2:
Fatto! chiaramente c'era da separare fuori dal ciclo while le richieste di apertura, la sua accettazione, le chiusure TCP e l'invio della risposta dal server al client.

Risposta 3:
Il primo client che si collega riceve effettivamente la somma cumulativa corretta, relativa ai soli dati inviati dallo stesso. Il secondo client invece (verificabile facilmente dagli output del server), invia i dati ma il server non ne salva la somma (ossia proprio non lo "caga"); l'unica cosa che funziona effettivamente è l'apertura della connessione, quindi lo scheletro iniziale del processo (già la chiusura non funziona correttamente). Come si vede dalle img, la risposta al secondo client in realtà è un errore di calcolo interno al client stesso probabilmente.







Esercizi basati su clientTCPChar.c e serverTCPChar.c

Per questi file non ci sono esercizi da svolgere. In ogni caso, il client invia un char ‘A’ e il server risponde con un char ‘B’. Estremamente statico.

Esercizi basati su clientTCPIO.c e serverTCPIO.c

1. Il client richiede al server un file specificandone il nome. Il server lo trasmette un byte alla volta. Il client lo salva in locale con lo stesso nome. Quale protocollo usiamo?

Quando devi trasferire un file come in questo caso, specialmente se è piccolo, hai bisogno che i dati arrivino in ordine, completi, senza duplicazioni, e che l'intera comunicazione sia affidabile. Ho quindi creato un server e un client che sfruttano le funzioni TCPSender e TCPReceiver della libreria network.h/c, nonchè le sys call open/read/write per la gestione del file, per far si che: client richieda nome file all’utente, lo mandi al server che attende il nome, carica il file e lo legge, mandandolo indietro al client un byte alla volta che a sua volta lo stampa su terminale.