Protocolli e Architetture di Rete

Appunti di teoria

Iacopo Ruzzier

Ultimo aggiornamento: 18 novembre 2024

Indice

1 Introduzione a reti e protocolli

1.1 Reti

Essendo difficile dare una definizione globale di rete, vediamone una **ricorsiva**: rete come

- insieme di **nodi** connessi tramite **collegamenti**
- insieme di **reti** connesse tramite **nodi**

La definizione introduce una prima astrazione: quando inseriamo una rete all'interno di una rete più grande, **non ci interessa come è fatta la singola rete** al suo interno

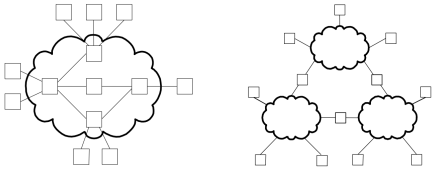

Fig. 1.1: Definizione ricorsiva di rete

1.1.1 Componenti fondamentali

- **nodi** → distinti per **ruolo**
  - host: termine astratto, identifica un **nodo terminale** (mittente o destinatario, dunque connesso ad un capo della comunicazione)
  - switch, bridge, router: nodi intermedi che abilitano la comunicazione tra host

  Vedremo che gli host devono implementare **tutti i protocolli di rete**, mentre i nodi intermedi possono conoscerne anche solo una parte (costo decisamente minore)

- **collegamenti (link)**: tutti i mezzi fisici usati per comunicare, cablati e non (wireless)

La definizione ricorsiva permette l'uso di **paradigmi di astrazione**: nascondo dettagli per focalizzarmi su funzionalità di interesse
Inoltre, l'approccio ricorsivo permette il **riutilizzo**

Esempio: passaggio di informazioni tra host in LAN diverse

LAN/WAN: Local/Wide Area Network; gli host si connettono alle LAN, le WAN collegano le LAN

Logicamente comunicano i due host terminali, ma in realtà le informazioni **attraversano tutti i nodi e collegamenti intermedi** → sorgono problemi di

- **instradamento** - che direzione far prendere al messaggio
- **condivisione delle risorse** - un nodo condiviso da vari host avrà prestazioni **variabili**

L'obiettivo, apparentemente semplice, di trasferire un messaggio in forma di sequenza di bit da un host a un altro, è in realtà difficile da raggiungere al meglio

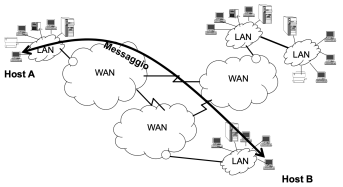

Fig. 1.2: comunicaz. logica

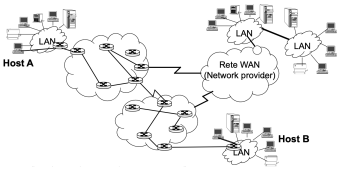

Fig. 1.3: comunicaz. reale

1.1.2 Aspetti fondamentali delle reti moderne

- host, nodi, link **eterogenei** → **sia HW che SW**
- la rete può **cambiare nel tempo** (es. aggiunta di host a comunicazione iniziata)
- **compromessi** per ottenere i massimi benefici:
  - **costi** vs **prestazioni** vs **affidabilità** vs **sicurezza**
  - condivisione della rete

L'estrema eterogeneità ci porta a preoccuparci di

- diversità HW/SW dei nodi
- com'è interconnesso l'host
- modalità di trasmissione del messaggio
- gestione del transito attraverso i n. intermedi
- servizi disponibili all'utente
- ...

Difficoltà (tutto può andare storto):

- interferenze elettriche (errori a livello di bit)
- congestioni (errori a livello di messaggi)
- guasti link/n. intermedi
- problemi SW di host

→ **ritardi, consegne fuori ordine, "ascolto da terzi"...**
⇒ per far comunicare tra loro delle entità di una rete, tenendo conto di tutte le problematiche e circostanze, si richiede **cooperazione**: **tutte le comunicazioni sono regolate da protocolli**

1.2 Protocolli

Protocollo: insieme di **regole e convenzioni** seguite da entità, dislocate **su nodi distinti**, che intendono **comunicare per svolgere un compito comune**

Obiettivo: assicurare una cooperazione **efficiente e affidabile** per la comunicazione e per la realizzazione di servizi di rete

Il singolo protocollo **non può risolvere tutti i problemi**! Usiamo pr. diversi a seconda del contesto e del nostro scopo

1.2.1 Elementi fondamentali (e rigorosi)

- **sintassi**: insieme delle **informazioni ammissibili** → insieme e struttura di comandi e risposte, formato dei messaggi, ... (correttezza a livello di "parola" - "presenza della parola nel dizionario")
- **semantica**: **significato** di comandi, azioni, risposte in seguito a ricezione e trasmissione dei messaggi ("sequenze ammissibili di aprole per formare frasi di senso computo")
- **temporizzazione**: specifica delle possibili **sequenze temporali di emissione di comandi, messaggi, risposte** ("seq. ammissibili di frasi per un discorso di senso compiuto")

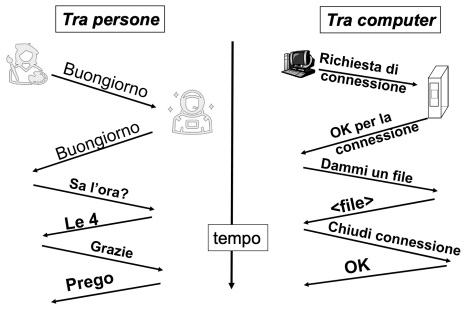

Fig. 1.4: Esempi di comunicazione

A differenza del linguaggio comune, dove qualche parola sbagliata non interferisce con la comprensione del messaggio, nelle reti informatiche può essere sufficiente un errore di pochi bit

4 domande fontamentali per la comunicazione tra entità

1. **architettura HW** → quale HW esegue il protocollo
   → dove viaggia l'informazione
2. **schema di naming/identificazione** → come si chiamano gli interlocutori
3. **architettura SW** → quale SW esegue il protocollo
4. **schema di comunicazione** → quale paradigma di comunicazione voglio usare

1.2.2 Stack di protocolli

Il vero obiettivo è il **trasferimento di un messaggio, garantendo**

- **prestazioni** (massima velocità possibile)
- **affidabilità** (superamento di guasti o malfunzionamenti)
- **sicurezza**

In un contesto eterogeneo rendono il problema non banale: si usa un approccio *divide et impera*, che in ambito informatico si concretizza usando astrazioni per mascherare la complessità
→ approccio stratificato (layering), realizzando uno **stack (o suite) di protocolli** in cui ogni livello cerca di risolvere un problema diverso

Quasi mai i nodi sono i singoli PC, bensì le varie **applicazioni in esecuzione (processi)**

Fig. 1.5: Esempio concettuale semplificato

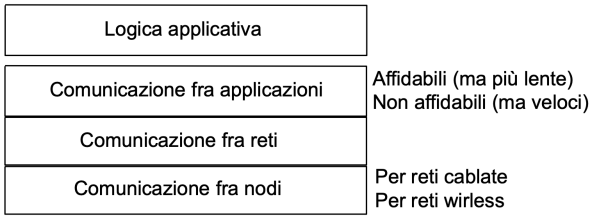

Fig. 1.6: Termini più generici

Il livello 1 è quello **HW** (ci interessa meno), e più saliamo più ci avviciniamo alla logica applicativa. I livelli superiori possono essere variabili, e ciascun livello può offrire protocolli "alternativi" a seconda dell'aspetto da sostenere (es. affidabilità vs velocità - una telefonata è preferibile in tempo reale, anche se perdo qualche parola ogni tanto)
In termini logici, salendo di livello le comunicazioni avvengono **sulla stessa rete locale** (1^o) → **su reti diverse** (2^o) → **tra nodi** (3^o)

Lo stack può "aumentare verso l'alto" indefinitamente, mentre i primi 3 livelli ci sono sempre

Ogni livello ha

- 2 interfacce **interne**, rivolte verso i livelli superiore e inferiore
- 1 interfaccia **esterna**, per comunicazioni tra mittente e destinatario *sullo stesso livello*

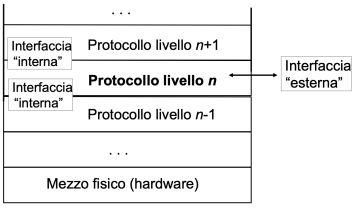

Fig. 1.7: Interfacce a livello n

Concettualmente e logicamente, la comunicazione avviene tra **protocolli allo stesso livello**. In realtà avviene in modo diretto solo al livello **più basso** (attraverso il mezzo fisico), mentre per gli altri è indiretta: il messaggio *scende* dal livello applicativo a quello fisico (mittente), per poi *risalire* lo stack (destinatario).

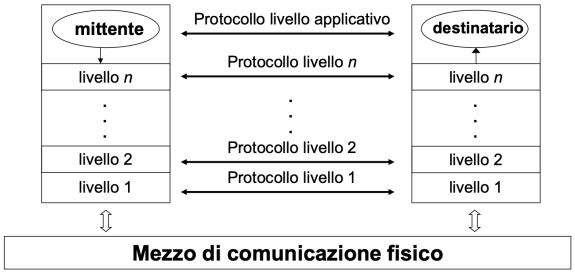

Fig. 1.8: Comunicazione logica

Fig. 1.9: Comunicazione fisica

Possiamo dire che ogni protocollo del mittente parla con lo stesso protocolo del destinatario.

1.2.3 Pacchetti

In termini di protocollo, l'unità minima di comunicazione è il **pacchetto**, struttura dati composta da

- **PCI** (Protocol Control Information) o **header**: insieme di tutti i metadati necessari per il funzionamento del protocollo
- **SDU** (Service Data Unit) o **payload**: il verto contenuto informativo scambiato

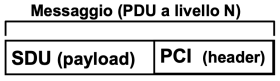

Fig. 1.10: Struttura del pacchetto a livello n

In generale, PCI+SDU=**PDU** (Protocol Data Unit)

Implementazione dell'interfaccia esterna

- quando un livello invia un messaggio, invia **l'intero PDU** a sua disposizione al livello più basso
- il livello inferiore **aggiunge il proprio header** al PDU ricevuto (rimane inalterato)
- a liv. 1 avremo molto payload + header liv. 1
- quando il messaggio arriva al destinatario, ogni livello **toglie l'header** del suo protocollo e **inoltra il suo payload** al liv. superiore

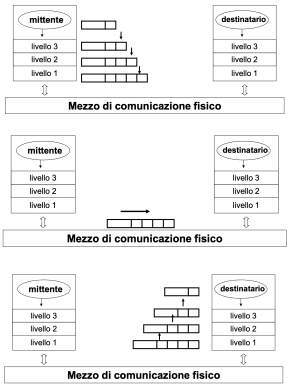

Fig. 1.11: Schema di funzionamento dell'incapsulamento e della gerarchia

È implicito che gli stack dei nodi host **devono essere gli stessi affinché avvenga la comunicazione**

Ci sono casi in cui non avviene solo questo, ma sono eccezioni oppure casi di errore (es. un livello riceve un PDU e non riconosce il PCI)

Sintesi sui protocolli

1. il sistema di comunicazione basato su internet richiede un **insieme di protocolli cooperanti** (stack)
2. si identificano **relazioni gerarchiche** nelle fz. che compongono i processi di comunicazione (layering)
3. **indipendenza funzionale** tra livelli - interfacce (e definizioni) indipendenti dall'implementazione
4. il liv. n (sfruttando il servizio di $n-1$) fornisce un servizio a $n+1$
5. la comunicazione avviene **logicamente tra pari**, ma in realtà attraversa tutti i livelli (incapsulamento)

Questo sistema permette la comunicazione tra processi su macchine diverse **come se fossero sulla stessa**

1.3 Comunicazioni e standard

Gli standard sono necessari per contesti eterogenei - da sempre in informatica si arriva a uno standard *de iure* e a uno *de facto*:

- **ISO/OSI** (*de iure*): l'organizzazione ISO (International Std. Org.) ha definito le specifiche dello std di protocolli per l'interconnessione di nodi eterogenei (Open System Interconnection)
- **TCP/IP** (*de facto*): nato nelle università come metodo di comunicazione estremamente efficiente, ma **non definito come standard**

1.3.1 Stack ISO/OSI - Panoramica di funzionamento

Lo stack ha 7 livelli:

1. l. **fisico:** gestisce i particolari meccanici ed elettrici della *trasmissione fisica* di un flusso di bit
2. l. **di collegamento** (data link): gestisce i collegamenti tra *host locali* (in LAN) (valida i dati ricevuti)
3. l. **di rete:** gestisce i collegamenti *tra reti* → nonostante stiamo comunicando tra tante piccole reti locali, vogliamo una comunicaz. *come in una grande rete locale*
4. l. **di trasporto:** gestisce le com. dei *processi tra ciascun host* - è il protocollo con cui si interfacciano le applicazioni

   I livelli superiori presentano funzionalità di aiuto alle applicazioni

5. l. **di sessione:** consente a utenti su macchine eterogenee di stabilire sessioni, mantenendo lo stato (es. A vuole comunicare con B - come tratto i msg? sono tutti per la stessa app/tutti eterogenei? → B deve implementare uno stato) (sessione è il termine tecnico per definire questo comportamento)
6. l. **di presentazione:** manipola le informazioni per presentarle nel modo migliore possibile su ciascun dispositivo (es. conversioni di codifica)
7. l. **di applicazione:** livello concettuale, in cui semplicemente inseriamo l'applicazione (fornisce un'interfaccia std per i programmi applicativi)

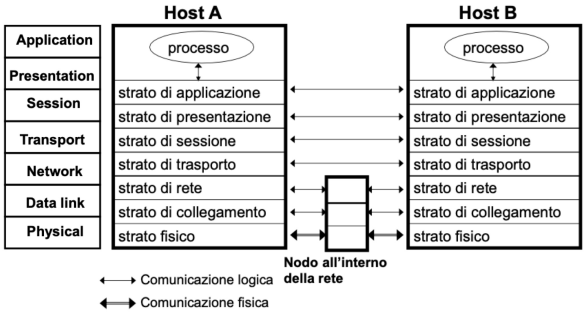

Fig. 1.12: Schema dello stack ISO/OSI

Reminder

Per ciascuna comunicazione, sono aggiunti/tolti i PCI di ogni livello → 7 livelli sono **troppi** - anche per questo si afferma TCP/IP (ne ha 5)

Il motivo principale per cui ha prevalso TCP/IP è che è uno stack **open, senza controllo centralizzato, e in generale più semplice e meno costoso**. Inoltre, aveva già un'API **funzionante**

Nonostante studieremo lo stack TCP/IP, per convenzione usiamo i numeri dei liv. ISO/OSI es. per dire che lavoriamo a liv. rete usiamo **3 e non 2**

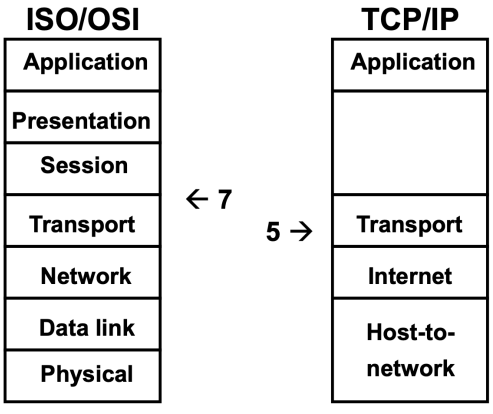

Fig. 1.13: Confronto tra ISO/OSI e TCP/IP

1.3.2 Stack TCP/IP

Definizione dei livelli

4. **trasporto**: supporta i trasferimenti fra processi in esecuzione su host
3. **rete**: trasferisce i pacchetti dal nodo mittente al destinatario
2. **link**: effettua i trasferimenti dei dati tra componenti della rete confinanti
1. **fisico**: trasferisce bit "sul cavo"

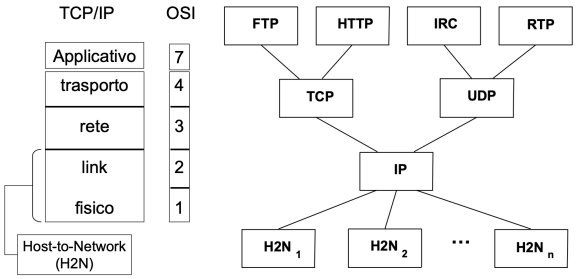

Fig. 1.14: Schema dello stack e implementazioni

I pr. del livello H2N comunicano con il pr. del livello network, cioè IP

A liv. trasporto:

- TCP - per creare comunicazioni **affidabili**
- UDP - non dà affidabilità

A liv. applicativo:

- HTTP - rete
- RTP - comunicazioni real-time

C'è molto altro, che vedremo in seguito o proprio non vedremo

Ricordiamoci che l'obiettivo è sempre far comunicare 2 host, e che i nodi intermedi analizzano il pacchetto a profondità differenti

Quando dobbiamo definire i vari nodi, li definiamo in base a ciò che implementano:

- **host**: implementa tutto lo stack
- **switch e bridge**: nodi di l. 2 (implementano fino a 2)
- **router**: liv. 3
- in generale, disp. di livello n implementano lo stack fino al liv. n, ma possono esserci eccezioni (es. switch di liv. 3)

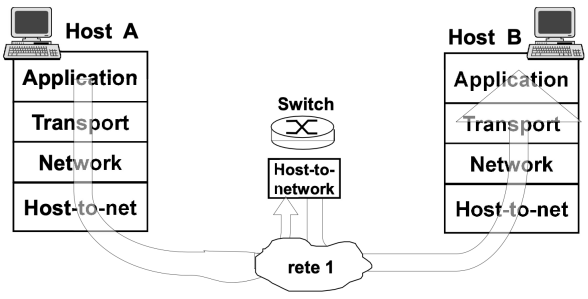

Fig. 1.15: Comunicazione a liv. 2 (LAN)

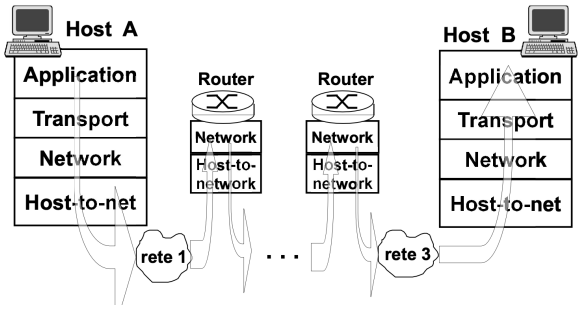

Fig. 1.16: Comunicazione a liv. 3 (Internet)

Il motivo per cui si usano disp. di livello più basso è **il costo a parità di velocità di gestione del traffico**

1.3.3 Problemi di instradamento e condivisione delle risorse

- i problemi di instradamento si risolvono tramite lo **stack**
- per la condivisione delle risorse usiamo il **multiplexing**

  → ogni volta che più dispositivi vogliono **comunicare tramite un canale condiviso**

  – la realizzazione dipende dal **paradigma** di com. e dalle **caratteristiche del canale** di com.

1.3.4 Modalità di trasferimento dati

Circuit switching

Alla base di protocolli analogici: 1 circuito virtuale ∀ comunicazione

- poco adatto a Internet: prevede l'**assegnazione statica del percorso**

→ poco efficiente in contesti dinamici

- Time o Frequency Division Multiplexing sono due esempi, entrambe dividono la risorsa a disposizione ma non permettono modifiche al percorso

Packet switching

Alla base di internet: dati suddivisi in **pacchetti** ed inviati in rete

- un pacchetto usa **tutta la capacità trasmissiva** di un link
- l'accesso al mezzo **non è temporizzato** (≠TDM) e **non è a prenotazione** → le risorse sono usate in base alla necessità

→ accessi **non decisi a priori**: paradigma la cui efficacia è stata dim. dalla teoria delle reti di code (Kleinrock, 1961)

- segue un principio di *MUX-ing statistico a suddivisione di tempo*: pacchetti da sorgenti diverse sono "mescolati" sullo stesso link senza slot temporali dedicati e occupando tutta la banda disponibile

Non essendoci garanzia di disponibilità della risorsa, possiamo incontrare 2 problemi principali:

- **collisione**: host ≠ vogliono comunicare nello **stesso esatto momento**

  → fenomeno a livello locale

- **congestione**: molti pacchetti arrivano **contemporaneamente ad un nodo intermedio**

  → malfunz. a livello non locale (buffer dei router)

  – es. dopo una com. corretta a livello locale, i pacchetti arriveranno ad un router che li *smisterà* in base alla destinazione; per farlo li *analizza* e li mette su dei *buffer di uscita* → congestione **quando i buffer sono pieni** ed è **obbligato a scartare un pacchetto**

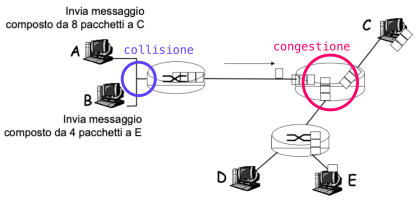

Fig. 1.17: Dove avvengono i malfunzionamenti locali (*collisione*) e non (*congestione*)

1.3.5 Metrica di prestazione

- principalmente usiamo la **larghezza di banda** (o bandwidth o banda di trasmissione), tipicamente in multipli di bit/secondo (es. Kbps o Kbit/s, Mbps, Gbps, ...)

→ identifica la velocità **a liv. fisico** (a liv. applicativo risulterà inferiore)

- la massima velocità ottenibile è detta **v. nominale** → non la raggiungo es. quando uso un canale di comunicazione condivisa

es. link a 1 Mbps (nominale), ciascun utente richiede 0.1 Mbps quando trasmette ed è attivo il 10% del tempo

– circuit sw.: **max 10 utenti**

– packet sw.: con **35 utenti**, la prob. che *10+ trasmettano contemporaneamente* è **0.04%** → **rischio minimo**

2 Livello H2N (Host-2-Network)

Unione dei livelli 1 e 2 dello stack ISO/OSI
Affronta le problematiche di

- **interconnessione** tra 2+ host
- **trasmissione dati** tra host connessi direttamente
- **conessione a Internet** di un host

Nello stack TCP/IP (non in ISO/OSI) la **modalità** di interconnessione, la relativa **tecnologia** ed i **protocolli** per la trasmissione di dati tra host interconnessi sono **strettamente dipendenti**

Modalità	Tecnologie/Protocolli
LAN wired	Ethernet, token ring, ...
LAN wireless	802.11x
PAN	Bluetooth, ...
mediante modem	SLIP, ...
WAN wireless	GSM, LTE, ...

Tabella 1: Esempi concreti di tecnologie e protocolli usati nelle varie modalità

Premessa

I servizi offerti da protocolli H2N $\neq$ possono essere **diversi**! Ad es., possono **garantire** o **no** l'**affidabilità** della consegna di un pacchetto (pur facendo parte dello stesso livello dello stack)
$\rightarrow$ il liv. network deve essere in grado di garantire l'arrivo dei pacchetti **anche in presenza di pr. H2N** $\neq$

Osservazione

Solitamente i collegamenti **wired** sono **più affidabili** rispetto a quelli **wireless**

- c. **wireless**: più soggetti a malfunzionamenti $\implies$ protocolli **più sicuri e complessi**
- c. **wired**: più sicuri fisicamente $\implies$ protocolli **meno sicuri e più semplici**

Studieremo il pr. Ethernet, che essendo pensato per reti cablate (sicure fisicam.) è generalmente **inaffidabile**

Alcune definizioni:

Modalità di trasmissione

Dipendono da chi sono i nodi terminali che vogliono comunicare

- **unicast**: 1 a 1 (mittente-destinatario)
- **multicast**: 1 a molti (vogliamo comunicare con **tutti i destinatari**)
- **anycast**: 1 a "almeno 1" (sufficiente 1 solo ricevente)
- **broadcast**: 1 a tutti gli altri nodi (multicast dove molti = tutti)