Laboratorio di Reti e Sistemi Distribuiti

1. Introduzione e Packet Analysis

Roberto Marino, PhD[1]
roberto.marino@unime.it

[1]Dipartimento di Matematica, Informatica, Fisica e Scienze della Terra
Future Computing Research Laboratory
Università di Messina

Last Update: 26th February 2025

Indice

Perchè questo corso?

- Perché imparare ad osservare ed utilizzare l'architettura di rete dei sistemi UNIX-like è un requisito fondamentale per chi vuole andare oltre la "teoria" delle `Reti a Commutazione di Pacchetto` e vuole inserirsi nel mondo della tecnologia e della ricerca scientifica.
- Perchè il modello distribuito sottende un gran numero di tecnologie oggi allo stato dell'arte: dall'apprendimento distribuito, alle reti di sensori, alla robotica.

Struttura del Corso

- Parte 1: Programmazione di Rete
- Parte 2: Sistemi Shared Memory e Progammazione Concorrente
- Parte 3: Sistemi Message Passing ed Algoritmi Distribuiti
- Course fil-rouge: Unix network toolkit

Chi sono

Roberto Marino, PhD

Education:

- B.Sc. Ingegneria Informatica e Telecomunicazioni, UniME
- M.Sc. Ingegneria Robotica - UniGE
- M.Sc. Automatica, Robotica, Segnali e Immagini - Ecole Centrale de Nantes
- Ph.D. Ingegneria Robotica - UniGE

`Interessi di ricerca: Robotics, AI e Scientific Machine Learning, Data-Driven Control, Distributed Systems`

Email: roberto.marino@unime.it

Telegram group: LabReSiD24-25@unime

Personal and Teaching Page: https://roberto-marino.github.io

Riferimenti


Achille Pattavina
Internet e Reti. Fondamenti
Pearson, 3rd edition


Kurose, Ross
Reti di Calcolatori ed Internet. Un approccio Top-down
Pearson, 8th edition


Andrew S. Tanenbaum
Sistemi Distribuiti. Principi e paradigmi
Pearson


Simone Piccardi
Guida alla programmazione in Linux
on-line

Software consigliato

E' caldamente consigliato installare ed utilizzare

Ubuntu 22.04 LTS (Jammy) o successive

Wireshark

GCC e Python3

net-tools

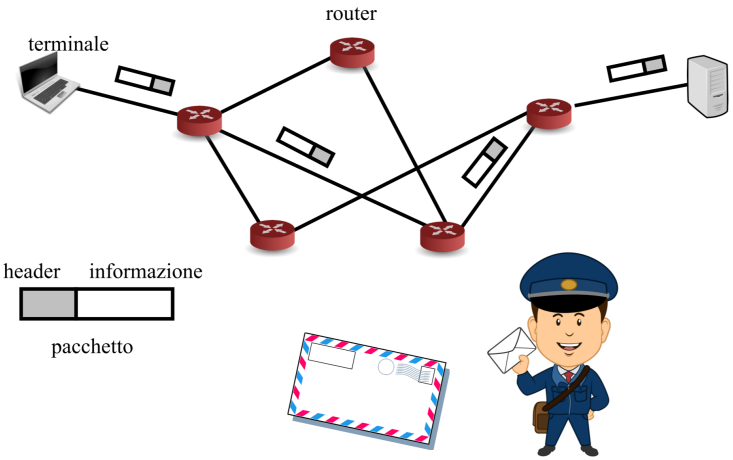
Ray.io

USATE IL TERMINALE! (non solo visual studio code

IMPARATE AD UTILIZZARE LATEX! (www.overleaf.com free account)

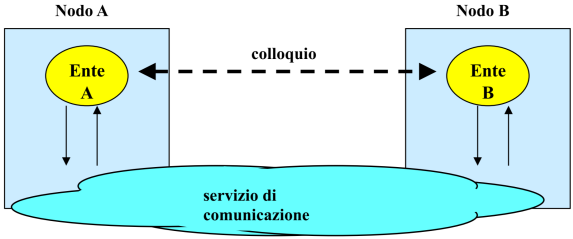
QUESTO E' UN LABORATORIO, NON UN CORSO DI RETI O DI SISTEMI OPERATIVI. La frequenza non è obbligatoria ma caldamente consigliata. E' necessario, per accedere alla prova finale (progetto + orale) consegnare i report di tutte le esperienze di laboratorio

Internet: definizioni e concetti



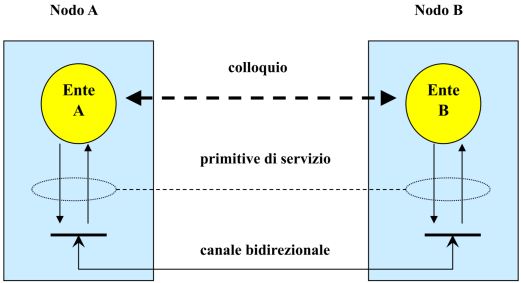
Il Servizio di Comunicazione: Entità, Protocollo, Primitive

La differenza tra protocollo e primitiva di servizio è **fondamentale** nel contesto delle reti a commutazione di pacchetto e dei sistemi distribuiti.



Primitive di servizio

Una primitiva di servizio è un'operazione elementare fornita da un livello di un sistema di comunicazione (ad esempio, un livello del modello OSI) per permettere ai livelli superiori di utilizzare i servizi offerti. Le primitive di servizio sono le "azioni" che un livello può eseguire, come richiedere una connessione, inviare dati o chiudere una connessione.



Primitive di servizio

Tipi di primitive di servizio:

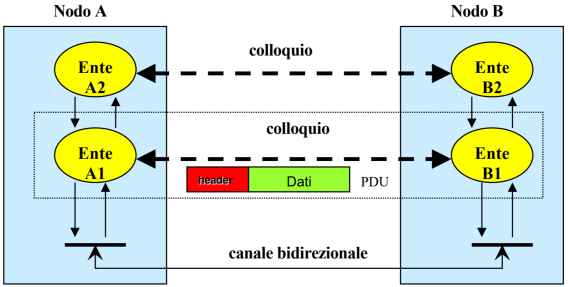
- Request (Richiesta): Un'entità richiede un servizio al livello sottostante.
- Indication (Indicazione): Il livello sottostante informa un'entità di un evento (ad esempio, una richiesta in arrivo).
- Response (Risposta): Un'entità risponde a una precedente indicazione.
- Confirm (Conferma): Il livello sottostante conferma il completamento di una richiesta.

Esempi di primitive di servizio: Socket API: Le primitive come connect(), send(), recv(), e close() sono esempi di primitive di servizio per la comunicazione di rete.

PROTOCOLLI

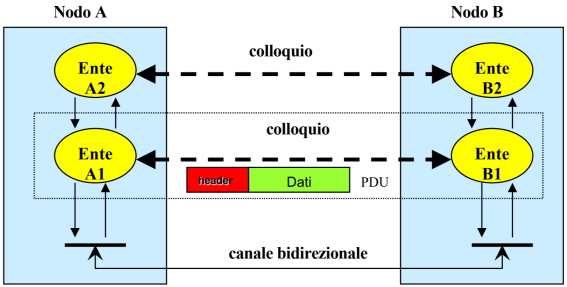
Un **protocollo** è un insieme di regole e convenzioni che definiscono come avviene la comunicazione tra due o più entità (ad esempio, dispositivi, sistemi o applicazioni). I protocolli specificano:

- Il formato dei messaggi (ad esempio, intestazioni, campi, checksum).
- La sequenza delle operazioni (ad esempio, handshake, conferme, ritrasmissioni).
- Le azioni da intraprendere in caso di errori o situazioni particolari.



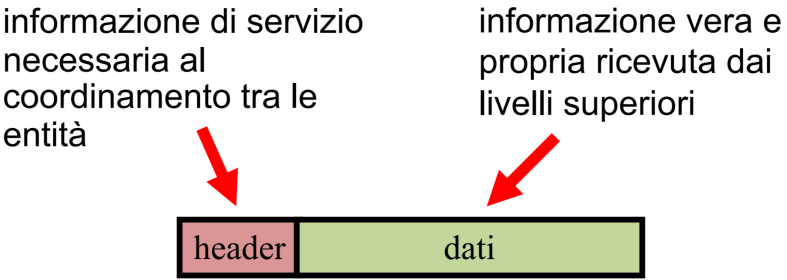
STACKING

Le entità che colloquiano in un servizio di telecomunicazione possono anche offrire un servizio di comunicazione a entità terze, dette di livello superiore. Le entità di un livello collaborano per fornire il servizio di comunicazione al livello superiore e si scambiano messaggi mediante il servizio offerto dal livello inferiore.



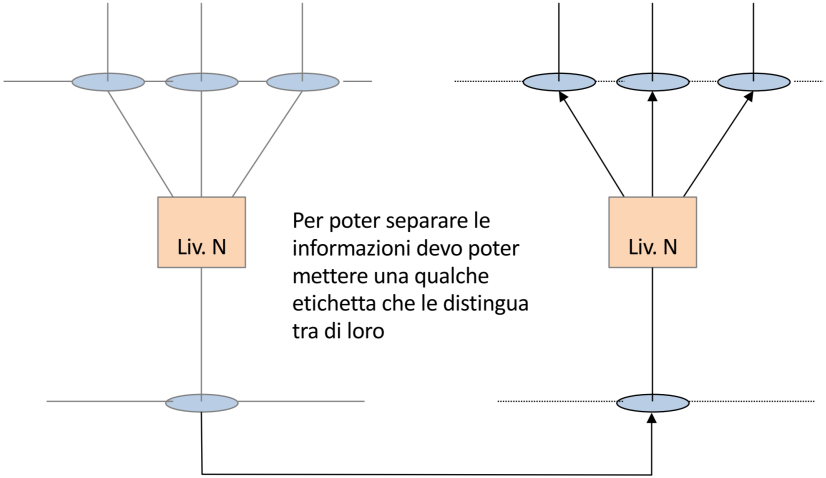
PACKET DATA UNIT

Un protocollo utilizza per il colloquio tra entità dello stesso livello delle unità di trasferimento dati dette PDU, che possono contere:



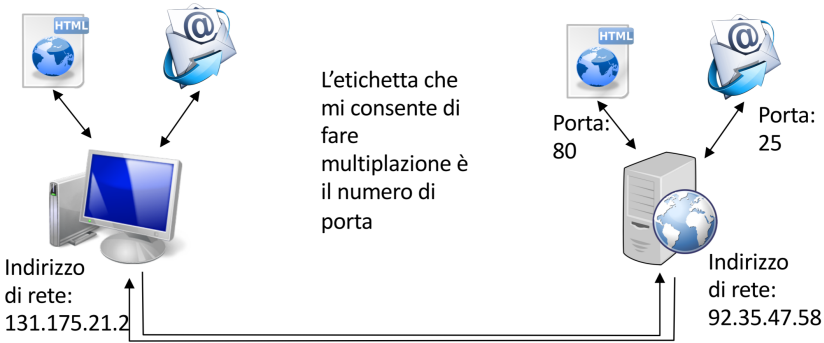
MULTIPLAZIONE

Più livelli superiori possono condividere lo stesso servizio di comunicazione

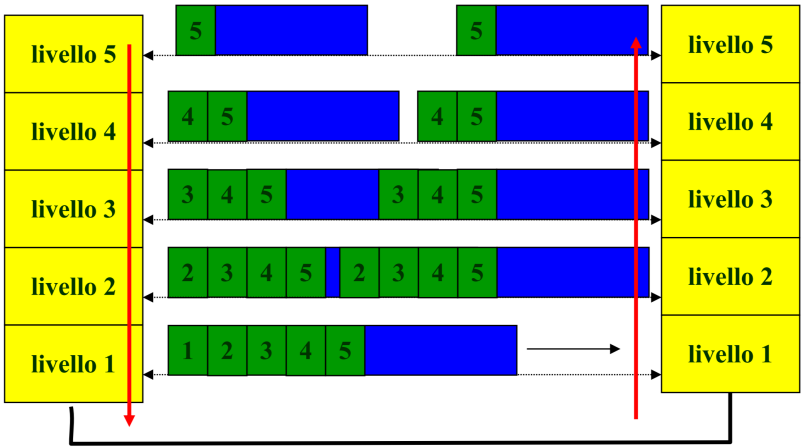


MULTIPLAZIONE

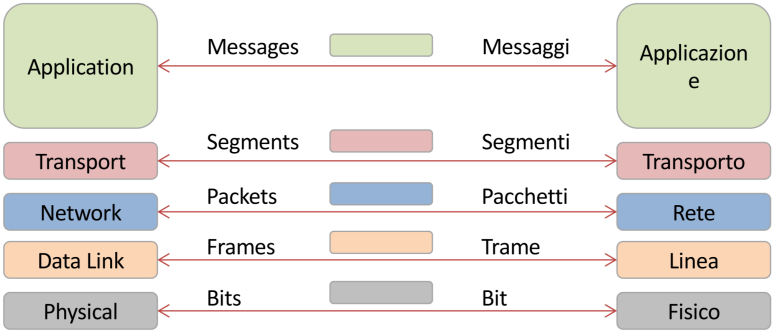
Più applicazioni condividono la stessa interfaccia di rete e sono distinte in base ad un numero di porta



INCAPSULAMENTO ED ESTRAZIONE



Nomenclatura



Request For Comment (https://datatracker.ietf.org/)

Un **RFC (Request for Comments)** è un documento tecnico che descrive standard, protocolli, metodi o concetti relativi a Internet e alle reti di computer. Gli RFC sono pubblicati dalla Internet Engineering Task Force (IETF), un'organizzazione internazionale che si occupa dello sviluppo e della promozione di standard per Internet.

Storia

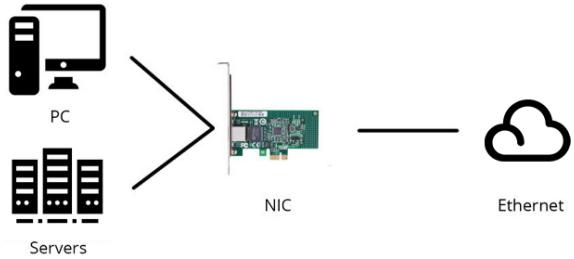
Gli RFC sono nati nel 1969, durante lo sviluppo di ARPANET, il precursore di Internet. Il nome "Request for Comments" riflette l'approccio collaborativo e aperto della comunità tecnica: i documenti venivano inizialmente distribuiti per raccogliere feedback e suggerimenti prima di diventare standard ufficiali.

Scopi degli RFC:

- Definire standard: Descrivere protocolli di rete e formati di dati.
- Fornire linee guida: Offrire best practice per l'implementazione di tecnologie.
- Documentare idee: Presentare nuove proposte o concetti per la discussione.
- Condividere informazioni: Fornire dettagli tecnici su aspetti specifici di Internet.

Network Interface Card (NIC)

Una interfaccia di rete è un punto di connessione tra un dispositivo (come un computer, un server, un router o un dispositivo IoT) e una rete. Funge da "porta" attraverso la quale il dispositivo può inviare e ricevere dati sulla rete. Ogni interfaccia di rete è associata a un indirizzo IP univoco (nella maggior parte dei casi) e a un indirizzo MAC (Media Access Control), che ne consentono l'identificazione e la comunicazione all'interno della rete.



Network Interface Card (NIC)

Interfacce fisiche

- **Ethernet (eth0, enpXsY):** Utilizzate per connessioni cablate (ad esempio, cavo Ethernet).
- **Wi-Fi (wlan0, wlpXsY):** Utilizzate per connessioni wireless.
- **Interfacce PPP/PPPoE:** Utilizzate per connessioni dial-up o DSL.

Interfacce Virtuali

- **Loopback (lo):** Un'interfaccia virtuale utilizzata per la comunicazione interna del dispositivo (ad esempio, l'indirizzo 127.0.0.1).
- **Veth (Virtual Ethernet):** Coppie di interfacce virtuali utilizzate per collegare namespace di rete (ad esempio, nei container).
- **Tunnel (tun0, tap0):** Interfacce virtuali utilizzate per creare tunnel VPN o altre connessioni crittografate.
- **Bridge (br0):** Un'interfaccia virtuale che funge da switch di rete, collegando più interfacce fisiche o virtuali.

Apparati di rete: HUB

Un hub è un dispositivo di base che opera al livello fisico **(Layer 1)** del modello OSI. Quando un hub riceve un pacchetto di dati da un dispositivo collegato, lo invia a tutti gli altri dispositivi connessi, indipendentemente dal destinatario effettivo. Questo processo è noto come broadcasting.

Non tiene traccia degli indirizzi MAC dei dispositivi collegati ed ha dominio di collisione unico!

Apparati di rete: SWITCH

Uno switch opera al livello di collegamento dati **(Layer 2)** del modello OSI e può anche funzionare a livelli superiori (Layer 3 o superiore) negli switch più avanzati. Quando uno switch riceve un pacchetto di dati, lo invia solo al dispositivo di destinazione, utilizzando l'indirizzo MAC per identificare il destinatario corretto. Mantiene una tabella degli indirizzi MAC (MAC address table) per gestire in modo efficiente il traffico di rete.

Domini di collisione multipli!

Apparati di rete: ROUTER

Un router è un dispositivo di rete che opera al livello di rete **(Layer 3)** del modello OSI e viene utilizzato per collegare reti diverse, come una rete domestica o aziendale a Internet. A differenza di hub e switch, che gestiscono il traffico all'interno di una singola rete locale (LAN), il router è progettato per instradare i dati tra reti diverse, come ad esempio tra una LAN e una WAN (Wide Area Network, come Internet).

Apparati di rete: Confronto

Caratteristica	Hub	Switch	Router
Livello OSI	Layer 1 (Fisico)	Layer 2 (Collegamento dati)	Layer 3 (Rete)
Funzione principale	Invia dati a tutti i dispositivi	Invia dati solo al destinatario	Instrada dati tra reti diverse
Gestione del traffico	Nessuna gestione intelligente	Gestisce il traffico in base ai MAC	Gestisce il traffico in base agli IP
Sicurezza	Bassa	Media	Alta (quando firewall integrato)
Utilizzo	Obsoleto	Reti locali (LAN)	Collegamento tra reti (LAN/WAN)

Table: Confronto tra Hub, Switch e Router

Bridge

Un bridge (in italiano "ponte") è un dispositivo di rete che opera al livello di collegamento dati **(Layer 2)** del modello OSI. Il suo scopo principale è quello di collegare due segmenti di rete separati, facendoli apparire come un'unica rete. A differenza di un router, che opera a livello di rete (Layer 3) e instrada i pacchetti tra reti diverse, un bridge lavora a livello di collegamento dati e utilizza gli indirizzi MAC per decidere se inoltrare o meno i frame tra i segmenti di rete.

Nota bene

Un bridge ha un numero limitato di porte (solitamente 2-4) e opera in modo più semplice. Uno switch è essenzialmente un bridge multiporta, con molte più porte e funzionalità avanzate come VLAN e gestione del traffico più efficiente. I bridge oggi sono ampiamente sostituiti dagli switch

IP ADDR

Il comando `ip addr` (o ip address) è utilizzato nei sistemi Linux per visualizzare e configurare gli indirizzi IP delle interfacce di rete. L'output di questo comando fornisce informazioni dettagliate sulle interfacce di rete presenti nel sistema, inclusi gli indirizzi IP, gli indirizzi MAC, lo stato delle interfacce e altre informazioni di configurazione.

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever

2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen
1000
    link/ether 08:00:27:53:8b:dc brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.100/24 brd 192.168.1.255 scope global dynamic eth0
       valid_lft 86384sec preferred_lft 86384sec
    inet6 fe80::a00:27ff:fe53:8bdc/64 scope link
       valid_lft forever preferred_lft forever
```

Loopback output parameters

- `1: lo:`: Indica che questa è l'interfaccia di loopback, utilizzata per la comunicazione interna all'interno del sistema.
- `<LOOPBACK,UP,LOWER_UP>`: L'interfaccia è di tipo loopback, è attiva (`UP`) e il collegamento fisico è attivo (`LOWER_UP`).
- `mtu 65536`: La Maximum Transmission Unit (MTU) è impostata a 65536 byte, massimo valore di trasferimento senza frammentazione.
- `qdisc noqueue state UNKNOWN`: La coda di disciplina (`qdisc`) è `noqueue` e lo stato dell'interfaccia è `UNKNOWN`.
- `link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00`: L'indirizzo MAC dell'interfaccia di loopback è sempre `00:00:00:00:00:00`.
- `inet 127.0.0.1/8 scope host lo`: L'indirizzo IPv4 dell'interfaccia di loopback è `127.0.0.1` con una maschera di rete `/8`. Lo scope è `host`, valido solo per il sistema locale.
- `inet6 ::1/128 scope host`: L'indirizzo IPv6 dell'interfaccia di loopback è `::1` con una maschera di rete `/128`. Anche questo è valido solo per il sistema locale.

Eth0 output parameters

- `2: eth0:`: Indica che questa è l'interfaccia di rete Ethernet, utilizzata per la comunicazione con la rete esterna.
- `<BROADCAST,MULTICAST,UP,LOWER_UP>`: L'interfaccia supporta la trasmissione broadcast e multicast, è attiva (`UP`) e il collegamento fisico è attivo (`LOWER_UP`).
- `mtu 1500`: La MTU è impostata a 1500 byte, valore standard per le interfacce Ethernet.
- `qdisc pfifo_fast state UP`: La coda di disciplina (`qdisc`) è `pfifo_fast` e lo stato dell'interfaccia è `UP`.
- `link/ether 08:00:27:53:8b:dc brd ff:ff:ff:ff:ff:ff`: L'indirizzo MAC dell'interfaccia Ethernet è `08:00:27:53:8b:dc` e l'indirizzo broadcast è `ff:ff:ff:ff:ff:ff`.

Eth0 output parameters

- `inet 192.168.1.100/24 brd 192.168.1.255 scope global dynamic eth0`: L'indirizzo IPv4 dell'interfaccia è `192.168.1.100` con una maschera di rete `/24`. L'indirizzo broadcast è `192.168.1.255`. Lo scope è `global`, valido per la rete esterna. L'indirizzo è ottenuto dinamicamente (ad esempio, tramite DHCP).
- `valid_lft 86384sec preferred_lft 86384sec`: Indica il tempo di vita valido e preferito per l'indirizzo IP (in secondi).

Scope e lifetime e coda di disciplina

- **LFT** (preferred e valid): tempo di validità dell'indirizzo prima che venga rimosso o deprecato (se è statico impostato a `forever`)
- **scope**: ambito di validità dell'indirizzo (`host` per la macchina, `link` per la sottorete, `global` per internet intero
- **Coda di disciplina**: un meccanismo del kernel Linux per la gestione del traffico di rete, definendo come i pacchetti vengono accodati e trasmessi attraverso un'interfaccia di rete (pfifo_fast (old), fc_codel (new))

ANALISI DEL TRAFFICO

L'analisi del traffico di rete si compone di due momenti fondamentali:

1. la cattura del traffico;

2. l'analisi vera e propria dei dati raccolti. In genere è difficile attuare l'analisi in tempo reale, mentre il traffico viene prodotto sulla rete, perche i volumi sono eccessivi per un operatore umano (ma in alcuni casi perfino per uno strumento di analisi automatico). Il problema esiste perfino per la semplice raccolta, perche la presa in carico dei pacchetti da parte del kernel e la successiva registrazione introducono dei ritardi ineliminabili: i tool di raccolta a volte segnalano che un certo numero di pacchetti sono stati scartati (dropped by kernel): quando l'analisi e particolarmente critica (p.es. in ambito forense), bisognera tenerne conto.

NIC in Promiscuous Mode

Comando ip

sudo ip link set eth0 promisc on

Off

sudo ip link set eth0 promisc off

Via tcpdump, solo per questa sessione

sudo tcpdump -i eth0

Verifica

ip link show eth0 |grep PROMISC

Metodologia

La comprensione personale dei protocolli di rete può essere approfondita vedendoli in azione, osservando la sequenza di messaggi scambiati tra due entità di protocollo, approfondendo i dettagli delle operazioni e forzando i protocolli a effettuare determinate azioni per verificarne le conseguenze.

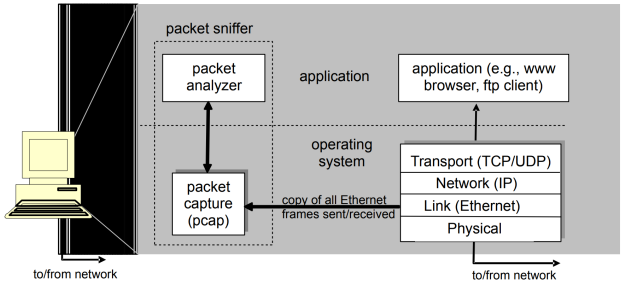
Packet Sniffing

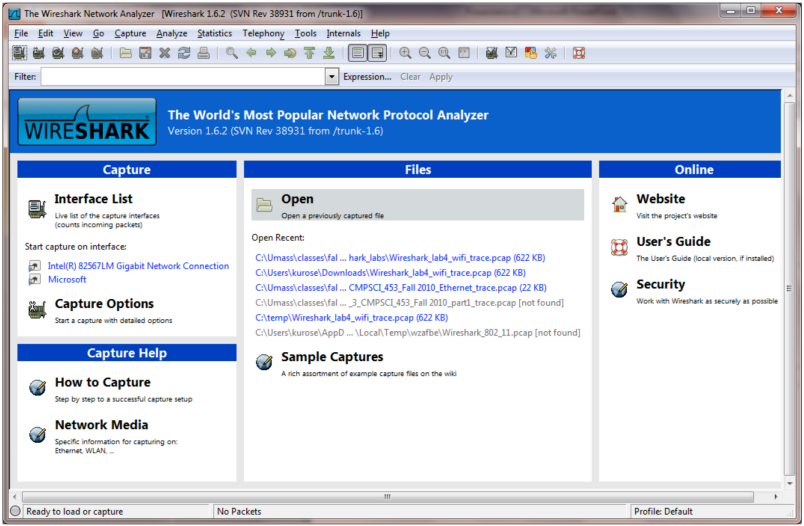
Lo strumento di base per osservare i messaggi scambiati tra entità di protocollo in esecuzione è chiamato packet sniffer. Come suggerisce il nome, esso copia passivamente (ossia "sniffa, annusa") i messaggi che vengono inviati e ricevuti dal vostro computer; inoltre, mostra i contenuti dei vari campi di protocollo e dei messaggi catturati. Un packet sniffer è una entità passiva: osserva i messaggi inviati e ricevuti dalle applicazioni e dai protocolli in esecuzione sul vostro computer, ma non manda mai egli stesso dei pacchetti. Allo stesso modo, i pacchetti che riceve non sono mai stati inviati esplicitamente al packet sniffer. Al contrario, il packet sniffer riceve una copia dei pacchetti che sono spediti/ricevuti dalle applicazioni e dai protocolli in esecuzione.

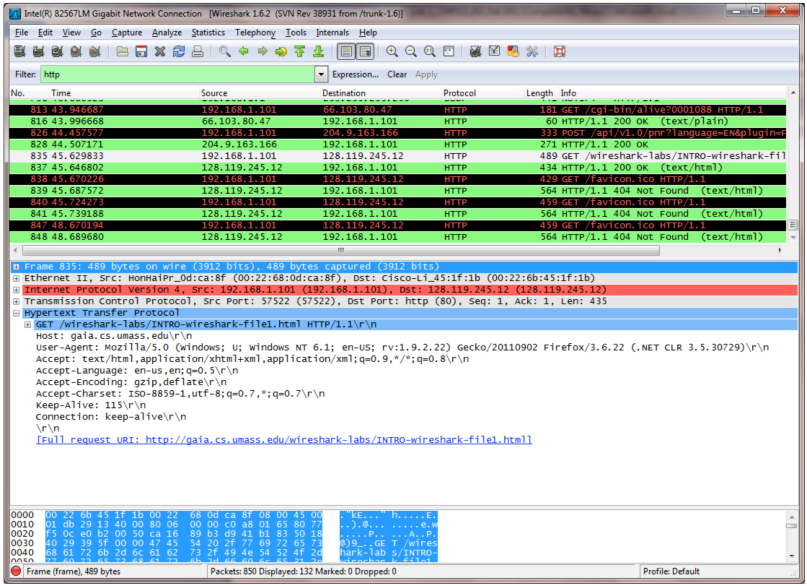
Wireshark Architecture

Component

- Capture Packet Library (libpcap @ UNIX)
- Packet Analyzer




The Wireshark Network Analyzer [Wireshark 1.6.2 (SVN Rev 38931 from /trunk-1.6)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: Expression... Clear Apply
WIRESHARK
The World's Most Popular Network Protocol Analyzer
Version 1.6.2 (SVN Rev 38931 from /trunk-1.6)
Capture
Interface List
Live list of the capture interfaces (counts incoming packets)
Start capture on interface:
Intel(R) 82567LM Gigabit Network Connection
Microsoft
Capture Options
Start a capture with detailed options
Capture Help
How to Capture
Step by step to a successful capture setup
Network Media
Specific information for capturing on: Ethernet, WLAN, ...
Files
Open
Open a previously captured file
Open Recent:
C:\Umass\classes\fal ... hark_labs\Wireshark_lab4_wifi_trace.pcap (622 KB)
C:\Users\kurose\Downloads\Wireshark_lab4_wifi_trace.pcap (622 KB)
C:\Umass\classes\fal ... CMPSCI_453_Fall 2010_Ethernet_trace.pcap (22 KB)
C:\Umass\classes\fal ... _3_CMPSCI_453_Fall 2010_part1_trace.pcap [not found]
C:\temp\Wireshark_lab4_wifi_trace.pcap (622 KB)
C:\Users\kurose\AppD ... \Local\Temp\wzafbe\Wireshark_802_11.pcap [not found]
Sample Captures
A rich assortment of example capture files on the wiki
Online
Website
Visit the project's website
User's Guide
The User's Guide (local version, if installed)
Security
Work with Wireshark as securely as possible
Ready to load or capture
No Packets
Profile: Default


Intel(R) 82567LM Gigabit Network Connection [Wireshark 1.6.2 (SVN Rev 38931 from /trunk-1.6)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: http
Expression... Clear Apply
No. Time Source Destination Protocol Length Info
813 43.946687 192.168.1.101 66.103.80.47 HTTP 181 GET /cgi-bin/alive?0001088 HTTP/1.1
816 43.996668 66.103.80.47 192.168.1.101 HTTP 60 HTTP/1.1 200 OK (text/plain)
826 44.457577 192.168.1.101 204.9.163.166 HTTP 333 POST /api/v1.0/pnr?language=en&plugin=f
828 44.507171 204.9.163.166 192.168.1.101 HTTP 271 HTTP/1.1 200 OK
835 45.629833 192.168.1.101 128.119.245.12 HTTP 489 GET /wireshark-labs/INTRO-wireshark-fil
837 45.646802 128.119.245.12 192.168.1.101 HTTP 434 HTTP/1.1 200 OK (text/html)
838 45.670226 192.168.1.101 128.119.245.12 HTTP 429 GET /favicon.ico HTTP/1.1
839 45.687572 128.119.245.12 192.168.1.101 HTTP 564 HTTP/1.1 404 Not Found (text/html)
840 45.721273 192.168.1.101 128.119.245.12 HTTP 459 GET /favicon.ico HTTP/1.1
841 45.739188 128.119.245.12 192.168.1.101 HTTP 564 HTTP/1.1 404 Not Found (text/html)
847 48.670194 192.168.1.101 128.119.245.12 HTTP 459 GET /favicon.ico HTTP/1.1
848 48.689680 128.119.245.12 192.168.1.101 HTTP 564 HTTP/1.1 404 Not Found (text/html)
Frame 835: 489 bytes on wire (3912 bits), 489 bytes captured (3912 bits)
Ethernet II, Src: HonHaiPr_0d:ca:8f (00:22:68:0d:ca:8f), Dst: Cisco-Li_45:1f:1b (00:22:6b:45:1f:1b)
Internet Protocol Version 4, Src: 192.168.1.101 (192.168.1.101), Dst: 128.119.245.12 (128.119.245.12)
Transmission Control Protocol, Src Port: 57522 (57522), Dst Port: http (80), Seq: 1, Ack: 1, Len: 435
Hypertext Transfer Protocol
GET /wireshark-labs/INTRO-wireshark-file1.html HTTP/1.1\r\n
Host: gaia.cs.umass.edu\r\n
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2.22) Gecko/20110902 Firefox/3.6.22 (.NET CLR 3.5.30729)\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
[Full request URI: http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html]
0000 00 22 6b 45 1f 1b 00 22 68 0d ca 8f 08 00 45 00
0010 01 db 29 13 40 00 80 06 00 0d c0 a8 01 65 80 77
0020 f5 0c e0 b2 00 50 ca 16 89 b3 d9 41 b1 83 50 18
0030 40 29 39 5f 00 00 47 45 54 20 2f 77 69 72 65 73
0040 68 61 72 6b 2d 6c 61 62 73 2f 49 4e 54 52 4f 2d
Frame (frame), 489 bytes
Packets: 850 Displayed: 132 Marked: 0 Dropped: 0
Profile: Default

TCPDUMP

`tcpdump` raccoglie traffico di rete (non solo TCP). E' basato sulle librerie pcap il cui uso e molto diffuso anche in altre applicazioni per l'analisi e manipolazione del traffico e che sono lo standard de-facto per la cattura dei pacchetti. Le operazioni di raccolta necessitano dei privilegi di amministrazione, perche occorre utilizzare l'interfaccia di rete in modalita promiscua. tcpdump puo essere usato anche per analizzare traffico (conservato in un file pcap): in questo caso non servono particolari privilegi (oltre ai permessi di lettura sul file!). **La modalita promiscua dell'interfaccia di rete fa si che non vengano scartati i pacchetti con destinatari diversi** (si ricordi che sul mezzo trasmissivo condiviso di una LAN arrivano tutti i pacchetti, almeno all'interno dello stesso collision domain).

Collision domain e promiscuous mode

Una scheda di rete (NIC) accetta **normalmente** solo pacchetti con:

- Il proprio indirizzo MAC come destinazione.
- Pacchetti broadcast o multicast.

In **modalità promiscua**, la scheda di rete:

- Accetta tutti i pacchetti indipendentemente dall'indirizzo di destinazione.

Che traffico vede una NIC posta in modalità promiscua e collegata in rete tramite uno switch?

TCPDUMP: USE CASES

Sniffare tutto il traffico di rete

tcpdump -i eth0

Filtrare il traffico proveniente da un ip

tcpdump host x.x.x.x

Filtrare per sorgente o destinazione

tcpdump src x.x.x.x
tcpdump dst x.x.x.x

TCPDUMP: USE CASES

Filtrare il traffico da o per una rete

tcpdump net x.x.x.x/24

Filtrare il traffico relativo ad una porta

tcpdump port 3389
tcpdump src port 1025
tcpdump portrange 21-23

Filtrare per protocollo

tcpdump icmp

TCPDUMP: USE CASES

Ottenere il contenuto in esadecimale

tcpdump -c 1 -X icmp

Leggere da o scrivere su un file

tcpdump port 80 -w capturefile
tcpdump -r capturefile

HARDCORE: logica booleana

tcpdump -nnvvS src x.x.x.x and dst port 3389
tcpdump dst x.x.x.x and not icmp

Tcpdump: struttura di un output

Tcpdump: output di esempio

```
16:30:15.123456 IP 192.168.1.10.54321 > 192.168.1.1.80:  Flags [S],
seq 123456789, win 64240, length 0
```

- **Timestamp**: `16:30:15.123456`
- **Protocollo**: `IP`
- **Sorgente**: `192.168.1.10.54321`
- **Destinazione**: `192.168.1.1.80`
- **Flags TCP**: `[S]`
- **Numero di sequenza**: `123456789`

Flags TCP Principali

- **S (SYN)**: Richiesta di connessione
- **F (FIN)**: Fine connessione
- **R (RST)**: Reset della connessione
- **P (PSH)**: Invio immediato dati
- **A (ACK)**: Conferma di ricezione
- **U (URG)**: Dati urgenti

Filtrare solo pacchetti con flag SYN

```
tcpdump 'tcp[tcpflags]  (tcp-syn) != 0'
```

Esempi di analisi

Three way handshake

```
16:30:15.123456 IP 192.168.1.10.54321 > 192.168.1.1.80:  Flags [S],
seq 123456789, win 64240, length 0
16:30:15.123789 IP 192.168.1.1.80 > 192.168.1.10.54321:  Flags [S,
A], seq 987654321, ack 123456790, win 65535, length 0
16:30:15.124012 IP 192.168.1.10.54321 > 192.168.1.1.80:  Flags [A],
ack 987654322, win 64240, length 0
```

Connessione stabilita correttamente

```
Client -> Server:  SYN
Server -> Client:  SYN-ACK
Client -> Server:  ACK
```

Esempi di analisi

Trasmissione Dati HTTP con flag PUSH

```
16:30:15.200123 IP 192.168.1.10.54321 > 192.168.1.1.80:  Flags [P.],
seq 123456790:123457000, ack 987654322, win 64240, length 210
16:30:15.200567 IP 192.168.1.1.80 > 192.168.1.10.54321:  Flags [A],
ack 123457000, win 65535, length 0
```

Trasmissione Dati HTTP

```
Client -> Server:  PSH (invio dati immediato all'applicazione)
Server -> Client:  ACK (ricezione confermata)
```

Esempi di analisi

Reset della connessione

```
16:30:20.400123 IP 192.168.1.10.54321 > 192.168.1.1.80:  Flags [R],
seq 123457001, win 0, length 0
```

Reset della Connessione

```
Client -> Server:  RST (connessione chiusa)
```