**Global Network**

La rete è un insieme di nodi interconnessi tra loro.

Ogni nodo corrisponde ad un dispositivo ed essi sono gestiti da router.

La rete è poi suddivisa in sottoreti anche esse gestite e collegate tra loro da router attraverso una sottorete chiamata backbone. L’insieme di sottoreti collegate tra di loro che condividono un certo tipo di comunicazione sono dette sistemi autonomi.

Un sistema autonomo puo essere a sua volta fornitore o cliente. Nel primo caso fornisce servizi, nel secondo li riceve.

All'interno di un sistema autonomo i singoli router comunicano tra loro, per scambiarsi informazioni relative alla creazione delle [tabelle di routing](https://it.wikipedia.org/wiki/Tabella_di_routing) o [tabelle di instradamento](https://it.wikipedia.org/w/index.php?title=Tabelle_di_instradamento&action=edit&redlink=1), attraverso un [protocollo](https://it.wikipedia.org/wiki/Protocolli_di_routing) [IGP](https://it.wikipedia.org/wiki/Interior_gateway_protocol) (interior gateway protocol).

L'interscambio di informazioni tra router appartenenti a sistemi autonomi differenti avviene attraverso un protocollo [BGP](https://it.wikipedia.org/wiki/Border_Gateway_Protocol) (Border Gateway Protocol) e punti di interscambio fisici tra i diversi sistemi ([NAP](https://it.wikipedia.org/wiki/Neutral_Access_Point)). La comunicazione avviene attraverso pacchetti che sono dei frammenti di informazione auto-contenuta. Al interno del pacchetto troviamo tutte le informazioni necessarie per indentificarlo e per conoscere la destinazione e il mittente. Tutte le informazioni al suo interno sono regolate da dei protocolli, che spesso cambiano da livello a livello. La comunicazione in internet è quindi poi suddivisa in strati, questi strati detti link:

**Fisico Collegamento Dati Rete Trasporto Sessione Presentazione Applicazione**

Negli anni 70 i dispositivi di rete erano costruiti da aziende diverse che realizzavano hardware e software conl’obiettivo di far comunicare esclusivamente i prodotti dell’azienda produttrice senza curarsi della comunicazione con sistemi diversi: si realizzarono quelli che furono in seguito identificati come sistemi chiusi (closed system). Con il passare del tempo nacque la necessità di collegare tra loro dispositivi anche a media distanza, per esempio tra due sedi delle stesse azienda poste in città differenti, e quindi i sistemi chiusi dovettero connettersi a sistemi e impianti di comunicazione, sia privati che pubblici, di altri produttori.

L'ISO (International Organization for Standardization) iniziò a dare le basi per la coordinazione dello sviluppo di standard per l’interconnessione dei sistemi di elaborazione, mettendo a punto lo standard OSI (Open System Interconnetcion) così facendo creò un protocollo di comunicazione che fosse in grado di far comunicare sistemi con caratteristiche diverse tra loro, passando dai sistemi chiusi ai sistemi aperti (Open System). Per esempio se vogliamo far comunicare due computer che usano due codifiche diverse per rappresentare i caratteri, il primo usa la codifica ASCII e il secondo la codifica UNICODE con dei sistemi chiusi e senza un protocollo di interconnessione non sarebbero in grado di "capirsi".

**Phisical Layer**

Il livello 1 è il livello più esterno, e quindi il più basso, della modello ISO/OSI. Il livello fisico rappresenta i bit in cui è codificata l'informazione, in un segnale adatto al mezzo trasmissivo. Il suo obiettivo è quello di gestire il mezzo trasmissivo che può essere un cavo coassiale, una fibra ottica, dei connettori o dei doppini telefonici tramite il quale avviene la trasmissione delle informazioni. Inoltre, il livello fisico ha la funzione di mettere in comunicazione l'elaboratore al canale di comunicazione si occupa di tutte le problematiche che permettono la corretta trasmissione del bit dal mittente al destinatario.

**Data Link Layer**

Il livello 2 si occupa di gestire il collegamento da un pc all'altro **appartenenti alla stessa LAN** (Local Area Network) o sottorete, è un sistema di comunicazione che permette ad apparecchiature indipendenti di comunicare tra loro, entro un’area delimitata, utilizzando un canale fisico a velocita' elevata e con basso tasso d’errore.

Altra funzione principale di questo livello è quella di **recuperare gli errori trasmissivi**: di solito questo avviene mediante tecniche di ritrasmissione automatica dei dati che sono stati ricevuti corrotti.

**Identificare i nodi connessi**  
**Le interfacce di rete NIC** (Network Interface Controller, anche Network Interface Card) dispongono di un numero cablato (di 48 bit) **detto MAC address**, non modificabile, univoco in tutto il mondo, viene impiegato per identificare le entita'.

I primi tre ottetti indicano il **produttore** Serve a rendere indipendente l’indirizzo IP dall’hardware: se cambio il server l’IP resta lo stesso.  
  
**Controllare gli errori**  
Aggiunge al pacchetto proveniente dal livello ISO/OSI superiore (il livello network) una sequenza di bit (**checksum**) che e' usata in ricezione per valutare la corretta trasmissione del pacchetto.

Se il "checksum calcolato" è diverso dal "checksum ricevuto" il destinatario individua un errore.

**Data Framing**  
Il secondo livello forma dei pacchetti dati, detti **frame** da far viaggiare lungo la dorsale di comunicazione.

Il livello Data Link **incapsula** il pacchetto proveniente dall livello IP in un nuovo pacchetto detto frame al quale aggiunge un **header** (intestazione) e un **tail** (coda).

*Preamble (7 ottetti): sequenza di 7 ottetti (10101010) SFD (1 ottetto): start of frame delimiter (1 ottetto 10101011) DA+SA (6+6 ottetti): destination e source address Length/Type (2 ottetti): lunghezza o tipo (lunghezza max 1518 ottetti) I di Ethernet esistono diverse versioni, che si differenziano in base alla funzione attribuita a questo campo Pad: per avere pacchetti sempre superiori alla lunghezza minima per rendere possibile la Collision Detection FCS: checksum di controllo*

Il frame ha una struttura più **semplice** di quella del pacchetto IP, ed è chiara la sua relazione con un dispositivo con poca memoria.

Il frame si apre con una lunga sequenza che alterna livelli 1 e livelli 0.

Poi vengono indicati i due indirizzi **MAC** relativi al **mittente** e al **destinatario**.

Viene anche speciﬁcato il tipo di **payload** (nel nostro caso IP) Dopo il payload (lungo al massimo 1500 ottetti) c’è un **trailer** con la **checksum** dell’intero frame, ed un periodo di silenzio di lunghezza determinata.

Per giustiﬁcare il formato del frame consideriamo i due sistemi utilizzati per gestire un frame in uno **switch**.

**I Store and forward**: il pacchetto viene registrato per intero nella memoria dello switch, e poi viene inoltrato verso la destinazione.

**I Cut-through**: il pacchetto inizia ad essere inoltrato appena ne è nota la destinazione.

Il sistema cut-through è più vantaggioso dal punto di vista del tempo di trattamento del frame, e della memoria necessaria nello switch.

Per questo è opportuno un formato che presenti prima gli indirizzi Per la stessa ragione è opportuno che la checksum sia registrata nel trailer.

Anche il protocollo **ARP** che permette di conoscere l'indirizzo fisico di una scheda di rete corrispondente ad un indirizzo **IP**, appartiene a questo livello.

Il protocollo ARP, che non è speciﬁco per Ethernet, funziona così:

1 Spedizione dell’indirizzo da risolvere (IP) all’indirizzo MAC di broadcast.

2 Risposta da parte del dispositivo con l’IP corrispondente.

3 Registrazione in una cache dove restano per un breve periodo, ad evitare di essere continuamente richiesti.

**Correggere gli errori mediante ritrasmissione**  
I pacchetti inviati sono tutti numerati.

Per ogni pacchetto ricevuto, il **destinatario** invia al mittente un segnale di **ACK** (acknowledgement, conferma) contenente lo stato della trasmissione (positivo o negativo) e il numero del pacchetto associato.

Il **mittente** deve ripetere l'invio dei pacchetti in corrispondenza dei quali non ha ricevuto il segnale di Acknowledge in un intervallo di tempo prestabilito.

L'ACK puo' essere trasmesso con un messaggio a sé stante, o in un campo all'interno di un pacchetto utente che viaggia in direzione opposta (questa modalità prende il nome di piggyback). Puo' capitare che, per via dei rinvii dovuti a ritardi nella ricezione dell'ACK, il destinatario riceva lo stesso pacchetto più volte: in questo caso le copie del pacchetto verranno scartate.   
**Controllare il flusso**  
Ovvero **sincronizzare** il dispositivo fisico piu' **veloce** portandolo alla velocita' di quello piu' **lento**.

In questo modo si evita che un mittente troppo veloce tenda a trasmettere pacchetti in quantita' superiore a quella supportata dal ricevente, evitando che quest'ultimo risulti completamente sopraffatto dal sovraccarico di lavoro (buffer over flow).   
**Multiplazione**  
Nelle reti la connessione fisica tra due elaboratori (linee bifiliari, collegamenti in fibra ottica, collegamenti radio ecc...) deve essere utilizzata in modo tale che piu' applicazioni possono contemporaneamente trasmettere informazioni sullo stesso canale : si pone quindi il problema di sfruttare la capacità del mezzo a disposizione per ottenere piu' canali di livello fisico (o logico) sullo stesso mezzo trasmissivo.

**Accesso condiviso al canale MAC (Medium Access Control)**  
Disciplina l'accesso **multiplo** di piu' nodi ad un canale di comunicazione condiviso (**multiplazione**) evitando o gestendo l'occorrenza di **collisioni**.

**Ethernet**

Ethernet è il tipo più diffuso di **rete locale** che esista al mondo.

Ethernet usa un solo cavo per collegare decine di stazioni di lavoro, ciascuna delle quali riceve contemporaneamente tutto quel che passa sulla rete, mentre solo una stazione alla volta ha la facoltà di trasmettere. Ogni stazione è indipendente e non esiste una singola entità che funzioni da arbitro.

Una rete Ethernet viene ad essere condivisa tra molti utenti, ciascuno con il suo indirizzo MAC e IP.

Esistono molti modi per organizzare la condivisione della risorsa, quello di Ethernet si basa(va) su una modalità detta **CSMA-CD** dove il nodo che deve spedire un pacchetto lo invia, ma controlla che la linea non sia allo stesso momento disturbata da un’altra trasmissione. Se questo accade il pacchetto viene rispedito dopo un intervallo casuale, che si allunga progressivamente (exponential back-off).

**Ethernet gestione collisioni**

Si parla di **collision domain** per un insieme di utenti incompatibili.

Incompatibili signiﬁca che non ci può essere contemporaneamente comunicazione tra due coppie di utenti nel dominio di collisione I Ad es.: se {A, B, C, D} è un dominio di collisione, se A sta comunicando con B allora C non può comunicare con D (e viceversa).

Dispositivo **hub** (antiquato), collision domain unico, solo una trasmissione alla volta.

Dispositivo **switch**, full-duplex utilizzando hardware più veloce delle interfacce (da Gigabit Ethernet).

Uno **switch** dispone di veri buchi in cui si innestano dei veri cavi I buchi si chiamano, con poca fantasia, **porte**.

Uno switch deve conoscere l’indirizzo MAC del nodo collegato all’altra estremità del cavo collegato ad una delle sue porte.

La corrispondenza non viene impostata dall’amministratore, ma con un semplice protocollo.

Si parla di apprendimento della corrispondenza MAC-porta analizzando i frame in ingresso.

Il protocollo non prevede messaggi speciﬁci, ma si basa sull’osservazione del trafﬁco.

Se su una porta ricevo un frame con un certo MAC mittente, quello è il MAC del dispositivo agganciato a quella porta. Se si riceve un frame per il quale non si conosce la porta corrispondente al destinatario, viene spedito su tutte le porte.

Così uno switch non deve essere conﬁgurato manualmente: **plug and play.**

**Loop**  
Una rete può essere collegata a più di uno switch (affidabilità) La topologia non è un albero, e possono esserci cicli.

**I frame possono restare intrappolati, e percorrere indeﬁnitamente il ciclo** (non c’è TTL).

Se non è nota la porta corrispondente ad un certo host (silenzioso), un frame per quell’host gira ﬁntantoché non è lui a spedirne uno. È necessario interrompere i cicli, per fare in modo che il meccanismo di learning funzioni in modo ottimale.

Per questo si introduce un algoritmo speciﬁco per calcolare uno spanning tree (o albero di copertura) L’algoritmo si basa sullo **scambio periodico di informazioni tra gli switch**, le Bridge Protocol Data Units.

La **BPDU** è un dato complesso che contiene, tra l’altro:

l’identiﬁcatore del mittente (ogni switch ha un proprio identiﬁcatore), la radice dell’albero, un’indicazione della distanza del mittente dalla radice.

La distanza si ottiene aggiungendo il costo del link alla distanza dell’origine del link dalla radice.

Ogni switch invia una **BPDU** ogni 2 secondi ad un indirizzo MAC speciale su tutte le porte.

**Spanning tree**

Ogni switch esegue lo stesso algoritmo:

1. Si identifica la radice dell’albero: che è lo switch con l’identificatore minimo, All’accensione uno switch ritiene di essere la radice se trova una **BPDU** con l’indicazione di una radice con id inferiore, la acquisisce come radice.
2. Identificazione della porta root di ogni switch: ogni switch deve capire chi è il proprio padre, che è la porta su cui è collegato lo switch, e viene chiamata **ROOT** porta radice, viene rilevata con le distanze ricevute dai vicini.
3. Identificazione porte designate o abilitate: quindi il vicino imposta quella porta come porta designata che viene utilizzata per la diffusione di informazione, e quelle che non sono segnalate designate vengono disattivate.

Ci sono diverse varianti dell’algoritmo, soprattutto per migliorarne la risposta in momenti critici, quindi guasti o modifiche di switch.

**Wifi**

*Il* ***WIFI*** *è un protocollo link noto come 802.11 e si differenzia da un protocollo cablato perché funziona senza cavi ma con trasmissione elettro magnetica.*

*Riproduce un’astrazione simile ad una rete locale costruita con ethernet.*

*Esistono due diverse modalità di funzionamento per una sottorete, che in questo caso si chiama (Basic Service Set) BSS.*

1. *Definita AD HOC in cui tutti i nodi partecipano allo stesso livello, il suo limite e che non avendo un componente controllo deve auto-organizzarsi.*
2. *La seconda è infrastructure e prevede una componente controllo che è l* ***access point****, e costruisce l’infrastruttura di comunicazione, e dal punto di vista astratto è simile allo switch e serve per offrire connettiva e coordinazione a tutti gli altri noti appartenenti allo stesso BSS. Tutti i nodi comunicano solo con l access point.*

*Quando un nodo wireless si accende cerca un AP per connettersi ed avere accesso al internet locale, cosi il nuovo nodo entra a far parte dell’infrastruttura.*

*Gli AP hanno un identificatore che è* ***SSID*** *che può essere trasmesso periodicamente con dei frame che si chiamano* ***BEACON****.*

*Il collegamento ad un AP si chiama* ***associazione*** *e richiede l’autentificazione del nodo che si vuole associare.*

*Quando si conclude l AP assegna al nodo un indirizzo IP. È paragonabile ad una sottorete ethernet.*

*Le collisioni in ethernet avvenivano quando due nodi contemporaneamente cercavano di accedere alla rete che non consentiva la trasmissione contemporanea, e per risolverlo chi spediva controllava se c erano altre collisioni CSMA-CD.*

*In una stazione Wi-Fi che è un trasmettitore e ricevitore di onde elettromagnetiche, nel momento della trasmissione il ricevitore è impedito da fare qualsiasi ricezione da altri nodi, quindi il ricevitore è assordato dalla trasmissione e quindi non si rilevano* ***collisioni****.*

*La transizione tra trasmissione e ricezione, non è istantanea, nel momento in cui smetto di trasmettere e avvio la ricezione passa un lasso di tempo che è* ***SIFS****.*

**Algoritmo Collision Avoidance**

Prima di avviare la trasmissione la stazione controlla la presenza di un’altra trasmissione durante un intervallo **DIFS** (maggiore di SIFS) I se la linea resta libera avvia la trasmissione, se la linea viene occupata entra subito in un exponential back-off, per evitare che tutti quelli che sono in attesa entrino in collisione. Al termine il ricevente attende **SIFS**, e spedisce un **ACK**.

Se chi ha spedito non riceve **l’ACK**, rispedisce il frame Importante: i diversi valori di DIFS e SIFS determinano una **priorità**. Il messaggio di **ACK** ha priorità superiore alla spedizione di un frame generico ed è fondamentale la sincronizzazione dei clock.

**Terminale Nascosto**

Il problema del terminale nascosto si presenta nel caso frequente in cui due clienti associati allo stesso **AP** non ricevono l’uno il segnale dell’altro Infatti, che il segnale di un cliente raggiunga l’AP non implica che raggiunga tutti gli altri nodi.

Quindi uno dei due clienti che non si ricevono (reciprocamente) può iniziare la trasmissione durante la trasmissione dell’altro La ﬁnestra di **collisione** si allarga all’intero frame Il problema è che la collisione perdura per tutto il tempo della trasmissione di ambedue i frame, degradando signiﬁcativamente le prestazioni della rete.

Si parla allora di problema del terminale nascosto.

**Il protocollo RTS/CTS**

Il protocollo **RTS/CTS** allevia questo problema facendo in modo che l’AP coordini gli accessi.

É basato su **un brevissimo frame** (RTS) inviato dalla stazione che ha intenzione di trasmettere.

La brevità del frame limita l’impatto di una eventuale collisione (sempre possibile) L’AP riceve il pacchetto RTS e, dopo SIFS (alta priorità), spedisce un altro brevissimo pacchetto CTS Il problema del terminale nascosto non esiste per il CTS: tutti ricevono l’AP Quindi tutti considerano la linea occupata, sino alla ricezione dell’ACK Il protocollo RTS/CTS viene applicato opzionalmente, e ha un impatto negativo sull’utilizzazione della rete.

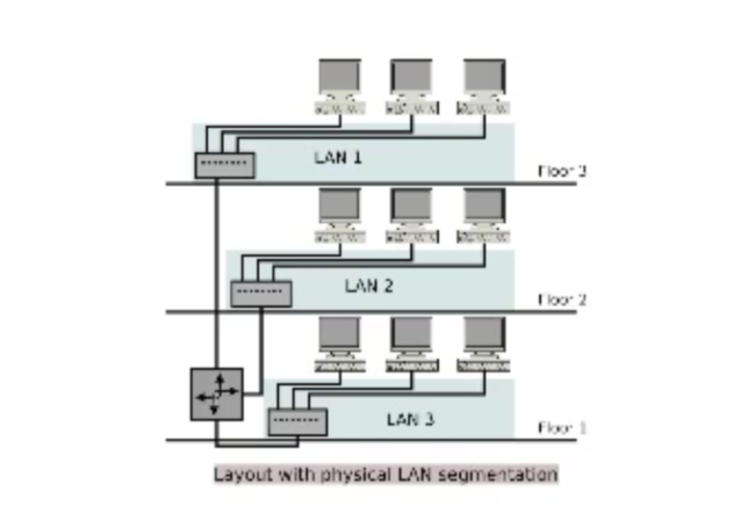
***Reti virtuali***

In [telecomunicazioni](https://it.wikipedia.org/wiki/Telecomunicazioni) e [informatica](https://it.wikipedia.org/wiki/Informatica) il termine VLAN (Virtual Local Area Network) indica un insieme di [tecnologie](https://it.wikipedia.org/wiki/Tecnologia) che permettono di segmentare il [dominio di broadcast](https://it.wikipedia.org/wiki/Dominio_di_broadcast), che si crea in una [rete locale](https://it.wikipedia.org/wiki/Rete_locale) basata su [switch](https://it.wikipedia.org/wiki/Switch), in più [reti locali](https://it.wikipedia.org/wiki/Rete_locale) logicamente non comunicanti tra loro, ma che condividono globalmente la stessa infrastruttura fisica di rete locale.

Le applicazioni di questa tecnologia sono tipicamente legate ad esigenze di separare il [traffico](https://it.wikipedia.org/wiki/Traffico_(telecomunicazioni)) di gruppi di lavoro o dipartimenti di un'azienda, per applicare diverse politiche di [sicurezza informatica](https://it.wikipedia.org/wiki/Sicurezza_informatica).

*La topologia cablata può essere diversa da quella organizzativa.*

*Se il cablaggio su tre piani diversi con 3 reti locali con funzionalità diverse. Questi stanno su 3 sottorete diverse e per motivi di sicurezza non si confondano con altre.*

**

*Ogni nodo può essere destinato a diverse esigenze, per ovviare il problema esiste l* ***OVERLAY*** *virtuale che rispetta le esigenze organizzative ed è sovrapposta alla rete.*

*Ad ogni interfaccia viene associata una* ***LAN*** *virtuale* ***VLAN****, che ha un identificatore di 12 bit*

*Ad ogni porta dello* ***switch*** *viene associato un* ***VLAN ID****, quindi quella porta è associata a quella* ***VLAN****.*

*Per comunicare e comunicare la destinazione del pacchetto, il* ***VLAN ID*** *viene inserito nel pacchetto* ***TAGGING****.*

*Gli* ***switch*** *inoltrano i* ***frame*** *con un certo* ***vlan id*** *solo sulle porte corrispondenti.*

*Ogni* ***VLAN*** *ha il proprio* ***SPANNING TREE*** *per non incorrere a Loop di cicli.*

**Point to point**

L’alternativa ad un collegamento su un supporto condiviso (come Ethernet) è il **collegamento punto a punto**. Utilizzando i collegamenti punto-punto è possibile anche connettere tra loro più sedi situate in due località geograficamente distanti fra loro.

*A differenza dl ethernet che è in broadcast, la* ***PPP*** *che è quella che utilizziamo con l’adsl che è una comunicazione che dal nostro modem tramite un cavo porta connessione sulla centralina.*

***Il frame:***

1. Ha un trailer e inizia e si conclude con una sequenza di bit speciale in 7E esadecimale,
2. Poi un byte solo che serve ad identificare il tipo di payload
3. E poi la checksum

La possibilità che **7E** sia nel payload rende ambigua la funzione del byte: payload o terminazione? Se 7E compare nel payload, un altro byte di escape (01111101, 7D in esadecimale) viene inserito (stuffed) prima di questo, per distinguerlo Se il byte di escape 7D compare nel payload, viene sempre raddoppiato inserendone un altro La tecnica si chiama del byte stufﬁng.

**Parametri di un protocollo e di un link**

overhead: rapporto percentuale tra la dimensione di header (e trailer) e la dimensione del pacchetto.

efﬁcienza: percentuale tra la dimensione del payload e la dimensione del pacchetto.

bitrate della linea: quanti bit vengono trasferiti in un secondo.

larghezza di banda: la differenza tra la massima e la minima frequenza che può passare attraverso una linea I in linea teorica corrisponde alla metà del bitrate (teorema di Nyquist).

throughput: prodotto tra l’efﬁcienza del protocollo e il bitrate della linea (ad es. 100 Megabit al secondo Mb/s)

**Definire la connessione logica LLC (Logical Link Control)**L' **LLC** definisce e controlla il collegamento logico tra i nodi di una rete. Questo sottolivello fornisce i servizi al Network Layer che nascondono i dettagli dovuti alle diverse tecnologie fisiche utilizzate.

**Network**

Il livello 3 si occupa della **traduzione del indirizzo logico dell’host destinatario in indirizzo fisico**, quindi **stabilisce il miglior percorso possibile fra due computer di una rete**. Questo percorso, attraverso altri nodi della rete, può essere definito staticamente cioè rimane fisso per tutta la durata della sessione di comunicazione, oppure può variare dinamicamente, cioè invia pacchetti su percorsi diversi in base alla necessità.

Il livello network ha una funzione di **convergenza**, si occupa quindi di unificare le molteplici tecnologie del livello link.

Questo livello si occupa di **gestire la comunicazione controllando il flusso dei dati** al fine di evitare la Congestione Della Rete.

I dati ricevuti dal livello superiore sono organizzati in pacchetti o datagrammi, che contengono il **Network Header** con l’indirizzo logico dell’host sorgente e dell’host destinatario. Provvede inoltre alla **commutazione dei singoli pacchetti** che compongono il messaggio e ciascun pacchetto viene fornito di un numero progressivo (che permette la ricostruzione dell'intero messaggio) e determina se il cammino deve essere lo stesso per tutti i pacchetti, oppure se i pacchetti possono seguire cammini diversi.

Il livello di rete definisce, inoltre, come devono **funzionare delle reti interconnesse** (Internetworks), e fornisce al 4° livello una connessione End to End tra i due elaboratori coinvolti nello scambio dei dati.

**Indirizzamento Logico**

Ogni nodo connesso è identificato con un indirizzo logico, IP, (Indipendente dall’hardware) che deve essere unico sull’intera Internetwork.

**Gli indirizzi IP** sono composti d 4 numeri di 8 bit, in Ip si indica con una dotted quad 102.34.6.154, e questi ultimi vengono associati ad indirizzi mnemonici come ww.google.com che rappresenta il server che sta gestendo la connessione di Google.

*Ogni pacchetto ha il proprio* ***time to live(****TTL) che è un indice che viene decrementato ad ogni* ***hop****, e a zero viene scartato senza farlo vagar in rete.*

La singola rete IP è un insieme di indirizzi di nodi che hanno in comune il prefisso di bit

Il livello di rete si occupa di scegliere (Routing) il cammino migliore o, in caso di guasto, un cammino alternativo.

**Il percorso si chiama route e seguendo i pacchetti si arriva al nodo di destinazioni.**

La route è costituita da una serie di nodi intermedi detti router attraversati dal pacchetto.

Un router è associato ad una o piu sottereti e i percorsi vengono decisi da criteri diversi, accomunati dall’obbiettivo di non sovraccaricare il router e di cercare il miglio percorso possibile.

Il Routing inoltre è relativo ad un singolo pacchetto mentre La route puo essere comune a piu pacchetti.

Gli algoritmi di Instradamento possono essere di tipo Statico che basano le proprie scelte su informazioni memorizzate in un archivio che viene aggiornato manualmente o di tipo Dinamico che utilizzano misure e stime del traffico sulla rete, in modo da instradare i dati sui percorsi che di volta in volta sembrano piu' promettenti.

**Datagram Encapsulation**

I messaggi ricevuti dal livello superiore vengono incapsulati all’interno di un’unità dati detta datagrams (o pacchetto) alla quale viene messa l’intestazione relativa al livello di rete (Network Layer Header).

**Error Handling and Diagnostics**

Speciali protocolli sono utilizzati a questo livello per permettere ai devices connessi logicamente di scambiarsi informazioni relative allo stato dei nodi o dei devices stessi.

**Fragmentation and Reassembly**

Il livello di rete deve inviare messaggi al livello sottostante Datalink. Alcune tecnologie associate al livello Datalink hanno dei limiti sulla lunghezza del frame **MTU** (Maximum Transmission Unit) che possono spedire. Se il pacchetto del livello di rete è troppo grande questo deve essere diviso in "pezzi" in modo che possa adattarsi alla grandezza del frame. Analogamente quando riceve diversi frame relativi allo stesso messaggio, il Network Layer deve occuparsi di riassemblarli.

Inoltre questo livello si deve occupare della **conversione dei dati** nel passaggio fra un tipo di rete fisica ed un altro tipo di rete con diverse caratteristiche fisiche (traduzione degli indirizzi di rete, nuova frammentazione dei pacchetti, ecc.…).

**Gestione delle Connessioni**

Alcuni protocolli di rete forniscono un servizio di gestione delle connessioni, ovvero richiedono che venga stabilito un canale di comunicazione fisso e dedicato prima che due host possano iniziare a scambiarsi dati.

Altri protocolli invece trasportano semplicemente i datagrammi a destinazione (IP, IPX) senza connessione. Il più importante protocollo legato al Network Layer è IP. Anche il principale protocollo diagnostico **ICMP** (Internet Control Message Protocol), utilizzato insieme ad IP, appartiene a questo livello.

*IL protocollo ICMP è un protocollo di* ***controllo****, che serve ad evitare interferenze, i messaggi vengono incapsulati nella payload, è strettamente collegato con IP.* ***Segnala la non raggiungibilità della destinazione*** *e viene riportato lo header del pacchetto che non può essere riconsegnato, poi manda anche pacchetti di controllo o ne invia uno di conferma* ***PING****.*

**Multicast**

**IGMP**

*L'****Internet Group Management Protocol*** *è un protocollo per la gestione dei gruppi* [***multicast***](https://it.wikipedia.org/wiki/Multicast)***.(*** quando uno stesso pacchetto deve essere inviato a tanti destinatari (ad esempio uno streaming), è scomodo e consuma banda inviare lo stesso pacchetto n volte (1 per ogni destinatario), si crea un gruppo a cui i destinatari si associano e questo pacchetto viene inviato al gruppo ( tramite l'indirizzo del gruppo).  
Saranno poi i router che gestiscono il multicast a sapere dove sono i vari destinatari ed infine saranno gli stessi destinatari a decidere se accettare il pacchetto (perché fanno parte del gruppo) oppure no.)

*Il loro range di indirizzo è di* ***classe D******o gruppo Multicast,*** *hanno anche un indirizzo* ***Mac*** *particolare.*

*Il nodo mittente invia ad un indirizzo Multicast e altre macchine che possono eseguire alcune join per iniziare a ricevere il Multicast trasmesso, possono essere sulla stessa o in diverse sottorete e aderiscono al protocollo* ***IGMP che serve a gestire il Routing per le comunicazioni e registra adesioni e abbandoni.***

*La comunicazione avviene tra nodi vicini con una connessione a livello 2 ed avviene tra router e destinatari di broadcast.*

*I messaggi* ***IGMP*** *sono incapsulati nel* ***Ip*** *il numero di protocollo è 2 il ttl è 1.*

*Il router che supporta l* ***IGPM*** *è particolare, spedisce ad intervalli regolari nella sottorete messaggi di* ***query*** *che servono a rilevare la presenza di altri router Multicast interessati ad un certo gruppo.*

*Gli host o router interessati ad un gruppo inviano un* ***report****, e chi lo riceve diventa membro del gruppo e risponde positivamente alle query del gruppo, e la catena di report raggiungerà un router Multicast che riceve lo stream del mittente.*

***La General query*** *richiede se ci sono interessi pendenti per un certo gruppo, serve quando il router non possa fare query. Quindi non porta messaggi espliciti di iscrizione, ma è captata dalla risposta del ricevente query tengono aggiornate le strutture dati.*

*IL* ***router querier*** *è delegato a spedire messaggi di* ***query****, fa parte di un gruppo speciale multicast che è il 224.0.0.1. ed è quello con l’indirizzo IP inferiore.*

*Per inviare il* ***report*** *invece c’è un* ***algoritmo*** *che limita ad uno solo i messaggi di report.*

*Quando il router manda un messaggio imposta un ritardo spedizione con componente casuale, quando il time out scade.*

**Trasporto**

livello 4 del modello ISO/OSI si occupa di fornire servizi al soprastante livello 5 (livello sessione) e per raggiungere tale scopo raccoglie i dati offerti dal sottostante livello 3 (livello rete). Lo scopo del livello di trasporto è fornire **un canale logico e affidabile di comunicazione end-to-end per "pacchetti"**. Il nome trasporto per tale livello puo' quindi trarre in inganno in quanto, non implementa alcun meccanismo di trasferimento logico e fisico dei dati sul canale quali accesso multiplo, indirizzamento e commutazione.

Ma si occupa di supplire alle mancanze delle funzionalità del trasferimento in termini di affidabilità, implementando le suddette funzioni come garanzie sul trasporto stesso, cioè chiudendo il cerchio su tutto ciò che il trasporto dovrebbe fare e garantire. Di seguito vengono riportati i servizi che vengono, in genere, offerti dal livello di trasporto. Èbene ricordare che nessuno di tali servizi è obbligatorio. Di conseguenza, per ciascuna applicazione è possibile scegliere il protocollo piu' adatto allo scopo:

[**Servizio orientato alla connessione**](http://www.isistassinari.gov.it/progettodicembre/reti/L4.html#servizio)

[**Corretto ordine di consegna**](http://www.isistassinari.gov.it/progettodicembre/reti/L4.html#corretto)

[**Trasferimento affidabile**](http://www.isistassinari.gov.it/progettodicembre/reti/L4.html#trasferimento)

[**Controllo di congestione**](http://www.isistassinari.gov.it/progettodicembre/reti/L4.html#controllo)

[**Orientamento al byte**](http://www.isistassinari.gov.it/progettodicembre/reti/L4.html#orientamento)

[**Multiplazione**](http://www.isistassinari.gov.it/progettodicembre/reti/L4.html#multiplazione)

**Servizio orientato alla connessione**  
Il livello di trasporto si incarica di realizzare una connessione **persistente** del tipo necessario al livello di sessione per ogni connessione richiesta, che viene poi chiusa quando non è piu' necessaria.   
**Corretto ordine di consegna**   
Il messaggio che l’host sorgente deve trasmettere generalmente viene scomposto in pacchetti numerati progressivamente, vengono messi in sequenza sulla rete: a causa della dinamicità del traffico non e' detto che tutti percorrano lo stesso canale e quindi arrivino a destinazione nello stesso ordine con cui sono partiti. È compito del livello di trasporto effettuare **la ricostruzione esatta dei dati rimuovendo possibili errori**.   
**Trasferimento affidabile**   
Il protocollo si occupa di **garantire che tutti i dati inviati vengano ricevuti**; nel caso il servizio di rete utilizzato perda pacchetti, il protocollo di trasporto si occupa dei protocolli per la ritrasmissione dei pacchetti corrotti.   
**Controllo di flusso**   
Se gli host coinvolti nella comunicazione hanno velocita' di trasmissione differenti, il livello di trasporto si occupa di evitare errori di over-running. Può capitare che un pc piu' veloce "inondi" di dati uno più lento portando alla perdita di pacchetti. Mediante il controllo di flusso, un host in "difficoltà " puo' chiedere di abbassare il tasso di trasmissione in modo da poter gestire le informazioni in ingresso.   
**Controllo di congestione**   
Il protocollo riconosce uno stato di congestione della rete e adatta di conseguenza la velocita' di trasmissione.   
**Orientamento al byte**   
Invece che gestire i dati in base ai pacchetti, viene fornita la possibilità di vedere la comunicazione come uno **stream di byte**, in modo da semplificarne l'utilizzo.   
**Multiplazione**  
Il protocollo permette di stabilire diverse connessioni contemporanee tra gli stessi due host, tipicamente utilizzando l'astrazione delle porte. Nell'uso comune diversi servizi utilizzano porte logiche di comunicazione diverse.

**Protocollo Udp**

IL protocollo **UDP** è un protocollo di trasporto, diverso da quello **link** anche se hanno molte usi comuni, utilizza pacchetti **IP** per trasferire i dati, avviene in modo **unidirezionale**, da un mittente ad un destinatario attraverso datagrammi che sono singoli pacchetti di informazioni auto contenuto e indipendente ed è piu capiente del pacchetto IP. Non c ‘è controllo di flusso e quindi possono essere perduti i pacchetti e possono essere ricevuti in modo disordinato ed è simile ad Ip.

Le **porte** **UDP** sono importanti per definire una caratteristica di udp che è il **multiplexing** cioè la possibilità di raggiungere applicazioni diverse che risiedono sullo stesso nodo raggiungibile da un indirizzo IP.

Quando voglio raggiungere una certa applicazione e spedire un certo pacchetto udp devo specificare anche la porta relativa all’applicazione che riceverà quel datagramma, poi a livelli Ip verrà associato un indirizzo Mac. Il numero di porte è rappresentato d’un intero 16 bit.

**UDP** è piu capiente di IP e porta fino 65Kbyte di dati in un singolo datagramma, non rispedisce pacchetti persi non recapita messaggi nello stesso ordine e non consente datagrammi duplicati.

L’organizzazione della comunicazione in **datagrammi** facilita protocolli basati su transizioni.

La caratteristica di non ritrasmettere l’informazione lo rende adatto si sistemi in real –time.

**L’assenza di un controllo di flusso rende inoltre possibile la comunicazione da uno a molti.**

Puo essere quindi utilizzato per comunicazioni da uno a molti **multicast** che fanno parte della classe D degli indirizzi e hanno come particolarità di rappresentare un intero gruppo, iniziano con la sequenza 1110.

IL nodo interessato a ricevere il multicast si iscrive al gruppo corrispondente al indirizzo in classe D con l’operazione **join** realizzata dal protocollo **IGMP** (allargato ad internet e relativamente vincolato di una parte di internet).

Un'altra possibilità di fare **multicast** è utilizzare un indirizzo derivato da quello della sottorete che è associato a tutti i nodi della sottorete a livello 2.

L’indirizzo è di **broadcast** è dato dall’indirizzo di rete composto da tutti i bit esclusi dal prefisso 1, spedendo su questo indirizzo, tutti i nodi della sottorete ricevono il datagramma.

La modalità invasiva è pericolosa, e l’uso è riservato alle funzioni di sistema.

**Interfaccia Udp (Api)**

Offre i propri servizi ad un’interfaccia e questa si chiama **API** (Application Programming Interface)

L’accesso è consentito al utente generico.

Bisogna avere una libreria di programma per poterlo utilizzare ed è l’API.

L’astrazione fondamentale è la **socket** (paragonabile al file), che corrisponde al punto di arrivo o partenza dei datagrammi.

IL **socket** è una sorta di input output da cui posso leggere e scrivere, lo scambio di un datagramma avviene tra due **socket** uno che manda e uno che riceve.

socket in ingresso bisogna specificare la porta attraverso cui riceverò i datagrammi

l IP puo essere implicito perché è quello dalla macchina in cui sono.

Per descrivere un socket in uscita è necessario **specificare la porta e l ip del destinatario**.

L IP locale puo essere lasciato implicito perché la porta è scelta a caso.

**coppia ip: porta**

**Protocollo Dhcp**

È un protocollo che serve alla gestione della rete e utilizza **UDP**, e non il TCP, perché è in broadcast e il tcp non supporta il broadcast, si applica nel caso in cui i **nodi che si collegano ad una certa sottorete cambiano**, un esempio è l’ufficio dove i dipendenti possono spostare il loro pc.

In molti casi è utile la configurazione plug and play dei nodi di una rete.

**Configura automaticamente un’interfaccia di rete assegnandogli un IP**,

Per il suo funzionamento si usa una funzione **broadcast** di ip e Mac.

Le sue funzioni sono:

* **identificare** gli indirizzi ip
* **associare** questo ip ad uno Mac dell’interfaccia
* determinare il **routing** semplificato per il nodo che è stato appena collegato
* fornire altre informazioni come l’indirizzo del **server dns**
* viene configurato un **server Dhcp** nella sottorete
* il server ha a disposizione un **insieme di ip da assegnare** e per ogni nodo si può assegnare un ip solo.
* nelle piccole lan spesso il server Dhcp è sul router

**Come funziona:**

Il nodo appena connesso invia in broadcast sulla sottorete nella quale è collocato un messaggio di richiesta: **discovery**, che è incapsulato in un datagramma udp.

Per realizzare il broadcast l’indirizzo ip è impostato a 255.255.255.255.

La porta su cui è inviato è la 67 che è riservata a **Dhcp**.

L’indirizzo ip del mittente è ignoto 0.0.0.0.

I server ricevono il datagramma di **discovery**, i server Dhcp rispondono con un pacchetto **offer** con:

1. l’indirizzo Mac corrispondente a quello del cliente
2. l’indirizzo ip a lui assegnato dal server
3. le altre info richieste

il **server** **Dhcp** considera impegnato l’indirizzo assegnato.

Il **client** può ricevere piu di una **offer** come piu sono i server ed ha un margine di decisione

Il **cliente** deve accettare un **offert** e per selezionare la **offer** i criteri sono vari, dopo aver scelto il cliente spedisce in broadcast un pacchetto di richiesta: **request** simile alla **discovery**.

Il **server** la cui offerta è stata assegnata risponde con un pacchetto **ack**, dal router verso il nodo, dal server verso il client simile a quello di **offer**, e conferma che quello è il suo indirizzo.

Il **server** la cui **offer** non è stata accettata liberano l’indirizzo proposto.

Quando riceve l **ack** il cliente può iniziare ad **utilizzare** l’indirizzo che gli è stato assegnato.

Questo collegamento ha una vita (**lease time**), allo scadere della quale bisogna rieseguire il protocollo.

Il protocollo **hdcp** è un esempio di transazione udp con uno scambio di datagrammi.

Tutto ciò avviene se i messaggi vengono recapitati correttamente e non vengono persi come può accadere in **UDP** che non garantisce appunto il recapito, e se questo pacchetto **discovery** viene perduto o non viene trattato il cliente ne rispedirà uno nuovo finché non riceve una **offer** entro il **timeout**.

Se è troppo lento il server può ricevere piu pacchetti **discovery**.

Il server risponde a richieste multiple dallo stesso client usando sempre gli stessi criteri di selezione dell IP per il pacchetto **offer.** z

Se il pacchetto **offer** viene perduto, lo stesso time-out provvederà a far spedire un nuovo pacchetto **discovery.**

Lo scambio **request-ack** funziona in modo simile.

***Routing***

***Routing Esterno***

*La consegna di un pacchetto:*

*Gill cammino di un pacchetto IP consiste in una sequenza di passi di ricezione e rispedizione del pacchetto detto* ***hop.***

*il percorso completo, dal mittente al destinatario viene detto* ***route,*** *e* ***routing*** *è l attivita che lo determina.*

*il nodo che riceve e rispedisce il pacchetto viene chiamato* ***gateway****.*

*Il* ***gateway*** *calcola la prossima destinazione del pacchetto sulla base della sua destinazione finale, utilizzando una struttura dati detta* ***tabella di routing****, della destinazione è nota la* ***sottorete****, indicata come un prefisso* ***ip****.*

*Il* ***calcolo della tabella di routing per ciascun gateway è una funzione che può essere molto complessa*** *e vitale, e viene svolta da un nodo detto* ***router****, nei casi piu semplici lo stesso nodo svolge sia la funzione di* ***router*** *che di* ***gateway****.*

*La* ***tabella di routing*** *può quindi essere rappresentata in modo semplificato con due colonne:*

* *la sottorete di destinazione*
* *un router adiacente, dunque raggiungibile con un singolo hop*

*Per rendere fattibile un routing* ***globale****, viene introdotta una struttura gerarchica, suddividendo la rete globale in* ***sistemi autonomi (as)***

*Un* ***sistema autonomo*** *contiene un insieme di router che scambiano informazioni attraverso un protocollo comune, i* ***router*** *e le* ***sottoreti*** *controllate vengono gestite dalla stessa organizzazione (es. telecom), gestione amministrativa centralizzata. Il* ***grafo*** *risultante è sempre connesso e correla i vari router, a meno di guasti che possono alterare la funzionalità del sistema autonomo.*

*Si differenziano due generi diversi di Routing:*

1. *Dentro un sistema autonomo attraverso router interni interior routing*
2. *Tra sistemi autonomi distinti, messi in contatto da router di confine exterior routing*

*I protocolli in uso in internet per i due livelli sono diversi e sono classificati in:*

1. ***Interior gateway protocol (rip)*** *che trattano lo scambio di informazioni tra router dello stesso sistema autonomo*
2. ***Esterior gateway protocol (Bgp)*** *che trattano il Routing tra router di sistemi autonomi diversi.*

***BGP***

*Il protocollo* ***BGP*** *tratta* ***semplicemente lo scambio di informazione tra i gateway,*** *senza specificare* ***algoritmi di determinazione*** *dei cammini. Non c’è un algoritmo che decide il Routing, ma sono le informazioni che vengono scambiate tra i gateway che determinano la tabella di Routing, questo perché a livello di exterior Routing la selezione di un certo cammino non avviene con un algoritmo astratto, ma deve essere suggerito dall’amministrazione del sistema in base a quelli che sono gli accordi commerciali o di peering tra i vari fornitori.*

*La selezione di un cammino avviene sulla base di politiche complesse, non per semplice ottimizzazione dei percorsi.*

*Un’informazione è quali servizi l’amministrazione del* ***AS*** *decide di utilizzare ed offrire, altre ragioni anche come quelle economiche politiche e di sicurezza.*

*La motivazione che spinge un'organizzazione all'utilizzo del protocollo BGP per scambiare le proprie route in internet è la necessità di avere una rete* ***multihomed****. Per rete multihomed si intende una rete che acquista transito internet da più provider di livello superiore (o anche più link con lo stesso provider) e che deve garantire la raggiungibilità dall'esterno dei propri servizi, tramite un set di indirizzi IP pubblici. Infatti, in una situazione standard senza l'uso del protocollo BGP, ogni link verso internet sarà dotato della sua classe di indirizzamento IP, non condivisibile con altri link. Risulta evidente quindi che pubblicando un servizio (esempio: sito web) sull'indirizzo IP raggiungibile direttamente dallo specifico link, alla caduta del link stesso quel servizio risulterà irraggiungibile dalle altre reti.*

*Invece, utilizzando il protocollo BGP per propagare in internet la propria classe di indirizzi IP, gli indirizzi stessi saranno raggiungibili tramite qualsiasi link verso il quale siano state propagate le route. (figura 2)*

***BGP realizza lo scambio di informazioni di Routing tra AS distinti.***

*In BGP abbiamo due funzionalità diverse e sono quella di* ***gateway*** *e quella di* ***speaker.***

*L’intero internet viene quindi rappresentato come un* ***grafo,*** *i cui nodi sono sistemi autonomi e gli* ***archi*** *sono connessioni tra* ***gateway*** *appartenenti ai sistemi autonomi.*

***I AS*** *scambiano tra di loro un’immagine coerente della raggiungibilità di un altro* ***AS****, queste info viaggiano al interno dello stesso AS e come fare a raggiungerle, quindi delle sottoreti interne al AS e ai AS per cui possono gestire il traffico.*

*La gestione del protocollo BGP non è necessariamente associata al nodo che ospita il gateway, anzi dal punto di vista delle applicazioni sono distinte*

*Distingue due tipi di nodi:*

1. ***border gateway*** *opera il trasferimento dei pacchetti*
2. ***Bgp speaker*** *riceve le operazioni di Routing e aggiorna e tabelle, colleziona le informazioni di Routing.*

*Due* ***AS*** *possono comunicare tra di loro, ma devono verificarsi due condizioni fondamentali:*

1. *Deve esistere un collegamento nel* ***data link layer*** *tra i due* ***as*** *e devono avere ciascuno un* ***border gateway*** *agganciato a quel* ***link****.*
2. *Entro ciascun* ***AS*** *i* ***Bgp speakers*** *devono condividere lo stesso collegamento data link con il* ***border gateway****.*

*Il collegamento tra i due Bgp* ***speaker*** *non richiede l’intervento del Routing interno al AS.*

*le comunicazioni tra i* ***Bgp speakers*** *passano necessariamente attraverso i due* ***border gatewa****y.*

*\*le due funzioni sono completamente indipendenti, quini possono essere associate a due nodi distinti, ma anche cumulate sullo stesso nodo che prenderà entrambe le funzionalità.*

*Un AS, può trovarsi ad essere:*

* *La destinazione d’origine, ed in questi casi si tratta di traffico locale*
* *Un tramite, ed in questo caso si tratta di traffico di transito*

*In caso di transito possiamo distinguere due circostanze:*

1. *È uno* ***stub*** *cioè adiacente ad un solo as, e di conseguenza non può essere luogo di transito.*
2. *Il as* ***è multihomed*** *e pur essendo vicino ad altri as non permette il transito*
3. *Il as è di* ***transito,*** *ed allora è adiacente a piu di un AS e ne accetta il transito*

*Il Protocollo BGP non introduce vincoli sulla topologia della rete, ed è in grado di evitare la formazione di cicli.*

***Le Route In Bgp***

***Il protocollo BGP rende pubbliche, e quindi utilizzabili da altri AS, route che rendono raggiungibili i nodi e le sottoreti interne al as.***

*Le route sono cammini completi, non semplici tabelle che associano un prossimo router alla destinazione*

*Quindi i nodi* ***SPEAKERS*** *possono applicare politiche complesse riguardo ai flussi di informazione in transito, ad esempio per evitare cicli.*

*Un AS che accetta traffico in transito può anche differenziare il servizio offerto alle altre as.*

*Una caratteristica vitale è che i* ***Bgp speaker*** *interni alla stessa* ***as*** *applichino politiche consistenti*

*le route che vengono rese pubbliche da* ***Bgp*** *diversi devono essere tra loro consistenti*

***funzioni Bgp***

***BGP*** *svolge 3 funzioni:*

1. *Acquisizione dei adiacenti, l****’adiacenza*** *è una proprietà* ***topologica*** *che mi dice che due as attraverso due* ***gateway*** *e* ***2 speakers*** *sono in grado di comunicare e inviare routing esterni*

*per questo vengono utilizzati messaggi tipo* ***open***

1. ***Mantenimento*** *del collegamento con i* ***Bgp******adiacenti****, per questo vengono utilizzati comandi di tipo* ***keepalive*** *e* ***notification****.*
2. ***Aggiornamento*** *della route per le reti che possono essere raggiunte, per questo vengono usati messaggi del tipo* ***update****.*

*Il protocollo* ***BGP*** *utilizza messaggi singoli inseriti in uno stream, non inseriti nella conversazione.*

***Utilizza un protocollo TCP*** *(il flusso è persistente, al contrario di UDP) sulla porta 179 e la comunicazione conseguentemente è affidabile.*

***I messaggi di Bgp***

*I messaggi utilizzati da* ***BGP****, sono tutti composti da un* ***header****, comune ai 4 tipi di messaggio, e da una parte* ***VARIABILE****.*

*La struttura dello header è la seguente:*

* ***marker*** *(16 ottetti): si tratta di una sequenza di ottetti che il destinatario è in grado di prevedere, quindi è utilizzata per l’autenticazione del messaggio, e per rilevare la perdita di consistenza tra i due partner. può essere indicato il non utilizzo riempiendo il campo di 0xff*
* ***lunghezza*** *(2 ottetti): lunghezza del messaggio header incluso, in ottetti. necessaria in quanto il formato non prevede un trailer, e la trasmissione utilizza il protocollo tcp, tuttavia la lunghezza massima di un messaggio è 4096 ottetti.*
* ***tipo****: specifica il tipo di messaggio tra*

1. ***OPEN, UPDATE, NOTIFICATION E KEEPALIVE****.*

***Acquisizione di un vicino***

*Il messaggio* ***open*** *viene inviato non appena si riesce a stabilire la connessione* ***tcp,*** *per acquisire il vicino.*

*Ciascuno dei due partner invia la* ***open*** *e se l'altro accetta risponde con un messaggio* ***keepalive****, e serve periodicamente per comunicare che c’è ancora comunicazione per evitare che la connessione resti attiva indefinitivamente.*

*IL messaggio* ***open:***

1. ***version****: 1 ottetto ed è la versione del protocollo*
2. ***my as*** *:2 ottetti è l’identificatore dell as cui appartiene il mittente*
3. ***hold time*** *2 ottetti: il massimo intervallo proposto tra due messaggi keepalive*
4. ***bgp id*** *4 ottetti indirizzo ip del bgp speakers del mittente*
5. ***lunghezza parametri*** *lunghezza dei parametri*
6. ***parametri*** *puo contenere la chave di autentificazione*

*Il messaggio* ***update:***

*Serve per aggionare il* ***routing****, contiene delle registrazioni* ***RIB*** *(routing information basis)*

*ogni* ***rib*** *desrive una route per raggiungere il* ***as****.*

*sono distinti in 3* ***database:***

1. ***adj. ribs-in****: acquisiti dall’esterno, tramite messaggi di* ***update***
2. ***loc-rib*** *ottenute dalle precedenti applicando politiche locali*
3. ***ad-ribs-out*** *sottoinsieme delle precedenti, comunicate all’esterno, tramite messaggi di* ***update****.*

***L’update*** *serve a comunicare la modifica del* ***RIB****, puo invalidare diverse* ***route****, e puo* ***abilitarne una sola nuova****.*

*Il messaggio comprende* ***3 campi di lunghezza variabile****, i primi due sono composti da un sotto campo di due ottetti che ne specifica la lunghezza e un altro campo contenente l’informazione. Le dimensioni del 3 sono ricavate per differenza.*

1. ***Il primo specifica le route che vengono invalidate.*** *Ogni route viene indicata con le sottorete di cui si desidera cancellare le precedenti informazioni di raggiungibilità. Un indirizzo di sottorete è indicato come:* ***ip, lunghezza prefisso.***
2. ***Gli altri due campi indicano una nuova ROUTE*** *da rendere pubblica: il primo indica una serie di ATTRIBUTI della ROUTE, il secondo la destinazione della route*

*Il campo che indica la* ***destinazione*** *viene espresso con un elenco di prefissi, ciascuno indicato come una coppia****: IP, PREFISSO.***

*Messaggio* ***keepalive e notification***

*I due messaggi sono usati per gestire il collegamento*

*Il messaggio* ***keepalive*** *contiene solo lo header e viene spedito periodicamente da ciascun partner per indicare la mantenuta adiacenza, nel* ***open*** *viene definito l’intervallo minimo tra i due messaggi keepalive, nel campo* ***hold time****.*

*Il messaggio* ***notify*** *indica la rilevazione di un errore: contiene una descrizione sintetica dell’errore, ad una parte si dati di lunghezza variabile.*

***Gli attributi*** *che devono essere necessariamente presenti sono:*

* ***L’origine dell’informazione*** *che ho utilizzato*
* ***Il cammino la route che voglio utilizzare*** *per arrivare il nodo destinazione, la sottorete di destinazione, viene resa pubblica ed è rappresentato con gli indici dei* ***AS*** *che il cammino attraversa*
* ***Il prossimo border gateway*** *da utilizzare per il cammino, indicato con il suo indirizzo* ***IP***

***Attributo route***

*Ci sono piu* ***route,*** *e le* ***route*** *sono l’insieme di cammini che possono essere intrapresi.*

*per costruire questa struttura dati, il* ***bgp speaker*** *aggiunge sé stesso, il suo indice, secondo certe regole, alla struttura analoga che ottiene elaborando quelle ottenute dai vicini.*

*La nuova* ***route*** *viene poi comunicata ai vicini con un’operazione chiamata* ***advertisement****.*

*quindi viene costruita al contrario, mi arriva un avvertimento per una certa* ***route****, io aggiungo il mio* ***id*** *come primo* ***nodo*** *e posso utilizzare questa nuova route per fare* ***advertisement****, quindi io vedo la* ***route*** *che è vicino a me per arrivare alla destinazione finale, e posso vedere se in questa route ci sono cicli e posso evitare di inserirmici.*

***Routing Interno***

***Rip***

*Nel caso piu semplice la tabella di routing è statica, impostata manualmente o automaticamente.*

***Esistono algoritmi in grado di configurare dinamicamente le tabelle di routing*** *sulla base dello scambio tra tabelle tra i router.*

*Lo scambio avviene tramite pacchetti IP O UDP, con la possibilita di perdere o invertire pacchetti.*

*Questi algoritmi possono modificare significamente le tabelle di routing.*

*Sono di due generi:* ***Distance vector*** *e* ***link state****. Il* ***costo*** *è associato al traffico uscente dal router. La sottorete a cui appartiene è zero.*

***rete:*** *rete a cui voglio arrivare*

***prossimo router****: il piu adiacente o quello che permette percorso piu breve*

***metrica****: lunghezza cammino*

*La rete che viene controllata da* ***RIP*** *deve essere relativamente piccola, non superiore a 16 passi.*

***RIP*** *si basa su un timer, circa 30 secondi entro i quali invia la propria* ***distance******vector*** *agli altri router comunicando la sua distanza dalle altre sottoreti.*

*Le sottoreti adiacenti hanno distanza zero.*

*Il contenuto di questi messaggi è detto* ***advertisment*** *e la sua spedizione puo avvenire in bradcast o in multicast.*

*Il router che riceve un* ***distance******vector*** *dal vicino* ***incrementa di un'unità la distanza*** *relativa a ciascuna riga e per ciascuna riga cerca nella propria tabella di routing una riga con destinazione uguale: A.*

*Se non viene trovata nessuna riga con destinazione uguale, allora viene aggiunta alla tabella, B.*

*Se viene trovata una riga con stessa destinazione, se proviene dal router indicato come prossimo router si aggiorna la route con lo stesso valore, altrimenti alla distanza indicata in tabella si sostituisce la minore tra le due, quella nota e quella appena ricevuta.*

*Spesso i protocolli di routing vengono formalizzati in un* ***grafo*** *a nodi in cui i nodi rappresentano le possibili origini e destinazioni del routing e gli archi, formati da triple, corrispondono a origine, destinazione e metrica.*

*Questa formalizzazione non corrisponde esattamente alla situazione di internet, comunque algoritmi teorici possono essere adattati ad una implementazione.*

***Se un link si interrompe:***

*Non c'è modo per indicare l'interruzione di un* ***link****, per questo motivo il funzionamento dei* ***RIP*** *è basato su un timer e la rilevazione delle interruzioni è associata ad un* ***timeout****; se il messaggio di aggiornamento previsto non viene ricevuto entro il limite si considera il link interrotto.*

*Per rimediare al problema dei messaggi interrotti si può ricorrere al sistema del* ***route poisoning****, che consiste nell'assegnare il valore 16 (infinito per la distanza trattata con RIP la cui lunghezza del cammino deve essere inferiore) al collegamento con il router assente.*

*Quindi si capisce perché RIP non è adatto a reti dinamiche ed estese.*

***Algoritmo Bellman Ford***

L'algoritmo di Bellman-Ford calcola i [cammini minimi](https://it.wikipedia.org/wiki/Shortest_path) di un'unica sorgente su un [grafo](https://it.wikipedia.org/wiki/Grafo) diretto pesato (dove alcuni pesi degli archi possono essere negativi). L'[algoritmo di Dijkstra](https://it.wikipedia.org/wiki/Algoritmo_di_Dijkstra) risolve lo stesso problema in un tempo [computazionalmente](https://it.wikipedia.org/wiki/Complessit%C3%A0_computazionale) inferiore, ma richiede che i pesi degli archi siano non-negativi. Per questo, Bellman-Ford è usato di solito quando sul grafo sono presenti pesi degli archi negativi.

L'idea è quella di partire dal nodo sorgente e cominciare a guardare i nodi adiacenti, cioè che distano un passo solo. Si assegna loro il valore del costo per raggiungerli (determinato dal costo dell'arco + il valore del nodo da cui si è partiti, che in questo caso è 0 visto che stiamo partendo dalla sorgente). A questo punto per ciascuno dei nodi raggiunti si procede allo stesso modo: si vedono i nodi che gli distano uno e si assegna loro il valore dell'arco percorso per raggiungerli più quello già assegnato al nodo da cui si è partiti (assegno loro il nuovo valore solo se è più piccolo di quello che già avevano: la prima volta che li raggiungi è certamente più piccolo in quanto abbiamo detto che inizialmente diamo a tutti i nodi valore infinito). Ad ogni passaggio i nodi raggiunti lo step precedente (considera nodi raggiunti solo quelli a cui hai aggiornato il valore) diventano punto di partenza per raggiungere i nodi adiacenti il cui valore diventa (se è minore di quello che già possiedono) quello dell'arco percorso per raggiungerli + il valore del nodo da cui li si è raggiunti (e in tal caso diventano a loro volta nuovi punti di partenza). Se il grafo ha N nodi è certo che dopo N-1 giri tutti i nodi hanno a loro assegnato il costo minimo per essere raggiunti dal nodo sorgente. Ovviamente ad ogni giro quando aggiorni il valore di un nodo devi salvarti il percorso associato per raggiungerlo dalla sorgente, così quando avrai finito le iterazioni oltre ad avere tutti i costi minimi avrai anche i percorsi associati cioè i percorsi a costo minimo per raggiungere ogni nodo del grafo dal nodo sorgente.

Svantaggi

Modesta scalabilità della rete.

Convergenza lenta: i cambiamenti nella topologia della rete non si riflettono velocemente nella composizione delle tabelle, poiché gli aggiornamenti sono fatti nodo dopo nodo.

Conto all'infinito: se si interrompe il collegamento con un nodo, gli altri nodi possono impiegare un tempo infinito per aumentare gradatamente la stima della distanza per quel nodo a meno di non adoperare uno scalare come soglia, oltre il quale, il nodo viene considerato non raggiungibile e quindi fuori dalla rete.

***Routing Parte 2***

***Ospf Base***

***Ospf: Protocollo Di Routing Link State Interno***

*E’un protocollo che si basa sullo* ***scambio di tabelle****, progettato con la finalità di configurare il routing all'interno di un sistema autonomo. L'evoluzione di internet inizialmente portò alla scelta di un algoritmo* ***Bellman-Ford******(Distance Vector)*** *con protocollo* ***RIP****, sostituito in un secondo tempo dall'algoritmo* ***Dijksrta (Link state)*** *con protocollo* ***OSPF****. La ragione alla base di questo cambiamento sta nelle migliori prestazioni del nuovo algoritmo.*

***Vincoli alla topologia della rete con OSPF:***

* *la rete è caratterizzata da* ***aree*** *e da un* ***backbone***
* *ogni area contiene almeno un router adiacente ad un router del* ***backbone*** *(router di confine)*
* *un'area diversa dal backbone non può essere adiacente ad un router che non sia nel backbone.*

*Questa topologia di rete, aderente alla realtà dei sistemi autonomi, si presenta come* ***un sistema a stella*** *in cui il centro è il* ***backbone*** *e alla periferia si trovano le aree.*

***Le aree:*** *il problema del routing viene suddiviso in aree, i router interni ad una certa area avranno un database degli stati dei link della stessa area.*

*Visto che gli algoritmi di link state richiedono che tutti i partecipanti vengano informati di tutti i link, dividendo il sistema in aree si limita la diffusione alle singole aree interessate.*

***Il backbone****: è il punto di contatto tutte le aree; i router che appartengono al backbone sono detti* ***border router*** *e sono quelli che effettuano i collegamenti tra le aree.*

***Tutti i router nel backbone devono essere tra loro contigui.***

*Entro ciascuna area interessa i router interni, che propagano* ***link******state****, e quelli di confine, che generano i link state della route.*

***Il flooding*** *è limitato entro ciascuna area e entro il backbone,quindi la suddivisione limita fortemente il numero di messaggi. Entro ciascuna area il flooding interessa:*

* *i router interni che propagano link state e ne generano al variare della metrica dei link a loro collegati*
* *i router di confine che generano il link state delle route calcolate verso le sottoreti in altre aree come se fossero propri link*

*Per calcolare il costo di questi link vengono utilizzati i risultati del routing entro il backbone ed entro l’area di destinazione. Esistono anche dei router di confine del Sistema Autonomo, che generano i link state verso le sottoreti in altri sistemi autonomi.*

***Generalità del protocollo OSPF:***

* *poggia direttamente su* ***IP***
* *le comunicazioni riguardano lo stato dei link che modificano le loro caratteristiche*
* *ogni router mantiene una tabella che descrive lo stato di tutti i link della sotto-rete di appartenenza*
* *i router di confine mantengono anche i costi verso le reti raggiungibili dal backbone*
* *ogni router misura le prestazioni dei link locali*
* *per ottenere la comunicazione da tutti i router viene usata la tecnica del flooding*

*Ogni router riceve quindi da tutti gli altri una* ***tabella con i link state dei router appartenenti alla stessa area****, i dati ottenuti vengono rappresentati come un grafo su cui viene applicato l'algoritmo* ***Dijikstra****.*

*Le route calcolate da un albero x hanno l'aspetto di un albero di cammini minimi con radice x.*

***Algoritmo di Routing***

*Tramite il flooding ogni router viene a possedere una tabella con i*

*link state relativi ai router della sua area*

*I dei link reali interni all’area di origine*

*I dei cammini nascosti attraverso il backbone e l’area di destinazione*

*Utilizzando queste informazioni genera la propria tabella di routing*

*I Viene applicato l’algoritmo di Dijkstra*

*Ciascun router fa la propria parte: il procedimento è distribuito*

*La route effettivamente seguita da un pacchetto è determinata*

*dalle tabelle di routing dei router incontrati sul percorso*

*I hanno l’aspetto di un albero di cammini minimi (SPF) per ciascuno*

*dei router.*

***Albero dei cammini minimi:***

1. *[****inizializzazione****]per i vari nodi del grafico viene definita una distanza preliminare con cui è anche registrato il primo hop sulla route preliminare, la distanza viene poi perfezionata.*

*L'algoritmo parte inizializzando la distanza preliminare della radice a 0, e quella di tutti gli altri nodi a ∞.*

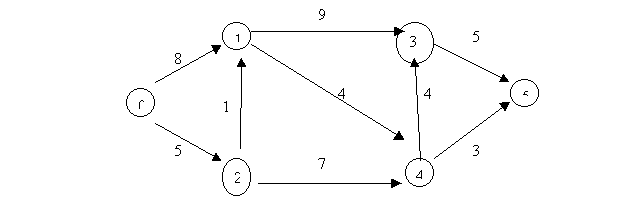
1. *[****iterazione****]inizialmente l'insieme dei nodi raggiunti è vuoto quindi si iterano tutti i passi fino a che non si raggiungono tutti i nodi. Partendo dalla radice:*

* *si seleziona il nodo non raggiunto con distanza preliminare minore, lo si inserisce tra quelli raggiunti;*
* *poi si calcola la distanza dalla radice di tutti i nodi adiacenti a quello selezionato e per ciascuno di questi nodi si aggiornano la distanza minima preliminare ed il primo hop.*

***Dijskra***

*L'algoritmo di Dijkstra consente di selezionare gli shortest path (cammini minimi) in un grafo ciclico caratterizzato da archi con pesi non negativi. Il cammino minimo è il percorso che permette di unire due nodi distinti del grafo. L'algoritmo è utilizzato nell’ottimizzazione della gestione delle reti idriche di trasporto e di rete.*

***# Alla funzione vengono passati il grafo della rete, definito***

***# dall’insieme "nodi" e dall’insieme "archi", che sono coppie***

***# di nodi collegati da un link. La radice dell’albero (cioè***

***# il nodo che esegue l’algoritmo) è nel parametro "radice".***

***L’algoritmo svuota progressivamente l’insieme dei "nodi" aggiungendo***

***# quelli che rimuove al dizionario "raggiunti" e calcolandone la***

***# distanza. Il dizionario "raggiunti" è la tabella di routing,***

***# e associa alla destinazione una tripla (ultimo hop, distanza,***

***# primo hop), e viene aggiornato con il risultato dell’algoritmo.***

***# Il primo elemento della tupla, l’ultimo hop, è funzionale***

***# all’algoritmo, ma non è rilevante ai fini del routing.***

***while*** *nodi: # Si arresta quando l’insieme "nodi" è vuoto*

*# Trovo nodo con distanza minima tra quelli raggiunti*

*nodo\_min = None*

***for*** *nodo* ***in*** *nodi:*

***if*** *nodo* ***in*** *raggiunti:*

***if*** *nodo\_min* ***is*** *None: nodo\_min = nodo*

***elif*** *raggiunti[nodo] [1] < raggiunti[nodo\_min] [1]: nodo\_min = nodo*

***If*** *nodo\_min* ***is*** *None:* ***break***

***print****("Elimino "+nodo\_min, nodi)*

*nodi.remove(nodo\_min)*

*distanza\_min = raggiunti[nodo\_min] [1]*

*# Calcolo mappa archi uscenti da min --> distanza da min*

*distanze\_da\_min=defaultdict(****list****)*

***For*** *arco* ***in*** *archi.keys ():*

***if*** *arco [0] == nodo\_min: distanze\_da\_min [arco [1]] =archi[arco]*

***if*** *arco [1] == nodo\_min: distanze\_da\_min [arco [0]] =archi[arco]*

***OSPF AVANZATO***

***Interazione con protocolli di Exterior Routing:***

*il protocollo* ***OSPF*** *può trattare anche informazioni raccolte da altri* ***AS****, queste vengono raccolte tramite* ***BGP*** *e vengono trattate come lo stato di un particolare* ***link****; possono essere etichettate con l'identificatore del router che le ha rese disponibili.*

*Il* ***AS*** *gestito dal protocollo* ***OSPF*** *è rappresentato da nodi che possono essere router, reti o singoli host:*

* *un* ***arco*** *tra due router indica l'esistenza di un data* ***link*** *tra i due*
* *un* ***arco*** *tra una rete ed un router indica la presenza di un'interfaccia di quel router su quella rete*
* *due* ***reti*** *possono essere collegate tramite un router*
* *una* ***rete*** *che ha archi entranti ed uscenti è una rete di transito (o uno stub).*

***Reti multicast o broadcast****: Una rete può essere abilitata ad eseguire un’operazione di broadcast o multicast; un esempio di rete di questo genere è ethernet. In una rete in cui è abilitato il broadcast viene eletto un router* ***designato detto “privilegiato” che rappresenta l'intera rete nel protocollo OSPF.***

***Reti non-multicast:*** *un esempio di rete di questo genere è un link PPP, in una rete di questo tipo può essere emulato il broadcast e in questo caso il protocollo OSPF si baserà sulla presenza di* ***un router designato che provvederà a diramare l'informazione nella rete.***

*In questo caso si parla di gestione* ***NMBA****.*

*Altrimenti la rete può essere gestita con la modalità Point-to-Multipoint, in questo caso tutti i collegamenti tra i router interni alla rete sono visti come link punto a punto.*

***Il grafo delle adiacenze per una rete broadcast***

******

***A/DA RT3 RT4 RT5 RT6 N2***

***RT3 X***

***RT4 X***

***RT5 X***

***RT6 X***

***N2 X X X X***

*I router eleggeranno il loro designato*

*Tutti i router sono accessibili dalla rete, e viceversa*

*****Sistema Autonomo***

*Ad ogni uscita da un router viene associata una metrica (costo) corrispondente al costo del link in uscita verso un altro router o verso una rete. Non vengono assegnati i costi in uscita da una rete.*

*Sono rappresentate alcune reti esterne (N12 ... N15) al sistema autonomo: i costi saranno calcolati utilizzando un protocollo di routing esterno. È rappresentato un nodo "stub" H1.*

***Cammino di un pacchetto***

*Il cammino di un pacchetto tra due nodi appartenenti a due aree che non siano il backbone viene quindi suddiviso in tre tronchi: il cammino intra-area dal nodo di partenza sino al router di confine*

*della sua area; il cammino entro il backbone tra l’area di partenza e l’area di destinazione; I il cammino intra-area nell’area di destinazione. I router di confine scambiano tra di loro informazioni sulla*

*raggiungibilità delle reti interne Utilizzando questi dati, insieme all’informazione sui link interni al backbone, ciascun router è in grado di calcolare il prossimo hop sulla route migliore per ciascuna delle reti interne alle diverse aree.*

*Il router di confine per uscire dall’area di partenza viene scelto in base alla sua distanza dall’area di destinazione. Altrimenti la rete può essere gestita con la modalità* ***Point-to-Multipoint****, in cui i collegamenti tra router interni alla rete sono visti come collegamenti punto a punto.*

***Cammino di un pacchetto:***

*il cammino di un pacchetto tra due nodi che non appartengono alla stessa area e che non siano il* ***backbone****, viene suddiviso in tre tronchi:*

1. *cammino intra-area dal nodo di partenza al router di confine*
2. *cammino entro il backbone tra l'area di partenza e quella di destinazione*
3. *cammino intra-area nell'area di destinazione*

*I router di confine scambiano tra loro informazioni riguardo la raggiungibilità delle reti interne*

*Usando questi dati ciascun router è in grado di calcolare l'albero dei cammini minimi per ciascuna delle reti interne alle diverse aree.*

*Il router di confine per uscire dall'area di partenza viene scelto in base alla sua distanza con l'area di destinazione.*

***Classificazione dei router:***

* *Router interni all'area: eseguono l'algoritmo di routing solo per determinare il cammino intra-area.*
* *Router di confine: eseguono l'algoritmo su ciascuna delle aree con cui sono collegati.*
* *Router di backbone: sono quelli che hanno un'interfaccia verso l'area di backbone, sono anche router di backbone o router che non hanno interfaccia con altre aree.*
* *Router di confine del AS: hanno il compito di diramare nell'AS le route esterne all'AS stesso, ognuno conosce le route per raggiungere i router di confine dell'AS.*

*Nella tabella, i nodi con archi in uscita (che intestano le colonne) sono:*

* *i router (R1-4) nell’area: il contenuto della colonna corrisponde al contenuto dei LSA (Link State Advertisement) emessi dal router.*
* *i router di confine del Sistema Autonomo (R5, R7): il contenuto della colonna corrisponde al contenuto dei LSA (Link State Advertisement) emessi dal router.*

*le reti interne (N3), con costo uscente nullo.*

*Il contenuto dei LSA emessi dai router interni all’area, R1 e R2, è limitata ai router di confine dell’area*

*I nodi di confine dell’area, R3 e R4, rendono disponibili come LSA i costi dei percorsi verso gli altri nodi del Sistema Autonomo, calcolati internamente al backbone*

*I nodi di confine del Sistema Autonomo, R5 e R7, rendono disponibili come LSA i costi dei percorsi verso altri nodi esterni al AS, calcolati con BGP.*

*L’area 3 è stata raggruppata sotto un’unica route.*

Network R3 LSA R4 LSA

N1 4 4

N2 4 4

N3 1 1

N4 2 3

*Il LSA inviata dai router di confine dell’area 1, R3 e R4, verso gli altri router del backbone*

* *contiene le metriche relative alle reti interne all’area come se fossero link collegati ai router di confine*

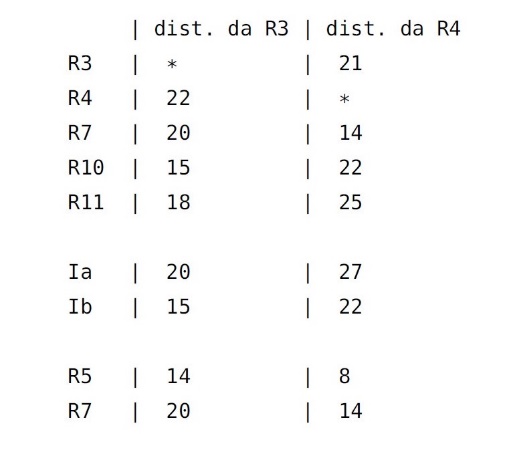
*I costi delle route sono relativi ai routers, e sono calcolati applicando l’algoritmo SPF sul database di area visto prima*

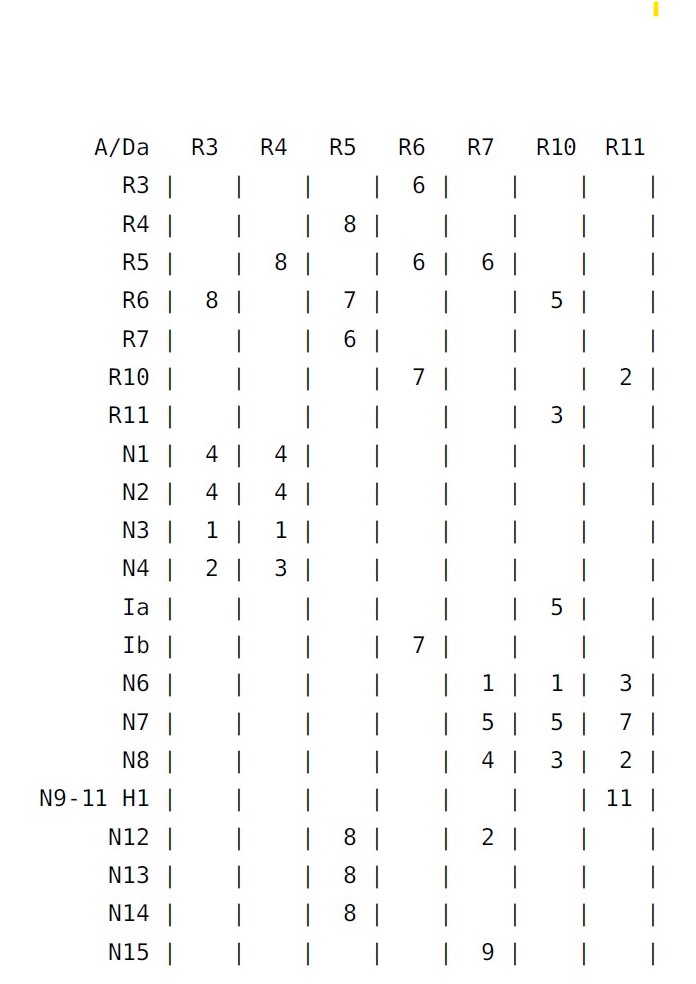
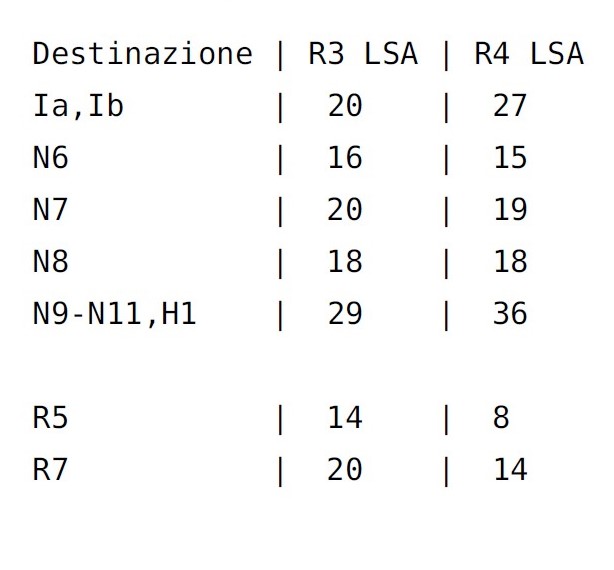
*Ad esempio, il costo da R4 a N4 (3) è ottenuto come:*

*1 (R4 ! N3)*

*0 (N3! R3)*

*2 (R3! N4)*

**



*Ogni router contribuisce con il contenuto di una colonna:*

* *La raggiungibità delle reti entro la propria area*
* *Lo stato dei link con i router adiacenti*
* *La raggiungibilità delle reti esterne all’AS*

*Sulla base della tabella precedente ogni router (qui, quelli del backbone) elabora la propria per ottenere la tabella di routing nel backbone.*

*Quindi calcola i cammini verso le reti del Sistema Autonomo. Si utilizzano per ciò le informazioni ricevute dagli altri routers del backbone.*

*Queste informazioni vengono distribuite entro la propria area, ed andranno a costituire il database dell’area, già visto in precedenza.*

***Link State Advertisement****:*

*Nei* [*protocolli*](http://www.dacrema.com/Informatica/Protocolli_rete.htm) *di tipo* [*Link state*](http://www.dacrema.com/Informatica/Protocollo_Link_state.htm) *i pacchetti inviati da un* [*router*](http://www.dacrema.com/Informatica/Router.htm) *e che, ricevuti dai vari router della rete, permettono la costruzione della mappa della rete, sono detti Link State Packet (****LSP****): pacchetto dello stato dei collegamenti. Ogni LSP può contenere più Link State Advertisement (****LSA****), messaggio sullo stato dei collegamenti, che diffondono informazioni sullo stato dei link come i tradizionali LSP ma non sono pacchetti. Gli LSA vengono generati da ciascun router dell'*[*AS*](http://www.dacrema.com/Informatica/Autonomous_System.htm) *a seconda delle competenze ad esso assegnat. Ci sono cinque tipi di LSA:*

1. [***Router LSA***](http://www.dacrema.com/Informatica/ROUTER_LSA.htm)*: informazioni sui router adiacenti a chi genera il LSA*
2. ***Network LSA****: generato dai designated router per descrivere la* [*LAN*](http://www.dacrema.com/Informatica/Reti_Locali.htm) *cui il designated è collegato*
3. ***Summary LSA****: genrati dagli area border router per comunicare la raggiungibilità delle altre aree ai router interni*
4. ***AS boundary router LSA****: generati dagli area border router per annunciare la raggiungibilità di un AS boundary router*
5. ***AS external LSA****: generati da un AS boundary router per annunciare destinazioni esterne al dominio di routing.*

*Inviata dai router di confine, contiene la raggiungibilità delle reti interne ad una certa area. I costi delle route sono relativi ai router e sono calcolati da OSPF internamente all'area.*

***Link state nei router di backbone****: ogni router conosce la raggiungibilità delle reti interne alla propria area, lo stato dei link con i router adiacenti, la raggiungibilità delle reti esterne all'AS. Ogni router elabora il proprio albero per ottenere la tabella di routing nel backbone, quindi calcola i cammini verso le reti del AS, usando le informazioni ricevute dagli altri router di backbone per costruire un database relativo all'area.*

***I Link state advertisement*** *possono quindi essere suddivisi in 5 tipi:*

1. ***Router LSA****: generato da* ***tutti i router*** *e diffuso nella propria area, indica lo stato delle* ***interfacce dei router.***
2. ***Network LSA****: generato dai* ***router designati****, indica lo stato delle* ***interfacce dei router collegati all'area****.*
3. ***Summary LSA****: generato dai* ***router di confine****, indica* ***lo stato di una route fuori dall'area verso una rete.***
4. ***Summary LSA:*** *generato dai* ***router di confine*** *indica* ***lo stato di una route fuori dall'area verso un router di confine dell'AS.***
5. ***AS-External LSA:*** *generato* ***dai router di confine*** *del AS e diffuso entro l'AS.*
6. ***Inizializzazione****, e test funzionale dello strato Data Link.*
7. ***Acquisizione dei vicini****: per questo viene utilizzato il protocollo*
8. ***Hello****, che serve anche all’elezione del Designato nelle reti broadcast e NMBA.*
9. ***Acquisizione delle adiacenze****: non necessariamente un vicino è adiacente.*

*L’adiacenza può essere negoziata, e necessita la sincronizzazione*

*delle strutture dati.*

*In una rete broadcast e NBMA l’adiacenza viene controllata dal*

*Designato.*

*5****. Flooding*** *dello stato dei link del router, periodico, e quando si*

*modifica. La diffusione resta confinata entro l’area. I router di confine eseguono il flooding separatamente nella propria area e nel backbone.*

***Adiacenze****:*

*Una caratteristica della relazione di adiacenza è la simmetria: una relazione di adiacenza viene stabilita quando un nodo determina che un vicino lo vede a sua volta come tale.*

***Il protocollo Hello****:* ***Ha il compito di stabilire e mantenere la relazione di adiacenza tra vicini*** *(non necessariamente un vicino è un adiacente). Un compito specifico del protocollo Hello è l'elezione del router* ***designato*** *in una rete broadcast o NBMA.*

*In una rete broadcast o NBMA, ogni router periodicamente esegue un broadcast di un messaggio di Hello sulla rete: così vengono rilevati dinamicamente nuovi router. Nello stato* ***down*** *non viene garantitala consistenza dei dati con il vicino. Nello stato* ***attempt*** *solo nelle reti broadcast e NBMA viene attivamente cercata la sincronizzazione, spedendo in broadcast pacchetti Hello con la cadenza richiesta.*

*La ricezione di un pacchetto Hello porta nello stato* ***init*** *per un periodo corrispondente alla cadenza richiesta per i messaggi di Hello. I router in questo stato vengono indicati nei pacchetti Hello successivi. La ricezione dal vicino di un pacchetto Hello che indica questo nodo porta nello stato* ***2-way****. In questo stato viene deciso il* ***Router Designato****. Lo stato* ***ExStart*** *indica che è stata stabilita l’adiacenza.*

***Instaurazione dell'adiacenza****: una volta che due router si sono riconosciuti uno nel pacchetto Hello dell'altro, inizia il processo di creazione dell'adiacenza; il primo passo consiste nello stabilire chi tra i due database sarà* ***master*** *(router con indirizzo superiore) e chi* ***slave****, dopodiché viene operata la sincronizzazione delle basi di dati (Database Descrption Packets) che descrivono il DB.*

*Il* ***master*** *poi spedisce pacchetti allo* ***slave*** *che li riscontra con un numero d'ordine e non ricevendo il riscontro il* ***master*** *rispedisce periodicamente il pacchetto non riscontrato. Solo il master può rispedire pacchetti.*

*Sia il pacchetto spedito che il riscontro contengono la descrizione di alcuni link che vengono inseriti nel DB, quasta fase di exchange termina quando sia master che slave hanno resettato* ***il flag moore*** *nel loro pacchetto. Al termine di queste operazioni i due router sono adiacenti e le loro BD complete.*

***Formato LSA:***

* ***LS-Sequence Number****: serve a determinare tra due LSA relativi allo stesso link quale sia il più recente. Si tratta di un numero che viene incrementato ogni volta che l'LSA viene aggiornato.*
* ***Checksum****: dell'intero LSA, escluso l'Age.*
* ***Lunghezza****: dell’intero LSA, in ottetti.*
* *informazioni varianti a seconda del tipo di LSA.*

***Flooding***

*Per la comunicazione tra router viene utilizzato l'algoritmo di* ***flooding****.*

*E’ un* [*protocollo di instradamento*](https://it.wikipedia.org/wiki/Protocolli_di_routing) *usato* dai [router](https://it.wikipedia.org/wiki/Router) che inoltrano un [pacchetto](https://it.wikipedia.org/wiki/Pacchetto_(reti)) in ingresso su tutte le *linee ad eccezione di quella da cui proviene. Ogni pacchetto in arrivo viene inoltrato su ogni linea di uscita eccetto quella da cui è arrivato.* ***Questo algoritmo genera un vasto numero di pacchetti duplicati;*** *in effetti,* ***un numero infinito****, a meno di non prendere qualche misura per fermare il processo.*

*Per evitare l'invio infinito di pacchetti duplicati si possono utilizzare due accorgimenti:*

* ***contatore di salto****: si inserisce nel pacchetto un contatore da decrementare ad ogni nuovo router attraversato. Idealmente il valore di tale contatore deve essere uguale al* [*percorso minimo*](https://it.wikipedia.org/wiki/Algoritmo_di_Dijkstra) *fra sorgente e destinazione ma non conoscendo la topologia della rete si può assegnare un valore uguale al diametro della rete.*
* ***numero di sequenza****: ogni router deve conoscere la presenza degli altri router e per ogni router dovrà solo controllare che il pacchetto proveniente da quello abbia un numero sequenza maggiore del precedente. Per evitare la crescita all'infinito si adotta una soglia k che riassume la ricezione di tutte le sequenze fino ad appunto k. Raggiunta la soglia il numero si azzera.*

***Il flooding può tornare utile*** *in applicazioni militari* ***dove non è tollerato il malfunzionamento di un router durante il percorso di inoltro.***

*Ha utilizzi pratici anche nelle reti wireless dove tutti i messaggi trasmessi da una stazione possono essere ricevuti da tutte le altre che si trovano nel suo campo.*

*Può essere utilizzato, infine anche come metrica di confronto per altri algoritmi di routing: il flooding segue sempre il percorso più breve poiché utilizza ogni possibile percorso in parallelo, ne segue che nessun altro protocollo riesce ad ottenere ritardi inferiori.*

*È un protocollo di routing molto semplice da implementare e con limitate necessità di elaborazione da parte dei router; il problema associato è dovuto al fatto che esso va ad utilizzare la capacità trasmissiva della rete in maniera ridondante utilizzando un numero di link che cresce allontanandoci dal router iniziale e aumentando quindi le probabilità di congestione di rete.*

**L’algoritmo di flooding** viene utilizzato per mantenere aggiornati i database dei routers del sistema autonomo. Si tratta di una operazione che conduce ad un broadcast, in cui ciascuno dei router si occupa di diffondere l’informazione solo ai propri vicini. L’informazione quindi finisce per raggiungere tutti i router. La procedura di flooding inizia alla ricezione di un pacchetto di Link State Update, che contiene uno o più LSA. Per ciascun LSA viene applicato l’algoritmo di flooding, che condurrà alla creazione di pacchetti di riscontro (Link State Acknowledgement), e di altri pacchetti di Link State Update.

Viene prima considerata la correttezza dell’LSA (utilizzando la checksum e test di ragionevolezza), e se questo debba essere accettato. Se dell’LSA appena ricevuto non esiste localmente una istanza, o se questa esiste ma è meno recente:

1. se il vecchio LSA è arrivato da meno di 1 sec. il nuovo viene

scartato: serve a rallentare valanghe;

1. altrimenti il nuovo LSA viene predisposto per essere rinviato su alcune delle interfacce del router, contribuendo al flooding, e il vecchio LSA viene rimosso da eventuali liste di ritrasmissione (innescata dal mancato acknowledgement); il nuovo LSA entra nel database del router, e viene schedulato il ricalcolo SPF viene predisposto l’invio di un acknowledgement per l’LSA;

*Se l’LSA appena ricevuto è già presente nel database, viene*

*predisposto l’acknowledgement; Se l’LSA appena ricevuto è meno recente di quello già presente*

*nel database, questo viene scartato senza predisporne l’acknowledgement. Invece viene predisposta la spedizione di un Link State Update contenente la copia locale (più recente) al*

*vicino da cui è stato spedito l’LSA.*

**FQDN**

L’indirizzo numerico IP è molto efficace per gestire il routing, ma è troppo legato all’infrastruttura a livello 2. Per rappresentare un Internet composto da servizi, più che da nodi, serve fare riferimento a entità amministrative, o a servizi. La rappresentazione **Fully Qualified Domain Name associa ad un indirizzo IP un identificatore** (ad es. www.google.com) che rappresenta una gerarchia amministrativa ed un servizio. Apre alcune possibilità interessanti:

* lo stesso IP può essere associato a servizi diversi
* la dislocazione di un nodo che ospita il servizio non interessa il FQDN

Il FQDN può essere associato a IP diversi in modo dinamico (bilanciamento del carico). Tuttavia si apre il problema dell’associazione (dinamica) di unFQDN ad un indirizzo IP.

*Il servizio* ***Domain Name System*** *(DNS)* ***serve a collegare un indirizzo numerico IP con un Fully Qualified Domain Name(FQDN), per brevità dominio****. In astratto la soluzione è semplice: è sufficiente costruire una tabella di corrispondenza tra* ***domini e indirizzi IP****. In concreto la quantità di indirizzi e la necessità di risolvere rapidamente la corrispondenza pone grossi problemi. La gestione dell’intera tabella viene distribuita su un gran numero di server DNS. Ogni server è* ***autorevole*** *(authoritative) per un certo numero di indirizzi IP, le cui corrispondenze sono configurate dall’amministratore del server.*

*Altri server intermedi registrano in una cache temporanea le corrispondenze che ottengono da altri server DNS.*

*Ogni nodo Internet viene configurato (manualmente o tramite DHCP) con l’indirizzo IP di un server DNS a cui richiedere la risoluzione dei FQDN. Ogni nodo ha un programma client (il resolver) per il protocollo DNS*

*Il DNS funziona così:*

* *l’applicazione (browser, ping ...) deve risolvere un FQDN (ad es.*

*"www.google.com)*

* *l’applicazione richiede al resolver l’indirizzo IP corrispondente ad*

*un certo dominio*

* *il resolver invia una richiesta di risoluzione al server DNS che*

*conosce*

* *il server tratta la richiesta, eventualmente come client di altri server,*

*e restituisce l’indirizzo IP al client sul nostro host*

* *quest’ultimo consegna l’indirizzo IP alla nostra applicazione, che la*

*usa per svolgere la sua funzione*

***Gli indirizzi FQDN sono organizzati secondo una gerarchia di domini****, introdotta anche per evitare duplicazioni (gli FQDN devono essere unici).*

*La gerarchia è espressa con la notazione puntata che conosciamo:*

* *per ultimo il dominio più ampio (it o com) anche chiamato Top Level*

*Domain (TLD),*

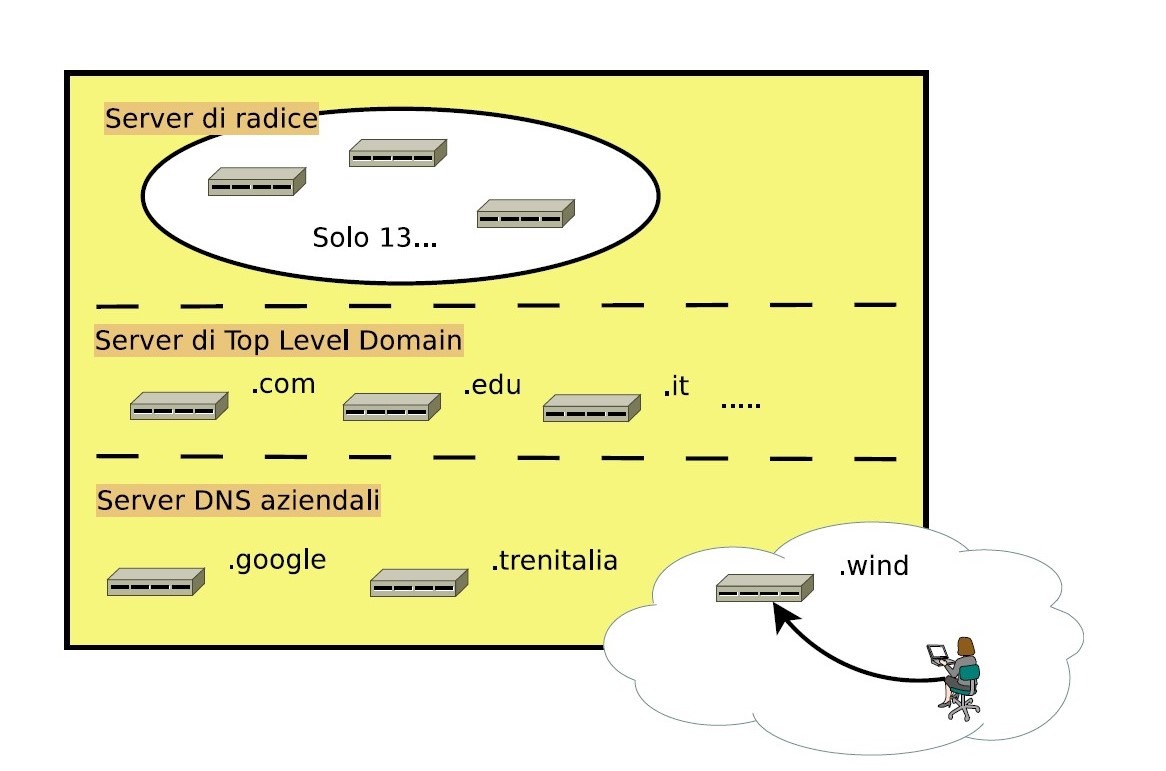
* *poi un dominio più ristretto (aziendale),*
* *e per finire un indicatore dell’host o di un servizio specifico (www)*

*Gli elementi possono essere anche più di tre. L’ordine è dalla foglia verso la radice.*

*La gerarchia dei server autorevoli ricalca quella del FQDN:*

1. *server root (ce ne sono una dozzina di repliche, con contenuto sincronizzato),*
2. *server distinti per i TLD*
3. *server aziendali*

*Il server mantiene un database rappresentato con Resource Records(RR) che rappresentano la corrispondenza tra un dominio, ed un indirizzo IP. Nel server aziendale è l’amministratore a inserire queste registrazioni. Il server viene definito autorevole (authoritative) per quelle corrispondenze.*

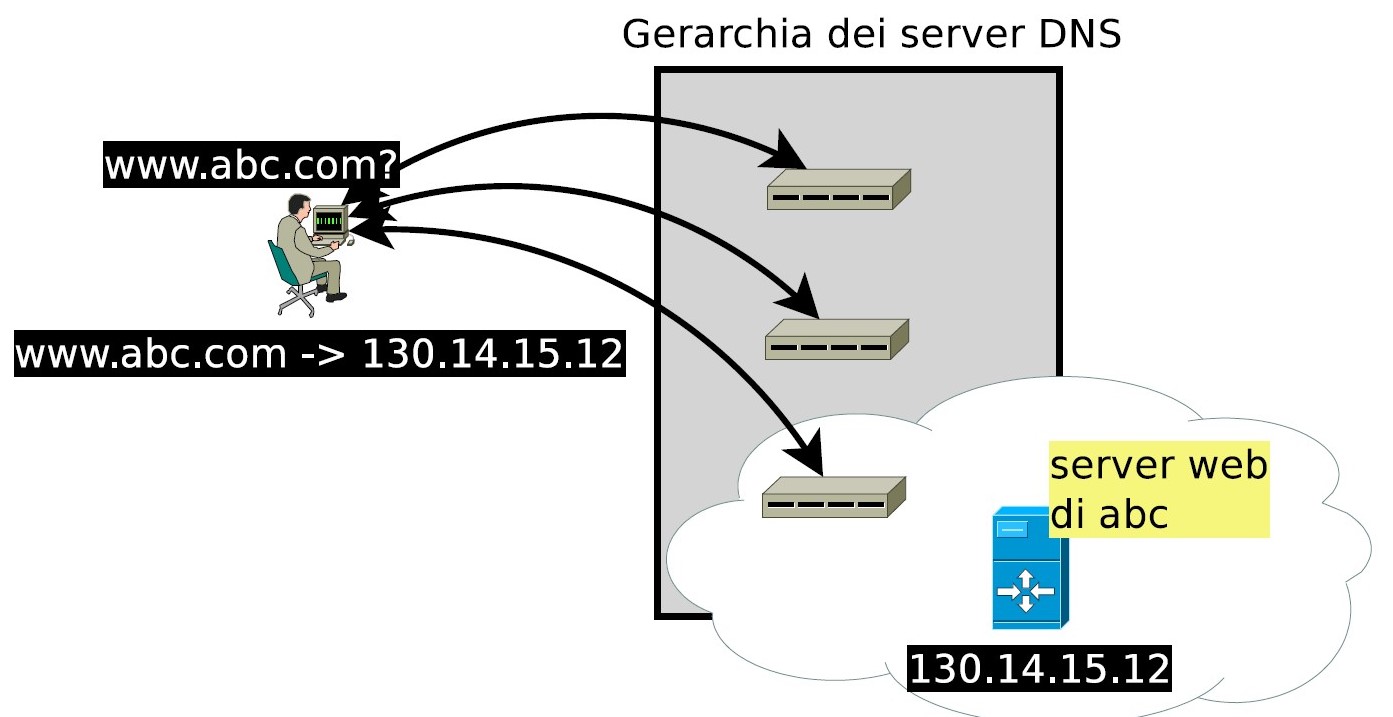


*Ciascun server è authoritative per i nodi con un certo prefisso. L’utente riceve (anche con DHCP) l’indirizzo di un server DNS primario e uno secondario (ricambio).*

***Non Ricorsiva***

Volendo utilizzare solo server autorevoli, una richiesta dovrebbe seguire il seguente schema:

1. richiedo ad uno dei server root (noti a tutti) l’indicazione del server autorevole per il TLD
2. richiedo al server autorevole per il TLD l’indicazione di un server autorevole per il dominio aziendale
3. eccetera, scendendo nella gerarchia dei server.



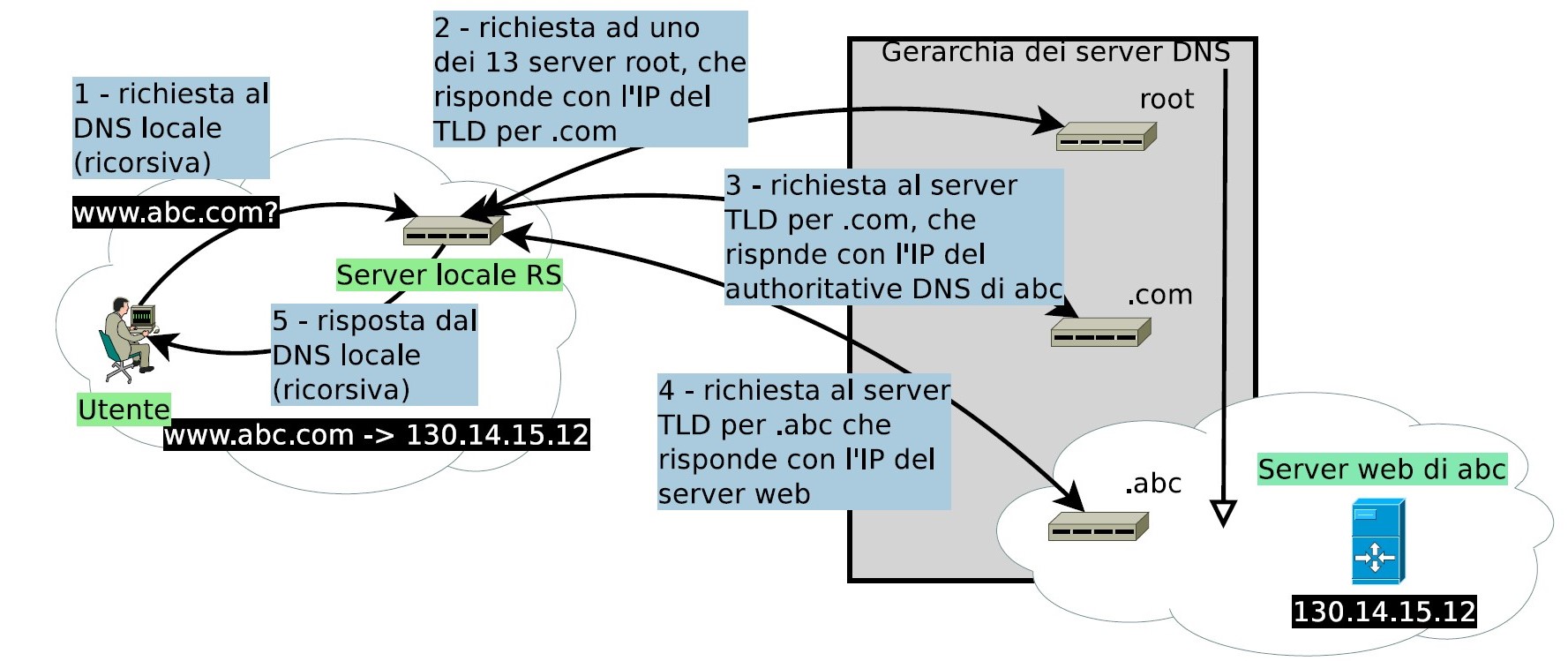
*Tuttavia il traffico che si verrebbe a creare sui server di root sarebbe enorme. Quindi viene introdotta una modalità* ***ricorsiva*** *per la risoluzione.*

*Il server configurato come ricorsivo:*

* *mantiene una cache degli indirizzi dei server authoritative, diventando a sua volta server non authoritative*
* *esegue i passi di ricerca visti sopra al posto del cliente.*

*Di solito i server DNS primari vengono configurati come ricorsivi, così da fare da filtro agli utenti locali. Questo genere di server sono poi attrezzati per gestire in modo ottimale una cache di grandi dimensioni.*

*L’utente non conosce i server di root, ma solo un server primario. L’utente chiede la risoluzione degli indirizzi al proprio server primario (RS in figura). Se questo non è in grado di risolvere l’indirizzo, consulta il server root, che gli comunica il server authoritative per il TLD.*



*Questo viene registrato nella cache di RS. Quindi RS interpella il server authoritative del TLD e via dicendo...In breve tempo il server locale avrà nella cache gli IP dei server autoritative dei TLD, e non dovrà più richiederli ai server di root.*

*La creazione di un FQDN avviene attraverso una operazione amministrativa complessa (più o meno costosa)*

*(*[*http://freedns.afraid.org/*](http://freedns.afraid.org/)*). Il servizio Internet che opera la registrazione del dominio è detto registrar (un termine usato anche in altri contesti). Il servizio offerto dal registrar consiste in: verificare che il dominio non sia duplicato e inserire un record con l’associazione dominio IP nel proprio server DNS (Resource Record, RR).*

*La configurazione iniziale di un server si ottiene inserendo un certo numero di Resource Record in un file, che viene letto dal server all’accensione*

*Così si determinano i domini per cui il server è authoritative.*

***DNS***

*Il* ***DNS*** *si basa sullo scambio di messaggi incapsulati in datagrammi UDP. La perdita di pacchetti non è un problema: il RR perduto può essere richiesto nuovamente dopo un timeout. La comunicazione si esaurisce con la consegna dell’RR: non serve un controllo di flusso. Gli RR possono essere ricevuti in ordine diverso da quello di spedizione.*

*L’integrità degli RR è importante: non esiste una checksum di un RR, ma ce ne è una per il datagramma.*

***Un messaggio DNS si compone di:***

* *Uno header di lunghezza fissa:*

*identifica la sorgente, specifica se si tratta di una interrogazione o di una risposta, specifica l’uso della ricorsione, quantifica il numero di sezioni dei vari tipi inserite nel pacchetto.*

* *Un numero variabile di sezioni, appartenenti a quattro categorie*

*nell’ordine in cui appaiono):*

1. *Query: contengono il nome del dominio di cui si vuole conoscere L’IP*
2. *Risposte: queste sono RR complete in risposta alla domanda*
3. *Risposte authoritative: come sopra, per cui il mittente è authoritative*
4. *Altre RR: possono essere aggiunte per arricchire la cache del ricevente*

*Il dato che rappresenta la corrispondenza le informazioni utili al DNS è detto Resource Record (RR). I campi principali di un RR sono i seguenti:*

* *NAME il FQDN*
* *TYPE: il tipo A indica una associazione (FQDN, IP), tipo AAAA indica una associazione (FQDN, IPv6), il tipo NS indica l’associazione (Name Server, zona) ecc.*
* *TTL il periodo di validità del dato quando presente in una cache*
* *RDATA il dato associato a NAME: può essere l’indirizzo IP*

*Esiste una funzionalità di Reverse-DNS: il server è in grado di fornire l’FQDN conoscendo l’IP*

*Queste informazioni vengono registrate in RR specifici.*

* *L’indirizzo ricercato (NAME) corrisponde all’IP (numerico) seguito*

*dal dominio in-addr.arpa.*

* *I domini (RDATA) corrispondono ai quattro numeri contenuti nell’IP.*

***TCP***

*Consente di trasferire tra due applicazioni una quantità di dati superiore ai 65KByte che caratterizzano il datagramma UDP.*

***La comunicazione avviene in ambedue le direzioni****. Durante il trasferimento le due applicazioni restano collegate tra*

*loro. Si dice che tra le due applicazioni esiste una connessione. Come per UDP, vengono introdotte le porte per ottenere multiplexing IP. Quindi un certo host può partecipare a più connessioni sulla stessa interfaccia.*

*I dati vengono frammentati e ciascun frammento (un segmento) è incapsulato in un distinto pacchetto IP.*

*Per assolvere alla funzione di trasferire una sequenza di ottetti in modo affidabile il protocollo TCP deve:*

*trasferire tutti gli ottetti, nonostante la possibile perdita di segmenti*

* *ricostruire l’ordine originale degli ottetti*
* *eliminare eventuali ottetti duplicati*
* *Per ottenere questi risultati TCP introduce alcuni pacchetti IP aggiuntivi, che servono per l’apertura e per la chiusura della connessione.*

*Altre informazioni di coordinazione vengono introdotte nello header TCP.*

*Tuttavia, poiché l’effetto della connessione persiste durante il trasferimento, il costo può essere ammortizzato.*

*L’operazione di apertura della connessione è asimmetrica: una delle due applicazioni ha la funzione di attendere una richiesta di connessione (server) l’altra applicazione ha la funzione di richiedere la connessione*

*(client).*

*Quindi la connessione è limitata a due applicazioni, è impossibile il broadcast e il multicast.*

*Una volta stabilita la connessione, la comunicazione diventa simmetrica: ambedue le applicazioni possono spedire e ricevere. Dunque, dal punto di vista delle due applicazioni, TCP mette a disposizione un canale di comunicazione bidirezionale affidabile.*

*TCP deve realizzare tre proprietà fondamentali:*

***completezza****: il destinatario deve ricevere tutto ciò che il mittente ha spedito*

***sequenzializzazione****: gli ottetti devono essere ricevuti nello stesso ordine in cui sono stati ricevuti*

***unicità****: ogni ottetto deve essere ricevuto una sola volta*

*Per questo il protocollo di scambio identifica la posizione di ogni singolo ottetto (byte) in ciascuna direzione*

*Questa informazione è detta* ***sequence number*** *(abbreviato SN):*

1. *i byte replicati vengono quindi riconosciuti (stesso SN)*
2. *i byte mancanti vengono richiesti nuovamente al mittente (SN mancanti)*
3. *i byte vengono ordinati sulla base del loro numero d’ordine (SN crescente)*

*Il* ***Sequence Number*** *di un ottetto viene indicato specificando, per ogni pacchetto IP, il SN del primo ottetto contenuto*

*Nelle due direzioni di comunicazione vengono utilizzati due sequenze indipendenti. Quindi il programma che riceve può riconoscere la duplicazione, la perdita e determinare la posizione degli ottetti ricevuti, Il mittente deve poter rilevare la perdita di segmenti.*

*Per questo il destinatario indica, nei pacchetti che fluiscono in direzione opposta, l’indice del primo byte non ancora ricevuto (Acknowledge, per brevità ACK).*

*Questi pacchetti possono anche contenere informazioni utili, oppure essere destinati solo al trasferimento dell’ACK.*

*In questo modo il mittente può capire se i byte che ha spedito sono stati tutti ricevuti, sono in parte ancora in transito, o non sono stati ricevuti*

*se ACK è successivo all’ultimo Byte spedito, tutto a posto*

*se ACK è precedente all’ultimo Byte spedito, allora alcuni byte potrebbero essere ancora in transito o perduti*

*In quest’ultimo caso il mittente ha due azioni da mettere in programma:*

1. *arrestare la trasmissione per non sovraccaricare la linea*
2. *rispedire i dati perduti*

*Se il mittente riceve più ACK con lo stesso SN c’è il sospetto che un segmento sia andato perduto e vada quindi rispedito, il destinatario non riesce a riempire un ’buco’.*

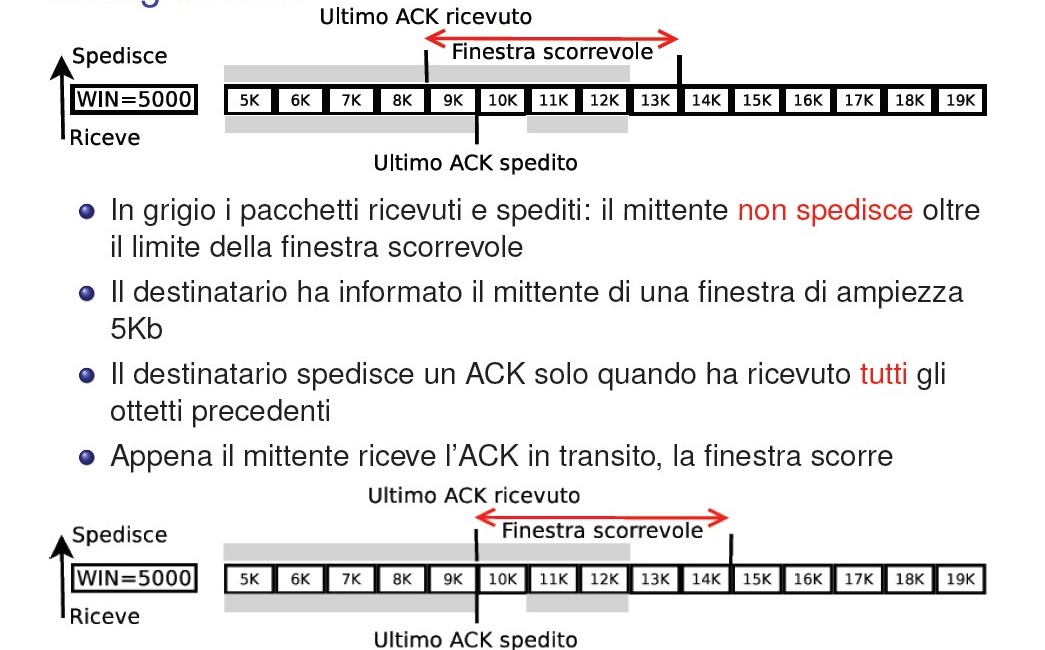
*Per decidere se continuare la trasmissione si utilizza* ***la finestra di ricezione*** *(WIN).*

*La decisione è determinata da un dato comunicato dal destinatario, che indica una finestra di ricezione (WIN)*

*Il mittente non spedisce byte che abbiano un SN superiore all’ultimo ACK più la finestra WIN*

*Si parla di* ***sliding window*** *perché man mano che vengono ricevuti ACK, la finestra si sposta progressivamente in avanti. La dimensione della finestra può variare nel tempo.*

***Sliding window***



***Header***

*I tre dati necessari a realizzare sequenzializzazione e unicità sono tutti inclusi nello header del segmento TCP*

***SYN (32 bit) SN del primo ottetto inviato del segment***

***ACK (32 bit) SN del primo ottetto atteso***

***WIN (16 bit) La finestra di ricezione***

*Per garantire anche la completezza serve che il mittente determini quando i dati sono andati probabilmente perduti*

*La segnalazione da parte del destinatario non è esplicita:*

*l’assenza di ACK può (non necessariamente) essere causata dalla perdita di dati. Il mittente dunque imposta un timeout sui dati che non sono stati ricevuti.*

*Allo scadere del timeout i dati successivi all’ultimo ACK vengono rispediti. Gli elementi noti di una connessione sono l’IP e la porta su cui il server riceve (ed è in attesa).*

*Gli altri elementi sono incogniti:*

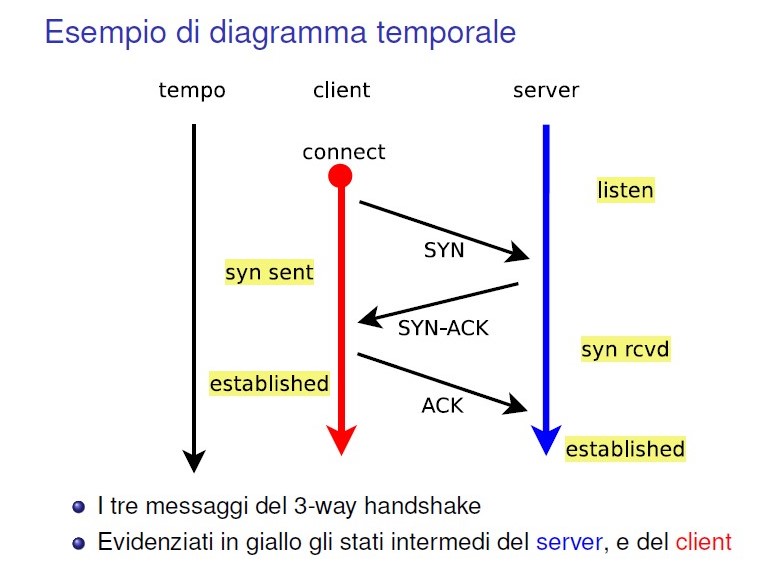
* *porta ed IP del cliente*
* *SN iniziale di client e server (che vengono scelti a caso)*

*Per stabilire questi elementi si utilizza una transazione a tre passi, detta three-way handshake*

***Apertura connessione***

***Three-way Handshake***

*La connessione viene iniziata dal client. Il server è già attivo, ed è in attesa di richieste di connessione.*



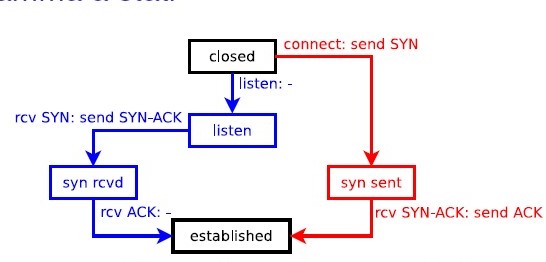
***Le tre fasi del 3-way handshake:***

*1 il cliente invia il primo pacchetto, indicando quindi il proprio IP e la porta (scelta casualmente), nonché il proprio numero di sequenza iniziale (SYN);*

*2 il server invia il proprio pacchetto, contenente il proprio numero di sequenza iniziale, e conferma al cliente i dati ricevuti (SYN-ACK);*

*3 il cliente restituisce un pacchetto di conferma (ACK); Il protocollo non viene influenzato da eventuali perdite, duplicazioni, o inversione di pacchetti, sono eventi che vengono rilevati e corretti.*

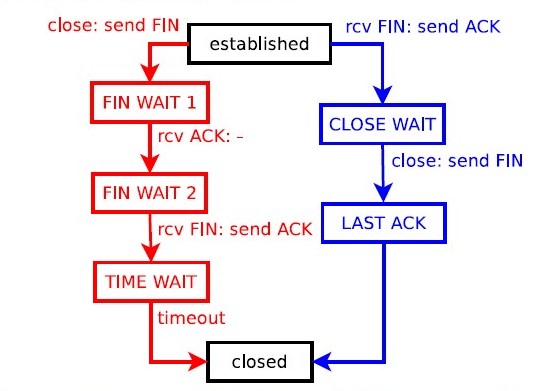
***Diagramma a Stati***

**

*Sono rappresentate solo le transizioni eseguite in*

*situazioni normali. Stati, eventi e transizioni in* ***blu*** *sono tipiche del* ***server****, in* ***rosso*** *per il* ***client****. Il protocollo standard introduce anche transizioni anomale, non rappresentate nella figura:*

* *la connessione viene chiusa prematuramente*
* *il server invia il SYN*
* *server e client inviano il SYN insieme*



*Transizioni tipiche per il* ***server, e il client*** *(generalmente iniziata dal cliente)*

*Gestita in due fasi, ciascuna* ***FIN*** *e poi* ***ACK***

*Complicata dalla necessità di dover aspettare pacchetti in sospeso.*

***SACK***

*La tecnica di ACK cumulativo che fa parte di TCP è inefficiente, e tratta male la perdita di singoli segmenti. La caratteristica di richiedere minimo controllo era importante agli albori di Internet, ma non lo è più.*

*L’inefficienza sta nel fatto che, in caso di perdita di un singolo frammento nel flusso (v. figura precedente): l’ACK non può procedere oltre il SN corrispondente al primo byte del segmento perduto e i segmenti successivi a quello perduto rischiano di essere (in parte) ritrasmessi.*

*Per ovviare a questo inconveniente si usa la parte opzionale dello header. Possono essere inserite alcune coppie di SN, ad indicate blocchi di dati contigui ricevuti dopo il SN inviato come ACK. La funzione Selective Acknowledge (SACK) è opzionale, ma spesso presente.*

*La procedura utilizzata per instaurare in modo affidabile una* [*connessione*](https://it.wikipedia.org/wiki/Connessione_(informatica)) *TCP tra due host è chiamata three-way handshake (stretta di mano in 3 passaggi), indicando la necessità di scambiare 3 messaggi tra host mittente e host ricevente affinché la connessione sia instaurata correttamente. Consideriamo ad esempio che l'host A intenda aprire una connessione TCP con l'host B; i passi da seguire quindi sono:*

1. ***A invia un segmento SYN a B*** *- il flag SYN è impostato a 1 e il campo Sequence number contiene il valore x che specifica l'Initial Sequence Number di A;*
2. ***B invia un segmento SYN/ACK ad A*** *- i flag SYN e* [*ACK*](https://it.wikipedia.org/wiki/ACK_(informatica)) *sono impostati a 1, il campo Sequence number contiene il valore y che specifica l'Initial Sequence Number di B e il campo Acknowledgment number contiene il valore x+1 confermando la ricezione del ISN di A;*
3. ***A invia un segmento ACK a B*** *- il flag ACK è impostato a 1 e il campo Acknowledgment number contiene il valore y+1 confermando la ricezione del ISN di B.*

*Il terzo segmento non sarebbe, idealmente, necessario per l'apertura della connessione in quanto già dopo la ricezione da parte di A del secondo segmento, entrambi gli host hanno espresso la loro disponibilità all'apertura della connessione. Tuttavia esso risulta necessario al fine di permettere anche all'host B una stima del timeout iniziale, come tempo intercorso tra l'invio di un segmento e la ricezione del corrispondente ACK.*

*Il flag SYN risulta utile nell'implementazione pratica del protocollo, e nella sua analisi da parte dei* [*firewall*](https://it.wikipedia.org/wiki/Firewall)*: nel traffico TCP i segmenti SYN stabiliscono nuove connessioni, mentre quelli con il flag non attivo appartengono a connessioni già instaurate.*

*I segmenti utilizzati durante l'handshake sono solitamente 'solo header', ossia hanno il campo Data vuoto essendo questa una fase di sincronizzazione tra i due host e non di scambio di dati.*

*Il three-way handshake si rende necessario poiché la sequenza numerica (ISN) non è legata ad un* [*clock*](https://it.wikipedia.org/wiki/Clock) *generale della rete, inoltre ogni pacchetto IP può avere il proprio modo di calcolare l'Initial Sequence Number. Alla ricezione del primo SYN non è possibile sapere se la sequenza ricevuta appartenga ad un ritardo dovuto ad una precedente connessione. Tuttavia, viene memorizzata l'ultima sequenza usata nella connessione, potendo così essere richiesta la verifica all'host mittente del SYN appartenente alla vecchia connessione.*

***Chiusura connessione***

*Dopo che è stata stabilita, una connessione TCP non è considerata una singola connessione bidirezionale, ma piuttosto come l'interazione di due connessioni monodirezionali; pertanto, ognuna delle parti dovrebbe terminare la sua connessione. Possono esistere anche connessioni chiuse a metà, in cui solo uno dei due host ha chiuso la connessione poiché non ha più nulla da trasmettere, ma può (e deve) ancora ricevere i dati dall'altro host.*

*La chiusura della connessione si può effettuare in due modi: con un handshake a tre vie, in cui le due parti chiudono contemporaneamente le rispettive connessioni, o con uno a quattro vie (o meglio 2 separati handshake), in cui le due connessioni vengono chiuse in tempi diversi.*

*L'handshake a 3 vie per la chiusura della connessione è omologo a quello usato per l'apertura della connessione, con la differenza che il flag utilizzato è il FIN invece del SYN. Un host invia un segmento con la richiesta FIN, l'altro risponde con un FIN + ACK, infine il primo manda l'ultimo ACK, e l'intera connessione viene terminata.*

*Il doppio handshake a 2 vie invece viene utilizzato quando la disconnessione non è contemporanea tra i due host in comunicazione. In questo caso uno dei due host invia la richiesta di FIN, e attende l'ACK di risposta; l'altro terminale farà poi altrettanto, generando quindi un totale di 4 segmenti.*

***Header TCP***

*Questo header viene aggiunto in testa a tutti i segmenti*

*Tra i flag troviamo FIN, SYN, ACK*

*Il Sequence Number si riferisce al primo ottetto del segmento (o al sequence number iniziale in fase di SYN).*

***La checksum comprende:***

* *lo header*
* *il payload*
* *alcune informazioni sensibili dello header IP: gli indirizzi mittente e destinatario, protocollo e lunghezza*

***Controllo congestione TCP***

*Il protocollo* ***TCP*** *è esposto a sovraccaricare la rete, perché trasmette un flusso continuo di dati. Deve esistere un sistema di controllo per fare in modo che i dati entrino con lo stesso ritmo con cui escono (equilibrio)*

*Il sistema previsto nello standard TCP è la finestra di ricezione. In breve, il mittente non spedisce il byte i-esimo dello stream se prima non ha ricevuto l’ack del byte (i-w) -esimo, dove w è l’ampiezza della finestra di ricezione (WIN)*

*A regime, per ogni byte che esce ne può entrare un altro (controllo di flusso).*

*L’ampiezza della finestra di ricezione è variabile, e viene inizializzata durante il 3-way handshake.*

*Il metodo riesce a tenere conto delle dinamiche lente interne del nodo ricevente (riduzione delle prestazioni)*

*Non riesce a controllare il transitorio che si presenta all’inizio di una connessione.*

* *il mittente può spedire sino a raggiungere l’ampiezza della finestra senza attendere ACK*
* *così crea un picco di traffico e congestiona la rete alcuni messaggi possono andare perduti*
* *questo innesca una ritrasmissione, che può nuovamente congestionare la rete.*
* *Se viene perduto il byte i-esimo (cioè, il segmento che inizia con il byte i-esimo):*

*esaurita la WIN il mittente smetterà di spedire, attendendo l’ACK del byte perduto*

*mentre il destinatario riceve altri segmenti successivi a quello perduto*

*allo scadere del timeout (lungo) il mittente rispedisce in rapida*

*successione gli ottetti seguenti l’i-esimo (alcuni di questi già*

*ricevuti!)*

*I il mittente inizia a ricevere gli ACK successivi all’i-esimo, ma intanto ha già prodotto un nuovo picco di traffico*

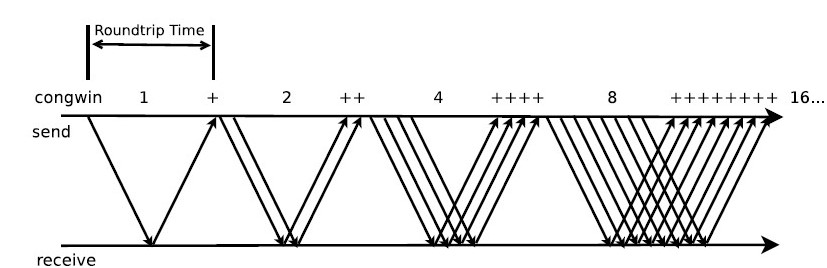
*Problemi: oscillazioni, rispedizioni.*

*Il problema è accentuato se il prodotto*

*(larghezza banda \* ritardo)*

*è alto, ma la tecnica SACK attenua questo fenomeno.*

***Algoritmo Slow start***



*Il mittente mantiene una propria finestra di spedizione, la Congestion Window (CONGWIN)*

*La differenza tra segmenti spediti, e ack ricevuti non essere superiore alla CONGWIN (simile alla finestra di ricezione WIN) La finestra conteggia in unità corrispondenti al Maximum Segment Size (MSS), la massima dimensione ammessa per un segmento, e non in ottetti la CONGWIN ha inizialmente un valore basso (da 1 a 10)*

*Ogni volta che viene ricevuto un ACK, la CONGWIN viene incrementata di 1 MSS: quindi la CONGWIN cresce linearmente con il numero di ACK ricevuti.*

*I Infatti al 10-mo ACK ricevuto la CONGWIN corrisponderà a 10\*MSS (se si parte da 1)*

*Rispetto al tempo però l’incremento è esponenziale: una finestra più ampia accelera il ritmo di spedizione, e quindi alimenta la crescita della CONGWIN. Durante il primo Rountrip Time (il tempo tra la spedizione di un pacchetto, e la ricezione dell’ACK) circola un solo pacchetto, durante il secondo due, poi 4 eccetera.*

*La crescita della CONGWIN si arresta quando si verifica una delle*

*due situazioni: la perdita di un pacchetto, si raggiunge un valore di soglia THRESHOLD (ad esempio*

*RECWIN, o maggiore).*

*Nel primo caso il valore della soglia THRESHOLD viene abbassato a metà del valore della CONGWIN (prima della perdita). A partire da uno dei due eventi, la crescita della CONGWIN cambia passo, e da esponenziale diventa lineare: fase di Congestion Avoidance.*

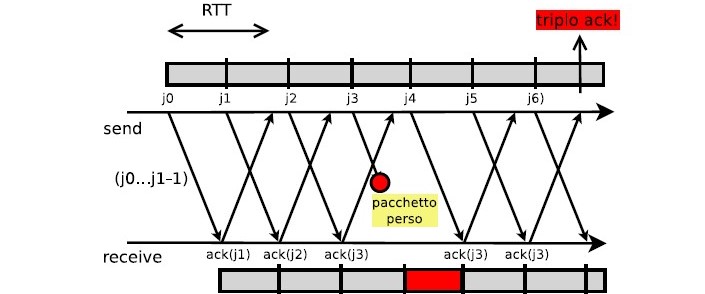
*In questa fase la CONGWIN viene incrementata linearmente nel tempo: si incrementa di un MSS ogni RTT (intervallo tra spedizione, e ricezione dell’ACK)*

*In caso di perdita di pacchetto, l’algoritmo viene reinizializzato con: la CONGWIN = 1, reinizializzando l’algoritmo di slow start e la THRESHOLD dimezzata.*

*Per rilevare la perdita di un pacchetto si utilizza la tecnica del triplo ACK: la ricezione di tre ACK per lo stesso SN indica la perdita di un pacchetto.*

*Il sintomo è affidabile, ed è molto più tempestivo del timeout.*

*Questo significa che dopo la spedizione del byte con un certo valore di ACK, il destinatario ha ricevuto altri due segmenti, senza che l’ACK avanzasse. Questo indica una probabile perdita del segmento che inizia con il byte il cui SN corrisponde all’ACK triplicato. I pacchetti potrebbero essere semplicemente disordinati, ma anche questo è indice di congestione La rilevazione è tempestiva, ed avviene entro due RTT.*



*Esistono diversi algoritmi che utilizzano il triplo ACK come sintomo*

*di congestione:*

***Tahoe:***

*il segmento con tre ACK viene subito rispedito (fast retransmit) l’algoritmo slow start riparte come dopo un timeout, con la soglia dimezzata e CONGWIN di 1 segmento.*

***Reno*** *(quello più usato sino a poco tempo fa)*

*il segmento con tre ACK viene subito rispedito, il mittente si mette in attesa di ricevere tutti gli ack pendenti: se li*

*riceve sia CONGWIN che la soglia vengono dimezzate (fast recovery), altrimenti fa ripartire la slow start.*

*Anche la tecnica del* ***Selective Acknowledge*** *viene usata per non rispedire inutilmente. Gli algoritmi BIC e CUBIC regolano in modo non lineare la CONGWIN (adatti a Wireless) adottati da Linux.*

*Diverse tecnologie a livello 2 determinano problemi specifici*

*Reti inaffidabili (wireless)*

*La perdita di dati spesso è dovuta a degradazione del pacchetto,*

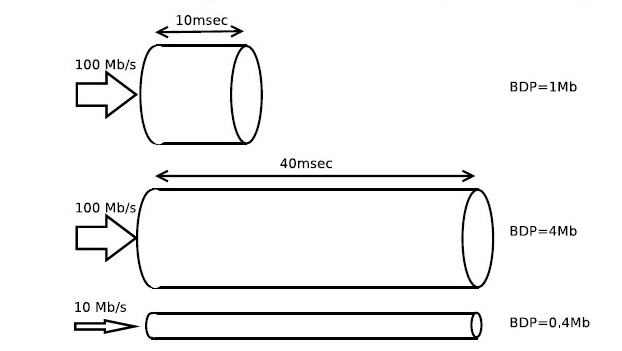
*non a congestione*

*In questo caso diventano fondamentali i metodi di fast recovery*

*nella fase di Congestion Avoidance*

*Long Fat Network: Round Trip Time (RTT) alto, alta velocità (Larghezza di banda). Questo significa che prima di rilevare un problema (servono alcuni RTT) vengono spedite grandi quantità di informazione(BPD=BW\*RTT)*

***Parametro BDP***



*diametro = Byte/Second (Bandwidth)*

*lunghezza = RoundTrip Time (Delay)*

*Il parametro Bandwidth Delay Product (BDP) corrisponde al volume del tubo*

*Significato intuitivo: quanta informazione viene spedita prima di ricevere un ack.*

***Congestione router***

*I responsabili delle perdite di pacchetti sono i router (o gateway). Questi componenti possono essere configurati per evitare o limitare l’impatto di una congestione. Il metodo inefficace per ottenere questo scopo è l’eliminazione di un pacchetto quando la coda di ingresso è piena (tail drop). Questo rimedio ha l’effetto immediato di limitare il traffico. Ma ha pure l’effetto meno immediato di indurre una modifica nei parametri di comunicazione degli utilizzatori (back-pressure). Ad esempio, la perdita di un pacchetto modifica la CONGWIN.*

***Tail Drop e RED***

*Tuttavia ci sono importanti difetti del tail drop:*

* *i pacchetti perduti si concentrano su pochi stream, in quel momento attivi: la perdita di segmenti consecutivi si compensa male (v. Tahoe e Reno)*
* *si possono avere effetti di sincronizzazione molto pericolosi: i flussi danneggiati si ritireranno tutti insieme, per tornare tutti insieme all’attacco!*

*L’algoritmo Random Early Detection (RED) cerca di distribuire la perdita di pacchetti su tutti i flussi, e di anticipare la congestione. La sua natura è probabilistica, in quanto è regolato dalla lunghezza media della coda di ingresso*

*Questa viene valutata con una media mobile, del genere* ***Exponentially Weighted Moving Average.***

***Rimozione pacchetto RED***

*Per ogni pacchetto viene valutata la possibilità di scartarlo, che viene decisa con un procedimento casuale (analogo a gettare un dado).*

*La probabilità che il procedimento indichi la rimozione del*

*pacchetto cresce con la lunghezza della coda*

*rimozione = min (K \_ lunghezzaCoda; 1)*

1. *coda corta, probabilità di rimozione bassa: quando la coda è poco*

*impegnata vengono trattati tutti i pacchetti*

1. *coda lunga, probabilità di rimozione alta: man mano che ci si*

*avvicina alla saturazione la probabilità cresce*

*Invece della semplice rimozione, il pacchetto può essere semplicemente degradato, consentendone la rimozione in stadi successivi.*

***EWMA***

*La EWMA (Exponentially Weighted Moving Average) è una modalità di filtraggio di serie numeriche*

*Il filtraggio è necessario per fare in modo che valori occasionali molto lontani dalla media non abbiano effetti indesiderati. L’idea è quella di avere una media mobile (Moving Average) che tenga conto della storia recente*

*Il peso attribuito dalla EWMA ai valori recenti decresce esponenzialmente andando verso il passato.*

*La EWMA può essere calcolata in maniera molto semplice ed efficace, e per questo il suo uso è molto diffuso.*

*Il procedimento ricorsivo correntemente utilizzato per calcolare la*

*EWMA è il seguente*

***ewmai =(k-1) ewmai-1 +vi: k;***

*Il valore della ewma viene calcolato in funzione*

* *del precedente valore della ewma (ewmai􀀀1), e*
* *del nuovo valore della variabile (vi)*

*Non è necessaria una memoria per i valori passati Il valore della costante k determina quanto peso viene dato alla*

*storia (tipico k=16).*

**Sessione**

Il livello 5 (Session Layer) del modello ISO/OSI è il livello mediante il quale gli utenti possono stabilire dei collegamenti logici o sessioni di lavoro che consentono il corretto trasferimento di informazioni. In particolare, in questo livello vengono definite le regole per aprire e chiudere una connessione logica. Il compito principale e' quindi quello di coordinare il dialogo tra utenti basandosi sul servizio offerto dal [livello di trasporto](http://www.isistassinari.gov.it/progettodicembre/reti/L4.html). In pratica, lo strato di sessione apre una connessione logica tra il nodo e la rete.

**Definizione della sessione**  
Il livello 5 si occupa di attivare la connessione tra due stazioni, mantenerla per tutta la durata del trasferimento dei dati e di terminarla a fine trasmissione. L’intero processo è chiamato appunto sessione; una sessione deve essere individuata, eventualmente interrotta e poi ripresa in base alla necessità.   
**Sincronizzazione**  
Permette ai processi coinvolti nella comunicazione di inserire dei checkpoint (punti di sincronizzazione) in un flusso dati; ciò permette di dividere (logicamente) il flusso in unità piu' piccole in modo che, in caso di interruzione della sessione, non sia necessario inviare nuovamente tutto il flusso dati ma solo dall’ultimo checkpoint in poi.   
**Avvio o apertura**  
Si avvia uno scambio di informazioni tra l'utente che intende utilizzare un servizio e il server che possiede tale servizio.   
**Lavoro in sessione**Il colloquio prosegue, sempre con lo scambio delle informazioni di sessione. Le tipiche applicazioni per il web prevedono spesso che nell'arco della sessione parte di queste informazioni possano essere variate o aggiunte di nuove.   
**Chiusura**  
Su richiesta dell'utente o del client, il server cancella le informazioni di sessione. In assenza di una specifica richiesta, nella gran parte delle applicazioni è prevista la chiusura, o la fine automatica della sessione, dopo un certo tempo in cui l'utente/client non invia alcun messaggio.

**Presentazione**

*Questo è il primo strato che si occupa del significato dei dati. Con stringhe di bit è possibile rappresentare dati di natura più diversa come testi, numeri in virgola fissa, numeri in virgola mobile, ecc.…, rappresentati con codifiche diverse. Lo strato di presentazione si occupa di rendere compatibili le rappresentazioni dei dati negli scambi tra computer che utilizzano codifiche diverse e rende la comunicazione delle applicazioni, presenti su computer con sistemi operativi differenti, in maniera indipendente.  
In questo livello si presentano vari standard, come:*

*ASCII ed EBCDIC (per file di testo)*

*GIF (Graphic Interchange Format)*

*JPEG (Joint Photographic Experts Group)*

*TIFF (Tagged Image File Format)*

*MPEG (Motion Picture Experts Group)*

*MIDI (Musical Instrument Digital Interface)*

*QUICK TIME (per i file con video e audio)*

***La crittografia*** *Per crittografia si intende la comunicazione tra due processi ed un tipo riservato. Le informazioni scambiate non devono essere comprensibili a chi intercettasse un messaggio lungo la linea.*

***La traslazione*** *Le sequenze di informazioni che si scambiano i processi applicativi devono essere convertite in flussi di bit. Lo strato di presentazione cambia in trasmissione il formato dei dati da quello del computer mittente (sintassi locale) a un formato comune (sintassi di trasferimento), in ricezione effettua l'operazione opposta: cambia i dati dal formato comune a quello del computer destinatario. Il livello di presentazione consente la gestione della sintassi delle informazioni trasmesse, diversamente dagli altri livelli che ge**stiscono una sequenza di byte.****Compressione dei bit*** *L’entità di questo livello si occupa, infatti, della compressione dei dati, cioè la modalità di ridurre i bit necessari a immagazzinare e trasmettere le informazioni cercando di lasciare invariata l'informazione a essi associata. Tale compressione permette di ridurre i tempi di comunicazione aumentando di conseguenza la velocita' di trasmissione. Le tecniche di compressione possono essere suddivise in due categorie: Lossy, che utilizzano processi con perdita di informazioni, e Lossless che utilizzano processi senza perdita di informazione. Un esempio di compressione Lossy è la codifica MP3: in questa codifica vengono persi i suoni la cui frequenza non fa parte della banda a cui è sensibile l'orecchio umano, pertanto si puo' ridurre lo spazio occupato di informazione senza una perdita consistente della qualita' del suono. Un esempio di compressione Lossless è la codifica ZIP o RAR dei testi, con la quale non si ha nessuna perdita di informazione che potrebbe comportare la mancanza di leggibilità*

**Applicazione**

*Il livello 7 (Application Layer) occupa lo strato più alto della pila e per questo motivo non fornisce servizi agli altri livelli, ma interagisce direttamente con le applicazioni usate dall’utente, fornendo i servizi di rete. Quindi il livello applicazione si occupa di fornire agli utenti un'interfaccia per accedere alle reti.   
Lo strato di applicazione comprende i protocolli richiesti dagli utenti.  
Esempio di servizio:  
Come principale sistema per la trasmissione d'informazioni sul web, con l'application layer utilizziamo il protocollo HTTP (Hyper Text Transfer Protocol) che applica l'architettura tipica client-server. L'HTTP funziona su un meccanismo richiesta/risposta (client/server): il client esegue una richiesta e il server restituisce la risposta. Nell'uso comune il client corrisponde al browser ed il server la macchina su cui risiede il sito web. Vi sono quindi due tipi di messaggi HTTP: messaggi richiesta e messaggi risposta. (Facciamo riferimento alla figura A).*

*Il layer 7 svolge diversi compiti:*

[*Scambio di e-mail (POP3, SMTP)*](http://www.isistassinari.gov.it/progettodicembre/reti/L7.html)

[*Trasferimento di file (FTP)*](http://www.isistassinari.gov.it/progettodicembre/reti/L7.html)

[*Accesso ai database*](http://www.isistassinari.gov.it/progettodicembre/reti/L7.html)

[*Accesso ai siti Web (DNS, HTTP)*](http://www.isistassinari.gov.it/progettodicembre/reti/L7.html)

[*La gestione remota di applicazioni distribuite*](http://www.isistassinari.gov.it/progettodicembre/reti/L7.html)

[*L’emulazione di terminali*](http://www.isistassinari.gov.it/progettodicembre/reti/L7.html)

*Ognuna delle applicazioni appena elencate utilizza un protocollo specifico dell’Application Layer.*

***HTTP***

*Il protocollo* ***HTTP*** *serve a trasferire contenuti informativi, chiamati* ***risorse****.*

***Una risorsa ha una o più rappresentazione dette entità****, che possono cambiare nel tempo o dipendere da situazioni ambientali.*

***Un programma cliente, spesso un browser*** *(ad esempio Firefox, o Internet Explorer) serve a* ***ricercare l’informazione***

*Più in generale****, il cliente è realizzato da uno User Agent.***

*Un programma server detto* ***Web server******interpreta le richieste del browser e lo aiuta nella ricerca.***

*L’entità associata alla risorsa può essere* ***statica****, cioè già presente sul server, oppure richiedere qualche forma di elaborazione* ***(risorsa dinamica).***

*Per indicare le risorse si pone il problema di un ulteriore livello di indirizzamento (oltre al FQDN del server e alla porta TCP, di solito la 80).*

***La Uniform Resource Location (URL)***

***Una risorsa HTTP viene indicata con una URL****, che contiene tutti gli elementi per identificarla*

*La sintassi di una URL:*

*<scheme>://<fqdn>:<porta><path>? <richiesta>#<frammento>*

*<scheme>:// -* ***il modo in cui viene interpretata la URL*** *(per noi* ***http****)*

*<fqdn>:<porta>* ***- l’indirizzo trasporto del programma server***

*<path> - indicazione della* ***collocazione della risorsa sul server***

*<richiesta> -* ***controlla la rappresentazione dinamica della risorsa***

*<frammento> -* ***indicazione di una parte della risorsa, ad es. un capitolo***

*Ad es.:* [*http://www.antonio.it:12000/prodotti?prezzo<2#ortaggi*](http://www.antonio.it:12000/prodotti?prezzo%3c2#ortaggi)

***Il protocollo di trasporto per HTTP***

*La gestione del protocollo avviene attraverso il protocollo* ***TCP****, spesso cifrato con* ***TLS****.*

*Il protocollo consiste in uno* ***scambio di messaggi****. I messaggi scambiati tra l’apertura e la chiusura della*

*connessione* ***sono una sessione http.***

*Una sessione si articola in una sequenza di transazioni a due fasi composte di:*

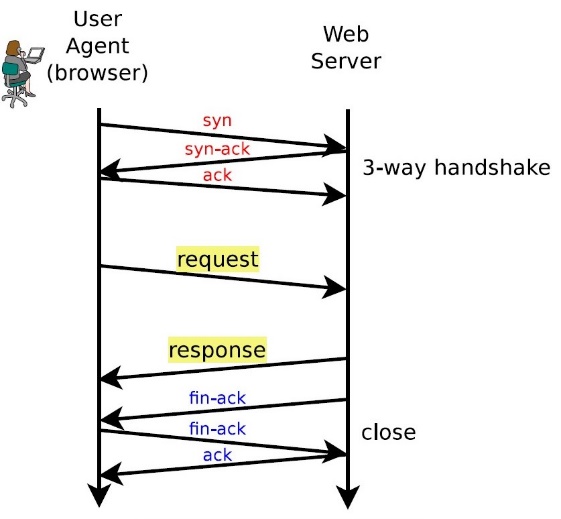
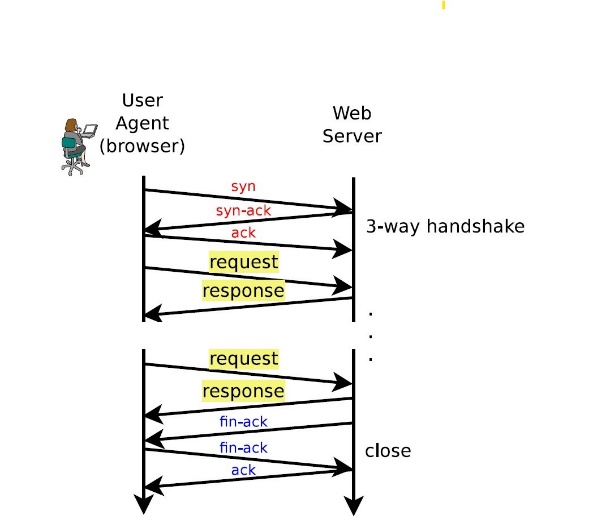
* *un messaggio di richiesta* ***(request)***
* *un messaggio risposta* ***(response)***

*Entro una sessione possono esserci molte transazioni (modalità persistente), o una sola.*

*Il Web Server* ***chiude*** *la sessione dopo un timeout.*

*La* ***Request*** *specifica quale operazione eseguire sulla risorsa, ovvero quale metodo applicare per ottenere l’entità.*

*La* ***Response*** *restituisce l’esito delle Request.*

**

***Formato del messaggio HTTP***

*Un messaggio HTTP inizia con una serie header composto da righe separate da a capo.*

***Una riga vuota conclude lo header***

*Dopo lo header può esserci un* ***body*** *contenente una* ***entità****. La prima riga dello header* ***(start-line)*** *è diversa per i due tipi di messaggi (****Request e Response****).*

*Le righe successive sono nella forma* ***attributo: valore*** *e definiscono una serie di attributi*

*vengono chiamati* ***field*** *(o campi). I campi sono raggruppati in tre categorie:*

1. ***campi general,*** *che si applicano all’intero messaggio*
   1. *ad es. il campo Date corrisponde alla data di produzione del messaggio*
2. ***campi specifici per la request o la response***
3. ***campi entity che si applicano all’entità***
   1. *Ad es. Content-length, Content-type, Title...*

*Spesso i campi sono anche chiamati header, per semplicità*

***Il messaggio di request***

*La* ***start-line*** *di un messaggio request contiene:*

*l’operazione o (metodo) da applicare alla risorsa, il <path> della risorsa, la versione del protocollo.*

*Esempi di attributi definiti in campi Request:*

*Accept: jpeg*

*Authorization: passw*

*User-Agent: mozilla*

*I metodi (o verb) principali sono:*

* ***GET*** *Richiede l’invio della risorsa indicata dal path*
* ***HEAD*** *Richiede l’invio della sola intestazione della risorsa, senza l’entità associata*
* ***POST*** *Indica una modifica alla risorsa (ad es. aggiungere una risposta in un blog)*
* ***PUT*** *Richiede la creazione di una nuova risorsa in corrispondenza al path*
* ***DELETE*** *Richiede la rimozione della risorsa corrispondente al path*

***Il messaggio di response***

*In un messaggio di response la prima riga contiene:*

*la versione del protocollo (HTTP)*

*un codice numerico di tre cifre che indica l’esito dell’operazione*

*una stringa descrittiva dell’esito dell’operazione*

*codici sono divisi in gruppi identificati dalla cifra iniziale*

*gruppi principali:*

*2xx : indica il successo*

*4xx : indica insuccesso per un problema del client*

*5xx : indica insuccesso per un problema del server*

***Esempi di campi nel Response header:***

* *Location: http://www.example.org/index.php*
* *Retry-After: 60*
* *Server: Apache/2.4.1 (Unix)*

***Entity body***

*Dopo lo header ci può essere un* ***body*** *(o corpo) contente una entità. È la rappresentazione di una risorsa, ad es. quella richiesta da una* ***GET****.*

*Il modo di rappresentazione è indicato negli header in termini standardizzati (Campo Content-Type).*

*Ad esempio, per una entità rappresentata in html si usa* ***Content-Type: text/html.***

*Il* ***corpo*** *è una sequenza di ottetti generica, può essere un’immagine in formato jpeg.*

*Il* ***formato*** *del corpo è specificato nello Entity header con il campo* ***Content-Type***

*La* ***lunghezza*** *del corpo è importante per definire la fine del messaggio:*

* *può essere definita nell’intestazione dell’entità*
* *può essere implicita nel formato del corpo*

***il formato del corpo può includere la lunghezza***

***il formato del corpo può indicarne la fine (</html>)***

***Esempio di GET e RESPONSE***

***GET***

*GET /index.html HTTP/1.1*

*Host:* [*www.example.com*](http://www.example.com)

***RESPONSE***

*HTTP/1.1 200 OK*

*Date: Mon, 23 May 2005 22:38:34 GMT*

*Server: Apache/1.3.3.7 (UNIX) (Red-Hat/Linux)*

*Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT*

*Content-Type: text/html; charset=UTF-8*

*Content-Length: 131*

*Connection: close*

*<****html****>*

*<****head****>*

*<****title****>An Example Page</****title****>*

*</****head****>*

*<****body****>*

*Hello, this is a very simple HTML document.*

*</****body****>*

*</****html****>*

**Response puo essere**

200 Ok , 201 Created, 400 Bad Request ,402 Unauthorized,403 Forbidden,404 Not Found,405 Method Not Allowed

***I componenti dell’infrastruttura HTTP***

*Oltre allo* ***User Agent*** *e* ***al Web Server*** *l’infrastruttura del web è composta da altri componenti:*

*i* ***gateway*** *con la funzione di raccogliere le richieste dello* ***UA****, trattarle, e passarle al server, ad esempio per proteggere l’accesso al server.Serve per poter comunicare all esterno della rete.*

*Il* ***proxy*** *(intermediario)con la funzione di apparire come un* ***server*** *allo* ***UA****, e di comportarsi come* ***client*** *per un altro* ***server****: ad esempio per limitare l’accesso ai soli siti aziendali.*

*Ad oggi, i server proxy vengono utilizzati per svariati impieghi come:*

* *Fornire l'anonimato durante la navigazione Internet (es. sistema TOR)*
* *Memorizzare una copia locale degli oggetti web richiesti in modo da poterli fornire nuovamente senza effettuare altri accessi ai server di destinazione (HTTP caching proxy)*
* *Creare una "barriera di difesa" (Firewall) verso il web, agendo da filtro per le connessioni entranti ed uscenti e monitorando, controllando e modificando il traffico interno*

*Lo* ***UA*** *apre la connessione con il* ***proxy****.*

*Il* ***tunnel*** *ha la funzione di istradare semplicemente i messaggi* ***HTTP*** *tra due connessioni diverse: ad esempio un firewall può filtrare le richieste.*

*Nell'ambito delle reti di computer, un* ***protocollo di tunneling*** *permette ad un utente di fornire o accedere ad un servizio non supportato o non fornito direttamente dalla rete. Un uso importante del* ***tunneling*** *è permettere ad un protocollo straniero di essere usato su una rete che non lo supporta; per esempio, usare IPv6 su una rete compatibile solo con IPv4.*

*la funzione di* ***tunnelling*** *viene controllata da uno specifico metodo* ***CONNECT*** *(non visto in precedenza).*

*Se il server (di solito un proxy) accetta la richiesta, da quel punto si limita a inoltrare i pacchetti TCP di quella sessione sull’altro server.*

***La cache nel proxy HTTP***

*Le richieste di accessi alle risorse presentano una certa regolarità che dipende dalla loro popolarità. Per esempio, sul server di un quotidiano le notizie del giorno saranno richieste molto più spesso di quelle risalenti ai giorni precedenti. Per sfruttare questa differenziazione, è possibile mantenere copie delle risorse più frequentemente consultate vicino a chi le richiede.*

*Il* ***proxy*** *è un server intermediario che svolge questa funzione*

* ***come server riceve la richiesta di una risorsa da un cliente***
* ***controlla se ne possiede localmente una copia***
* ***se necessario si comporta da client per richiedere la risorsa ad un altro server, eventualmente un altro proxy***

***La richiesta condizionata***

*Il protocollo* ***HTTP*** *prevede diverse funzionalità che consentono di verificare se l’entità posseduta è o meno aggiornata rispetto alla risorsa:*

* *tra gli header di response è di solito presente un attributo con la data dell’ultimo aggiornamento*
* *il metodo HEAD consente al proxy di scaricare soltanto lo header della risorsa originale, che contiene la data dell’ultimo aggiornamento (Last-Modified)*
* *nel metodo GET è previsto un* ***campo If-Modified-Since****: questo indica che la GET è condizionata dal fatto che l’oggetto sia stato aggiornato*

***Funzionamento della GET condizionale***

*1 Il destinatario della GET condizionale confronta la data della risorsa disponibile con quella indicata nel campo If-Modified-Since*

*2 Se sono identici, spedisce una response 304 Not Modified, senza spedire l’oggetto*

*3 Altrimenti spedisce una response 200 Success contenente l’oggetto.*

*Quindi se l’oggetto sulla cache è aggiornato non viene spedito*

***Altre funzioni del Proxy HTTP***

*Un Proxy HTTP può svolgere altre funzioni correlate alla sicurezza:*

* *consentire l’accesso a determinate risorse esterne, altrimenti rese inaccessibili da un firewall*
* *filtrare le richieste, proteggendo per ragioni di sicurezza determinati server o risorse*

***I cookies***

*Una caratteristica molto importante di HTTP è che non ha uno stato interno per la singola sessione, il modo in cui viene servita una certa* ***request*** *non dipende dalle precedenti, a meno che non venga modificata una risorsa*

*Questa è una caratteristica importante per evitare che il server debba tenere traccia del comportamento di tutti client*

*Tuttavia in certi casi si preferirebbe avere un comportamento del server che dipenda dalla* ***storia*** *passata di un certo User Agent ad esempio, per riprendere un gioco da dove è stato interrotto. Per questa funzione si utilizzano delle registrazioni sullo User agent, i* ***cookies****, in un certo senso****, lo stato del Server viene registrato sullo User Agent.***

*In termini pratici e non specialistici, un cookie è simile ad un piccolo file, memorizzato nel computer da siti web durante la navigazione, utile a salvare le preferenze e a migliorare le prestazioni dei siti web. In questo modo si ottimizza l'esperienza di navigazione da parte dell'utente.*

*Nel dettaglio, un cookie è una stringa di testo di piccole dimensioni inviata da un web server ad un web client (di solito un browser) e poi rimandata indietro dal client al server (senza subire modifiche) ogni volta che il client accede alla stessa porzione dello stesso dominio web.*

***Uso del cookie***

*Il server, al termine della connessione, registra il proprio stato sullo User Agent*

*Per questo inserisce tra le intestazioni della risposta un campo Set-Cookie:*

***Set-Cookie: livello=4***

*In risposta il cliente registra il valore del campo in un file e lo associa a quel sito*

***Questa registrazione è il cookie***

*Lo UA, connettendosi ad un certo web server, inserisce sempre i valori dei cookies associati ad esso, come valori del campo Cookie. Nell’esempio, visitando lo stesso sito invierà il campo:*

* *Cookie: livello=4*
* *Il server utilizzerà questa informazione per alterare il proprio comportamento in quella sessione*

***Dettagli importanti di un cookie***

*Il cookie può essere* ***associato*** *ad una singola risorsa, o ad un web server*

*Il cookie* ***può restare registrato*** *sino ad una data di scadenza, oppure essere* ***cancellato*** *quando lo UA termina*

*Il cookie può essere* ***cifrato****, per evitare che altre applicazioni possano impadronirsi dei dati registrati (ad esempio i dati personali dell’utente), o per evitare che l’utente possa controllare il comportamento del server.*

***Convergenza su HTTP***

*L’infrastruttura Web è ampiamente diffusa, offre importanti servizi che non dipendono dal sistema operativo*

*L’infrastruttura HTTP è progettata per transazioni di due messaggi: richiesta/risposta*

*In molte (recenti) applicazioni basate su HTTP il* ***protocollo domanda/risposta non basta***

*Ad esempio:*

* ***una risposta prolungata nel tempo, come nel caso dello streaming***
* ***molte risposte distribuite nel tempo, come nel caso di aggiornamenti in tempo reale (PUSH)***

*Per queste situazioni sono state trovate soluzioni parziali, che cercavano di aggirare le caratteristiche proprie dell’infrastruttura Web. Recentemente si è cercato di sistematizzare alcune estensioni*

***WebSocket***

*Lo standard* ***WebSocket*** *consente di avviare una connessione simile a* ***TCP*** *entro una sessione* ***http****, lo standard HTML 5.0 comprende l’uso di WebSocket.*

*Per questo viene utilizzata una forma di sessione particolare che contiene uno header, Upgrade, request.*

*Ha lo scopo di* ***concordare*** *(fase di handshake) una particolare funzionalità non disponibile su tutti i server*

*La fase di* ***handshake*** *si articola una singola transazione:* ***request, e response***

*Al termine il browser e il server hanno a disposizione un canale bidirezionale simile a quello fornito da TCP*

*Il canale può essere utilizzato:*

1. ***per scambiare dati tra gli utenti (ad es. in chat)***
2. ***per trasferire dati ad un’applicazione (streaming multimediale)***
3. ***per acquisire dati in tempo reale (publish/subscribe)***

***Esempio***

***Lo UA richiede la connessione***

*GET /text HTTP/1.1\r\n*

*Upgrade: WebSocket\r\n*

*Connection: Upgrade\r\n*

*Host: www.websocket.org\r\n*

*..Altri campi...*

*\r\n*

***Il Web server accetta la connessione***

*HTTP/1.1 101 WebSocket Protocol Handshake\r\n*

*Upgrade: WebSocket\r\n*

*Connection: Upgrade\r\n*

*...altri campi...*

*\r\n*

***La risposta 101 indica che l’upgrade è stato accettato. Da questo punto in poi il server può comunicare con lo UA senza necessità di una request.***

***Il paradigma REST***

*Nella sua forma più semplice si basa sull’utilizzo dei metodi propri del protocollo HTTP (Fielding stesso ha lavorato sullo standard) per effettuare tramite semplici richieste al server le consuete operazioni CRUD, garantendo al tempo stesso un facile disaccoppiamento tra client e server.*

*Il termine REST rappresenta un sistema di trasmissione di dati su HTTP senza ulteriori livelli (quali ad esempio* [*SOAP*](https://it.wikipedia.org/wiki/SOAP)*). I sistemi REST non prevedono il concetto di sessione (sono, come approfondito successivamente, stateless).*

*L’approccio REST (Representational State Transfer)*

***HTTP deve la sua fortuna all’essere un protocollo semplice***

*I fondamenti del paradigma REST sono:*

* *distinzione dei due ruoli client/server attorno alla transazione a due passi (request/response)*
* *protocollo* ***stateless****: il server non conserva uno stato della transazione.*
* *uso esplicito della cache: l’utente specifica se vuole utilizzare risultati in cache*
* *compatibilità con l’infrastruttura HTTP (proxy, gateway...). La motivazione per l’introduzione del paradigma REST (2001) è una specie di controriforma nei confronti del paradigma Remote Procedure Call, rappresentato da Java RMI e SOAP*

*Il* ***vantaggio*** *in questo contesti rispetto a tecnologie più impegnative come SOAP/RPC è nella riduzione della parte di configurazione e nella riduzione dei parametri di contollo: non abbiamo cioè bisogno di spedire “messaggi” per informare client o server di ciò che sta per avvenire. La semplicità ottenuta ha trovato una ampia adesione presso moltissimi framework e api, e piattaforme “social” molto note.*

***Java RMI e SOAP***

*Nella prospettiva RPC (anni 90) il web server è in grado di essere invocato per operazioni generiche, definite dal server*

*Quindi si supera la visione tradizionale HTTP, che limita l’interazione ai 4 verbi. Questo ha generato un certo numero di problemi, ed una architettura delle soluzioni decisamente complessa: ad esempio nella serializzazione dei dati per* ***RMI.***

*L’architettura software è appesantita da strutture destinate alla descrizione del servizio. Il paradigma REST interviene per ristabilire il primato della filosofia HTTP, rispetto ad estensioni.*

***REST e WebSocket***

***WebSocket*** *propone un altro genere di astrazione, in genere accompagnata dall’uso di protocolli per lo scambio di messaggi (come XMPP)*

*Si tratta di una visione ancora diversa, ma compatibile con una visione REST*

*REST ammette che possano essere negoziate nuove modalità di comunicazione. Dal punto di vista dello sviluppatore, l’uso dei messaggi è estremamente più naturale dell’RPC: almeno questo è il sentimento corrente.*

***IoT e convergenza su HTTP***

*IoT sta per Internet of Things*

*L'Internet delle cose è una possibile evoluzione dell'uso della Rete: gli oggetti (le "cose") si rendono riconoscibili e acquisiscono intelligenza grazie al fatto di poter comunicare dati su se stessi e accedere ad informazioni aggregate da parte di altri*[*[7]*](https://it.wikipedia.org/wiki/Internet_delle_cose#cite_note-7)*. Le sveglie suonano prima in caso di traffico, le scarpe da ginnastica trasmettono tempi, velocità e distanza per gareggiare in tempo reale con persone dall'altra parte del globo, i vasetti delle medicine avvisano i familiari se si dimentica di prendere il farmaco. Tutti gli oggetti possono acquisire un ruolo attivo grazie al collegamento alla Rete.*

*In breve, usare le tecnologie disponibili per raccogliere dati(monitorare) il nostro ambiente di vita...*

*...ed eventualmente innescare risposte automatiche intelligenti*

*È un hype: una moda più o meno passeggera che mobilita grandi investimenti*

* *grande entusiasmo iniziale con grandi promesse*
* *grande richiesta e conseguenti investimenti*
* *delusione per risultati modesti e scarso ritorno economico*
* *stabilizzazione sui risultati acquisiti e cristallizzazione*

***Le promesse di IoT:***

*IL miglioramento della qualità della vita, maggiori condizioni di sicurezza, risparmi energetici*

***I dati trasportati con HTTP***

*La modalità con cui i* ***dati vengono trasferiti*** *sul server* ***ThingSpeak*** *si basa su* ***http****, è un esempio significativo dei molti usi di http.*

*I dati vengono inseriti in messaggi di POST inviati al server. Vediamo una implementazione di un semplice sensore*

1. *rileva le caratteristiche operative di una macchina*
2. *le codifica in una serie di parametri di un POST*
3. *invia il POST*

***Il programma è scritto in Python***

*Potete scaricarlo clonando il repository*

*https://bitbucket.org/labreti/thingspeak-tmt.git.*

*ricopiate la Write key nel campo key*

*installate il pacchetto python-psutil (apt-get install python-psutil)*

*Dopo averlo lanciato, tornate sul sito di ThingSpeak e visualizzate il grafico dei dati*

***WEB FRAMEWORK***

*è un* [*framework software*](https://it.wikipedia.org/wiki/Framework) *progettato per supportare lo sviluppo di* [*siti web*](https://it.wikipedia.org/wiki/Sito_web)[*dinamici*](https://it.wikipedia.org/wiki/Web_dinamico)*,* [*applicazioni web*](https://it.wikipedia.org/wiki/Applicazione_web) *e* [*servizi web*](https://it.wikipedia.org/wiki/Web_service)*.*

*Lo scopo del framework è quello di alleggerire il lavoro associato allo sviluppo delle attività più comuni di un'applicazione web da parte dello* [*sviluppatore*](https://it.wikipedia.org/wiki/Programmatore)*. Molti framework forniscono ad esempio delle* [*librerie*](https://it.wikipedia.org/wiki/Libreria_(software)) *per l'accesso alle* [*basi di dati*](https://it.wikipedia.org/wiki/Basi_di_dati)*, per la creazione di* [*template*](https://it.wikipedia.org/wiki/Template)[*HTML*](https://it.wikipedia.org/wiki/HTML) *o per gestire la* [*sessione*](https://it.wikipedia.org/wiki/Sessione) *dell'utente. Uno dei principi fondamentali è riassunto dall'acronimo DRY ([don't repeat yourself](https://it.wikipedia.org/wiki/Don%27t_repeat_yourself" \o "Don't repeat yourself)), nel senso che viene fortemente consigliata l'adozione di tecniche di* [*riuso di codice*](https://it.wikipedia.org/wiki/Riuso_di_codice)*.*

***Gestione di un servizio Web***

*Il protocollo* ***HTTP*** *permette di connettersi ad un server web per sottoporre* **REQUEST.**

*Sappiamo che una request può essere di diversi tipi: oltre alla GET abbiamo PUT, POST ecc.*

*Se alla GET la risposta può essere statica, restituendo un pacchetto con HTML, CSS e Javascript, agli altri verb si associa in genere una elaborazione.* ***Ancora di più quando HTTP serve come base per una API REST***

*Anche un* ***CMS****(è uno strumento* [*software*](https://it.wikipedia.org/wiki/Software)*, installato su un* [*server web*](https://it.wikipedia.org/wiki/Server_web)*, il cui compito è facilitare la gestione dei contenuti di* [*siti web*](https://it.wikipedia.org/wiki/Sito_web)*, svincolando il* [*webmaster*](https://it.wikipedia.org/wiki/Webmaster) *da conoscenze tecniche specifiche di* [*programmazione Web*](https://it.wikipedia.org/wiki/Programmazione_Web)*.) tuttavia associa spesso una elaborazione lato server per l’esecuzione di una* ***GET.***

*Ad esempio, per differenziare il contenuto di una pagina tra utenti registrati e non.*

*In questo caso chi sviluppa il servizio deve gestire in modo articolato la* ***request****.*

***Elaborazione lato server***

*Sappiamo che* ***HTTP*** *viene trasportato dal protocollo* ***TCP****, e sappiamo come ricevere e spedire uno stream TCP*

*Tuttavia la complessa gestione delle richieste HTTP attraversa un percorso in larga parte prevedibile ed automatizzabile. Quindi sono state realizzate delle architetture software* ***server-side*** *che semplificano il lavoro dello sviluppatore.*

*Si chiamano web* ***framework*** *e sono composte da:*

* *un server HTTP che, sul server, gestisce la ricezione delle* ***request e*** *la spedizione dei* ***response***
* *una o più applicazioni accessorie, come l’organizzazione dei dati da inserire nelle response, o la gestione di altri protocolli di comunicazione*
* *una serie di librerie che offrono un accesso semplificato alle funzionalità realizzate al server web e alle* ***applicazioni***

***Tentativo di prospettiva storica***

*Il primo strumento che può essere assimilato all’uso dei web framework è CGI (Common Gateway Interface)*

*Un programma CGI (script) è scritto in un qualsiasi linguaggio adatto (ad es. Perl o Python)*

*Il documento HTML contiene un form che invoca lo script CGI. Lo script produce un documento HTML, quindi nel suo codice dominano frammenti di HTML trattati come stringhe*

*Il server web (ad es. Apache)*

* *invoca il programma CGI*
* *aggiunge un header HTTP*
* *spedisce la risposta al cliente*

*Semplice come struttura, ma complesso nell’uso: il programma è infarcito di codice html. Il bundle LAMP segue questo schema*

***I web framework attuali***

*Un* ***web framework*** *è strettamente legato ad un* ***linguaggio di programmazione****, mentre il* ***CGI esegue qualunque programma eseguibile come script***

*Non esiste una associazione path! file*

* *Apache associa ad un path nella URL un path analogo in una directory (ad es. /var/www)*
* *nel web framework esiste un costrutto linguistico, la route, che ha un funzionamento simile a quello di uno switch in un linguaggio di programmazione*

*Le pagine dinamiche non sono prodotte da programmi nel linguaggio del framework*

* *vengono invece generati a partire da file di template*
* *hanno la struttura di un file HTML, ma contengono espressioni che vengono valutate utilizzando valori variabili dinamicamente*

*Ce ne sono molti. Solo per Python ne esistono almeno due principali:* ***Django e Flask****.*

*Noi introdurremo le caratteristiche principali di Flask.*

***Flask***

***Flask è un web framework per il linguaggio Python****. Rispetto ad altri offre funzionalità minimali, ma è facilmente*

*espandibile e configurabile*

*Quindi esistono un gran numero di librerie, per tutte le funzioni principali*

*Solo il linguaggio Jinja per i template è indicato come default*

* *ma può essere comunque cambiato Invece non esiste un legame specifico con un database*
* *può essere usato SQL importando una opportuna libreria*

***Model, View, Controller con Flask***

*Il* ***pattern Model View Controller*** *(MVC) è uno schema generico che si applica anche al web frameworks*

*In Flask possiamo sovrapporre lo schema MVC così:*

*Il Model si traduce nel contenuto del database, rappresentato da oggetti Python. Flask non porta con sè un database, ma esistono librerie per utilizzarne un certo numero*

*La View si traduce nei template, e lo strumento per controllarli è Jinja*

*Il Controller è definito dalle route inserite nel programma Python principale*

*Nel nostro percorso incontreremo prima il* ***Controller*** *rappresentato dalle route*

*Poi introdurremo la parte di View considerando semplici esempi di template*

*Infine un accenno alla rappresentazione dei dati nel Model, ma senza introdurre un vero database*

***Cominciamo con un hallo world***

***from*** *flask* ***import*** *Flask*

*app = Flask(\_\_name\_\_)*

*@app.route (’/’)*

***def*** *hello\_world():*

***return*** *’Hello, World!’*

*La prima riga carica la libreria*

*La seconda lancia il web server*

*Dopo viene definita una singola route, corrispondente al path vuoto /*

*L’elaborazione associata alla route è rappresentata da una funzione*

*Il risultato della funzione è la pagina HTML inviata al client*

***Aggiungiamo una route***

***From*** *flask* ***import*** *Flask*

*app = Flask(\_\_name\_\_)*

*@app.route(’/’)*

***def*** *index():*

***return*** *’Index Page’*

*@app.route(’/hello’)*

***def*** *hello():*

***return*** *’Hello, World’*

*Qui viene aggiunta la risposta alla route /hello*

***Gestiamo un post***

***from*** *flask* ***import*** *Flask,request*

*app = Flask(\_\_name\_\_)*

*@app.route(’/’, methods=[’GET’, ’POST’])*

***def*** *login():*

***if*** *request.method == ’POST’:*

***return*** *"Hallo "+request.form[’username’]+"!"*

***else****:*

***return*** *’’’*

*<h2>Come ti chiami?</h2>*

*<form action="" method="post">*

*<input type="text" name="username">*

*<input type="submit" value="Invia">*

*</form>*

*’’’*

*Nella definizione della route (terza riga) viene specificato che si* ***risponde anche al metodo POST***

***di default risponde solo a GET***

*Nella funzione associata alla route ci sono due rami in un costrutto if-else*

***Gestiamo un post***

***from*** *flask* ***import*** *Flask,request*

*app = Flask(\_\_name\_\_)*

*@app.route(’/’, methods=[’GET’, ’POST’])*

***def*** *login():*

***if*** *request.method == ’POST’:*

***return*** *"Hallo "+request.form[’username’]+"!"*

***else****:*

***return*** *’’’*

*<h2>Come ti chiami?</h2>*

*<form action="" method="post">*

*<input type="text" name="username">*

*<input type="submit" value="Invia">*

*</form>*

*’’’*

*I due rami dipendono dal metodo indicato nella request*

* *I Se il metodo è GET restituisce la pagina con il form*
* *I Se il metodo è POST viene stampata la pagina di benvenuto*

***Introduciamo i template***

***from*** *flask* ***import*** *Flask,request,render\_template*

*app = Flask(\_\_name\_\_)*

*users=["luigi","gianni","maria"]*

*@app.route(’/’, methods=[’GET’, ’POST’])*

***def*** *login():*

***if*** *request.method == ’POST’:*

***if*** *request.form[’username’]* ***in*** *users:*

***return*** *render\_template("hallo.html",nome=request.form[’username’])*

***else****:*

***return*** *"Non autorizzato"*

***else****:*

***return*** *render\_template("login.html")*

*L’applicazione ora riconosce se l’utente è incluso nella lista*

*Invece di inserire l’html nel testo, si usano template*

*La funzione render\_template invoca Jinja*

***Il template login.html***

*<****html****>*

*<****body****>*

*<****h2****>Come ti chiami?</****h2****>*

*<****form action****=""* ***method****="post">*

*<****input type****="text"* ***name****="username">*

*<****input type****="submit"* ***value****="Invia">*

*</****form****>*

*</****body****>*

*</****html****>*

*Questa è una semplice pagina statica*

***Il template hallo.html***

*<****html****>*

*<****body****>*

*<****h2****>Ciao {{nome}}!</****h2****>*

*</****body****>*

*</****html****>*

*Questa invece è una pagina dinamica*

***Jinja*** *sostituisce l’espressione {{nome}} con la variabile nome indicata nell’invocazione*

*verificate nel primo return della funzione login*

***Uso dei cookies***

***from*** *flask* ***import*** *Flask,request,render\_template, make\_response*

*app = Flask(\_\_name\_\_)*

*users=["luigi","gianni","maria"]*

*@app.route(’/’, methods=[’GET’, ’POST’])*

***def*** *login():*

***if*** *request.method == ’POST’:*

***if*** *request.form[’username’]* ***in*** *users:*

*resp=make\_response(render\_template("hallo.html",nome=request.form[’username’]))*

*resp.set\_cookie("utente",request.form[’username’])*

***return*** *resp*

***else****:*

***return*** *"Non autorizzato"*

***else****:*

***return*** *render\_template("login.html")*

*@app.route(’/id’, methods=[’GET’])*

***def id****():*

*yourname=request.cookies.get("utente")*

***return*** *"Sei "+yourname;*

***NAT***

*Con l’allargamento dell’uso di Internet, il numero di utenti è aumentato, e l’acquisizione di un indirizzo IP pubblico è diventata onerosa.*

*Un profilo di utente specifico è quello dell’utente che può accedere ad Internet, senza però offrire servizi.*

*Questa categoria di utenti* ***non ha necessità di avere un IP pubblico.***

*In altri termini si tratta di un utente che usufruisce di servizi, senza però offrirne.*

***Circostanze:***

*una impresa o ufficio che dispone di computer connessi ad Internet, ma che non ha interesse (o non vuole) rendere accessibili all’esterno alcuni di questi*

*un cliente di un fornitore di servizi che offre il collegamento ad Internet (eventualmente dinamico).*

***Network Address Translation***

*La soluzione in questi casi è un router che, oltre alle consuete operazioni di routing, modifichi gli indirizzi IP (NAT, Network Address Translation)*

*A valle del router verranno utilizzati degli indirizzi IP privati che solo il router conosce*

***Gli indirizzi privati*** *devono essere scelti in un range ben definito: i più utilizzati sono 192.168.0.0/16 e 10.0.0.0/8*

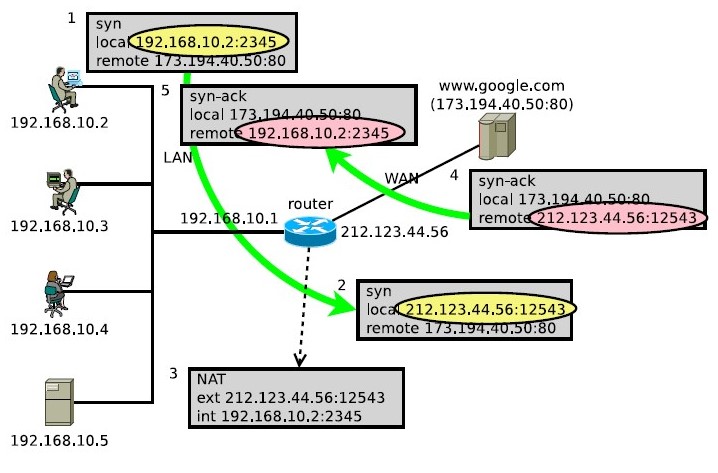
*La rete a valle del router viene anche detta intranet, oppure rete*

*mascherata (masquerade) attenzione, niente a vedere con la maschera di rete!*

*All’interfaccia del router rivolta verso l’esterno può essere associato un indirizzo IP pubblico.*

***Configurazione Router***

*Il router traduce così gli indirizzi che trova nei pacchetti IP:*

* *nei pacchetti provenienti dall’interno e diretti all’esterno dell’intranet l’indirizzo privato del mittente viene sostituito con l’indirizzo pubblico del router*
* *nei pacchetti provenienti dall’esterno l’indirizzo del destinatario viene sostituito con un indirizzo interno appropriato*

*La seconda operazione richiede che il router capisca a quale degli host interni è destinato il pacchetto IP*

*Il router dunque mantiene un* ***soft state*** *per le connessioni aperte dai nodi interni all’intranet*

*Per queste, il router ha abbastanza informazioni per poter associare un pacchetto entrante ad un certo host interno*

*Infatti i pacchetti del* ***3 way handshake*** *specificano gli IP e le porte di client e server: bastano a determinare il destinatario di un pacchetto entrante.*

***Un server nell’intranet (port forwarding)***

*Non va altrettanto bene per le connessioni aperte dall’esterno: un cliente esterno non può indicare un server*

*nell’intranet.*

*In realtà la situazione è anomala: di solito un server ha un IP pubblico.*

*Il router in questo caso viene configurato per inviare le comunicazioni su una certa porta verso uno specifico host interno (port redirection).*

*Ad esempio,*

*il droghiere Giuseppe ha in negozio un PC alla cassa e uno in magazzino collegati da una piccola rete*

*a monte ha un piccolo router/modem ADSL collegato ad un provider che gli associa un singolo indirizzo IP dinamico*

*Giuseppe vuole poter accedere in SSH (porta 22) al PC del magazzino da casa sua, collegata ad Internet su un’altra linea*

***Soluzione:***

*configura il router in negozio in modo che le richieste di connessione dall’esterno verso la porta 22 del router vengano*

*tradotte dal router in richieste verso il PC alla cassa prima di lasciare il negozio, annota l’IP del router.*

***Peer-to-peer***

*I sistemi peer-to-peer (anche p2p) nascono per rendere possibile la condivisione di risorse tra gli utenti, ma sappiamo che lo stesso problema sussiste tra i server.*

*Un sistema peer-to-peer si caratterizza per il fatto che le risorse oggetto del servizio sono distribuite tra gli utenti*

*Quindi accade che gli utenti comunicano prevalentemente tra di loro, non con un server, per condividere le risorse*

*Dal punto di vista di TCP, gli utenti si possono comportare da* ***clienti****, nel momento in cui richiedono risorse ad altri utenti, o da* ***server****, nel momento in cui cedono le proprie risorse.*

*Esempi di sistemi peer-to-peer:*

* *vari sistemi di condivisione di file (anche multimediali)*
* *streaming in multicast (peercast)*
* *servizi di telefonia in Internet, come il primo Skype*
* *i giochi in rete*
* *grandi basi di dati di genere scientifico*
* *middleware per gestione di infrastrutture complesse*

***Dinamicità di una rete p2p***

*La composizione e l’organizzazione del sistema cambia continuamente, in quanto gli utenti possono entrare e uscire*

*Questo può limitare l’affidabilità del sistema, in quanto una delle risorse può essere non disponibile in un certo istante, o andare perduta.*

*Il sistema può basarsi sulla presenza di una* ***registry*** *contenente:*

* *l’insieme degli utenti presenti,*
* *la distribuzione delle risorse (chi ha cosa)*

*Ad esempio, se sto cercando un certo brano musicale, chiederò alla* ***registry*** *quali utenti lo possiedono, e quali di questi è online. La modalità di realizzazione della ricerca delle risorse è un fattore fondamentale.*

***Registry centralizzata***

*In questo caso esiste un unico server di* ***registry****, gestito da qualche forma di organizzazione (tipo Napster)*

*La tecnica cui viene realizzata la* ***registry*** *è detta* ***hash table****.*

*Remind:*

* *il cliente consegna una* ***chiave di ricerca***
* ***il server estrae dalla chiave gli elementi rilevanti e li organizza in una nuova chiave sintetica***
* *la chiave sintetica viene ricercata nella prima colonna della* ***hash table***
* *in corrispondenza, nella seconda colonna troveremo un insieme di dati corrispondenti, tra i quali quello cercato.*

***Il server centralizzato (NAPSTER)***

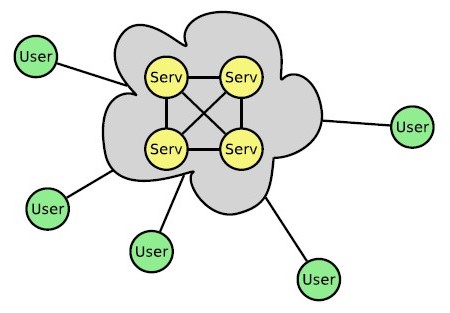
*Il server centralizzato consente una forma di controllo, e quindi eventualmente proventi economici*

*Per la stessa ragione è anche controllabile dall’esterno, se il sito viene oscurato per qualche ragione (anche lecita...) il*

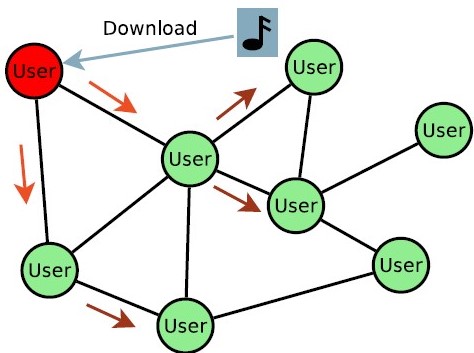
*servizio si spegne.*

*Da un punto di vista tecnico introduce un collo di bottiglia, un punto in cui tutti devono passare che condiziona prestazioni ed affidabilità*

***Questo limita la scalabilità dell’architettura, mentre il p2p mira ad architettura estremamente scalabili.***

**

***Gnutella: eliminare il registro.***

******

***Non esiste una funzionalità di registro:*** *ogni nodo conosce solo i file locali.*

*Gli utenti inviano le richieste in multicast* ***flooding****. Un protocollo specifico costruisce dinamicamente una rete sovrapposta (overlay) a quella di Internet.*

***Costruzione della rete in overlay***

*Si tratta di una rete composta* ***da connessioni TCP****, quindi due nodi vicini su questa rete possono essere lontani in termini di Internet (e viceversa).*

*Nel momento in cui un nuovo cliente entra nella rete per la prima volta, deve contattare una* ***registry*** *nota, e da questa risalire ad altri nodi utente da utilizzare come vicini, solo in questa fase viene utilizzato un numero limitato di server di registry.*

***Flooding***

*A questo punto, un utente che richiede una risorsa* ***invia la richiesta ai propri vicini****, che a loro volta la inoltrano ai propri vicini(flooding).*

*La richiesta contiene un contatore (****TTL****), che viene decrementato ogni volta che la richiesta viene inoltrata*

*L’utente che riceve una richiesta con il contatore uguale a zero non inoltra ulteriormente la richiesta*

*L’utente che riceve una richiesta che può soddisfare invia il proprio indirizzo IP in modo che l’interessato possa stabilire la connessione.*

*L’uso di* ***port forwarding*** *consente la partecipazione anche di utenti che usano un router NAT. ( Nelle* [*reti informatiche*](https://it.wikipedia.org/wiki/Rete_informatica) *il* ***port forwarding*** *è l'operazione che permette il trasferimento dei dati (forwarding) da un* [*computer*](https://it.wikipedia.org/wiki/Computer) *ad un altro tramite una specifica* [*porta*](https://it.wikipedia.org/wiki/Porta_(reti)) *di comunicazione. Questa tecnica può essere usata per permettere ad un utente esterno di raggiungere un host con* [*indirizzo IP privato*](https://it.wikipedia.org/wiki/Indirizzo_IP_privato) *(all'interno di una* [*LAN*](https://it.wikipedia.org/wiki/LAN)*) mediante una porta dell'*[*IP pubblico*](https://it.wikipedia.org/wiki/IP_pubblico) *dello stesso. Per eseguire questa operazione si ha bisogno di un* [*router*](https://it.wikipedia.org/wiki/Router) *in grado di eseguire una traduzione automatica degli indirizzi di rete, detta* [*NAT*](https://it.wikipedia.org/wiki/Network_address_translation)*.)*

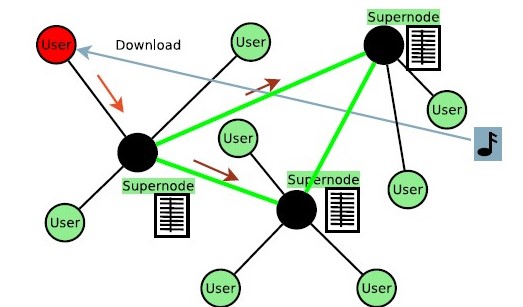
***Problemi con Gnutella***

*Questo protocollo ha due problemi:*

* *da un lato il* ***flooding*** *che, per quanto limitato dal TTL delle richieste,* ***genera una quantità di traffico notevole.***
* *dall’altro, proprio* ***il TTL limita il raggio di ricerca****, quindi mette a disposizione dei clienti solo una parte delle risorse realmente disponibili.*

*Diversamente dal caso precedente, questa alternativa non garantisce un ritorno economico da parte del produttore del codice, che non ha problemi di diritti di autore,* ***chi gestisce il contenuto è solo l’utente****, che può essere perseguito.*

***Gli utenti realizzano anche un registro (KaZaA)***

******

*L’applicazione che realizza* ***l’utente*** *analizza la capacità di* ***calcolo, la connettività, e la frequenza di connessione dell’host su cui è in esecuzione.***

***Se queste sono superiori ad una certa soglia****, l’host viene utilizzato come* ***server (supernode),*** *e farà da registry per un certo numero di utenti.*

***Le registry*** *si organizzano in una rete* ***overlay*** *simile a quella di Gnutella, ma di estensione notevolmente inferiore.*

***Distributed Hash Table***

*Insieme, i supernodi implementano una struttura dati distribuita, che viene chiamata* ***Distributed Hash Table*** *(anche* ***DHT****)*

*In questa modalità si utilizza una gerarchizzazione a due livelli, utilizzando le tecniche di* ***flooding*** *e* ***accesso tramite DHT.***

***KaZaA*** *realizza proventi non rilasciando il sorgente (ma l’eseguibile può essere scaricato gratuitamente) ma inserendo*

*messaggi pubblicitari (a pagamento).*

*Tuttavia, analizzando il suo funzionamento ed il traffico generato (reverse engineering), KaZaA è stato clonato, eliminando la pubblicità :-)*

*I tre applicativi usati negli esempi (Napster, Gnutella e KaZaA), hanno avuto il loro momento di gloria e sono poi stati superati da altri (aMule): restano validi come esempio.*

*Tutti hanno avuto grossi problemi per la violazione dei diritti d’autore, attenuati dal fatto che il software rilasciato non è responsabile del contenuto trasferito*

*Quindi, gli eventuali responsabili di violazione dei diritti d’autore erano gli utenti, spesso inconsapevoli e perciò incolpevoli, che mettevano a disposizione di altri contenuti coperti da copyright*

*dopo aver seguito questa lezione siete perseguibili...*

***Il protocollo Bittorrent***

***Bittorrent è un protocollo che viene utilizzato per trasferire file di grandi dimensioni attraverso reti P2P***

*In questo caso la probabilità che l’utente che fornisce il file esca dalla rete non è trascurabile (con conseguente interruzione del download)*

*La soluzione è quella di frammentare il file di dimensioni considerevoli (ad esempio, un CD di installazione, o un filmato), distribuendo i frammenti su più clienti.*

*Nel caso in cui i frammenti scaricati provengano da host serviti da un collegamento* ***ADSL*** *(Asymmetric (perché la velocita di down è differente da quella di up) Digital Subscriber Line) che hanno limitazioni di banda in uscita, il sistema* ***aumenta la velocità di download****, in questo caso più download possono essere eseguiti contemporaneamente da host diversi, incrementando la velocità complessiva.*

***La distribuzione di un file***

*Il nodo* ***seed*** *che vuole mettere a disposizione un file tramite* ***BitTorrent*** *lo frammenta e produce un file di descrizione, con* ***estensione .torrent*** *che contiene:*

* ***l’identificazione del tracker****, il server che gestirà la distribuzione del file.*
* ***le checksum dei frammenti*** *in cui è stato suddiviso il file (per evitare alterazioni)*

***Il .torrent*** *viene consegnato al* ***tracker*** *che cura la distribuzione del file.*

*Gli altri utenti che vogliono scaricare il file si mettono in contatto con il* ***tracker*** *(tramite un* ***indexer****) per sapere da dove scaricare i frammenti.*

*Mano a mano che gli interessati scaricano il file, vengono registrati dal* ***tracker*** *e diventano a loro volta distributori dei singoli frammenti.*

*Il* ***tracker*** *è un punto critico di* ***Bittorrent****.*

***Utenti dietro un NAT o firewall***

*Buona parte dei protocolli peer-to-peer si rivolgono alle* ***utenze periferiche****, agganciate a linee* ***ADSL****, molto spesso dietro un* ***router NAT*** *(Network Address Translation).*

*In questi casi il nodo periferico deve poter indicare una porta attraverso la quale è raggiungibile*

* *in questi protocolli si usa prevalentemente* ***UDP***
* *nel P2P si usa anche* ***TCP***

*Questo è un problema in quanto il NAT impedisce la funzionalità server agli host nell’intranet, non sempre è possibile (e comunque pericoloso) configurare un* ***port forwarding*** *e**il NAT maschera gli indirizzi interni.*

*In questi casi si utilizza un protocollo di* ***NAT traversal****, che rivela all’utente il modo in cui viene reso visibile dal suo NAT.*

***STUN: un protocollo per il NAT traversal***

*Il protocollo* ***STUN*** *mette a disposizione gli strumenti per scoprire come il nodo sia visibile alla rete.*

*Per questa operazione utilizza un* ***server esterno****, che gli fa da sponda. L’operazione fondamentale di* ***STUN*** *è lo* ***scambio*** *di due pacchetti di "****binding****" tra* ***il cliente e il server****. Così viene a conoscere l’indirizzo* ***IP: porta*** *attraverso il quale altri possono raggiungerlo.*

*Utilizzando* ***STUN*** *si possono implementare diversi algoritmi per il* ***NAT traversal.*** *Uno di questi era originariamente indicato in un* ***RFC****, successivamente deprecato.*

***Operazione di binding di STUN***

*Il* ***client*** *spedisce un pacchetto al server specificando la* ***modalità della risposta da parte del server.***

*1 usando lo stesso IP e porta da cui riceve il pacchetto oppure*

*2 sostituendo IP o porta con quelli indicati dal cliente*

*Il server segue le modalità indicate dal client e invia la risposta.*

*Il server riporta sempre nel pacchetto di risposta l’indirizzo da cui ha ricevuto il pacchetto in andata.*

*Vediamo (semplificando) come il cliente viene a conoscere la coppia ip/porta in cui viene tradotto il proprio indirizzo*

* *solo* ***Full cone NAT****, come lo abbiamo visto nella lezione dedicata al NAT.*

*Il procedimento fallisce se il* ***firewall*** *associa coppie IP: porta* ***diversi*** *anche in funzione della destinazione, una contromisura per impedire il NAT traversal.*

***Algoritmo (deprecato) di NAT traversal***

*Il cliente chiede al server una risposta di* ***tipo 1****: se non riceve risposta i pacchetti non passano e basta*

*Altrimenti controlla l’indirizzo segnalato dal server, se corrisponde a quello del client, non c’è NAT, altrimenti c’è NAT e si acquisisce l’IP e la porta del pacchetto.*

***A questo punto è noto l’IP e la porta assegnata dal NAT all’utente.***

*Richiedendo al server la modifica degli indirizzi di risposta si arriva a determinare anche la presenza di firewall.*

***La sicurezza dei dispositivi in Internet***

*Un capitolo della sicurezza di Internet riguarda la conservazione dei dati e delle attrezzature collegate alla rete*

*I modi per attaccare un host connesso ad Internet sono molteplici, ma in genere la collocazione in una rete mascherata già fornisce una discreta protezione.*

*In genere si distinguono due generi di attacco:*

* *passivo, se è mirato ad acquisire informazioni circa i dispositivi in rete: ad esempio, intercettando le comunicazioni si possono acquisire informazioni sugli indirizzi IP e le porte utilizzate.*
* *attivo, generalmente è preceduto da una fase passiva, e mira a danneggiare i dispositivi.*

***Attacchi passivo***

* ***Denial of Service (DOS)*** *L’attacco DOS punta a sovraccaricare il dispositivo, rendendolo* ***inservibile****. Il dispositivo può essere sovraccaricato inviando* ***un numero di richieste eccessivo (ping flooding).*** *Oppure l’attacco può inviare pacchetti con mittente falso corrispondente alla macchina sotto attacco, causando un picco di traffico.*
* ***Spoofing*** *Consiste nel produrre dei pacchetti che sembrano provenire da un altro mittente. Possono essere alterati gli indirizzi IP o anche l’indirizzo MAC*

***Attacchi attivi***

***Man in the middle (MITM)***

*È in realtà una forma di* ***spoofing****: l’attaccante si interpone tra due**interlocutori, intercettando la comunicazione e alterandola****.*** *Tutti questi tipi di attacchi posso arrivare all’installazione di software*

*dannoso: virus o altro.*

***Sfruttamento delle debolezze (exploit)***

*Oppure l’attacco può rivolgersi a debolezze nella gestione del protocollo di comunicazione (ad esempio pacchetti fuori standard possono causare guasti)*

*Molto spesso la ragione è una leggerezza nella programmazione:*

*una situazione di errore creata ad arte produce un danno.*

***Il buffer overflow*** *è un errore di programmazione che, se non rilevato e neutralizzato, è sfruttabile, la zona di memoria destinata ad un pacchetto è limitata, un pacchetto*

*troppo lungo gestito male può addirittura mandare in esecuzione un virus, o aprire una falla.*

***Protezione dagli attacchi: firewall***

*(E’ un componente di difesa perimetrale di una* [*rete informatica*](https://it.wikipedia.org/wiki/Rete_informatica)*, originariamente passivo, che può anche svolgere funzioni di collegamento tra due o più segmenti di rete, fornendo dunque una protezione in termini di* [*sicurezza informatica*](https://it.wikipedia.org/wiki/Sicurezza_informatica) *della rete stessa.*[*[3]*](https://it.wikipedia.org/wiki/Firewall#cite_note-:2-3)[*[4]*](https://it.wikipedia.org/wiki/Firewall#cite_note-:3-4)

*Di norma, la rete viene divisa in due sottoreti: una, detta esterna, è tipicamente una WAN (*[*Wide Area Network*](https://it.wikipedia.org/wiki/Wide_Area_Network)*) che può comprendere* [*Internet*](https://it.wikipedia.org/wiki/Internet)*, mentre l'altra interna, detta LAN (*[*Local Area Network*](https://it.wikipedia.org/wiki/Local_Area_Network)*), comprende una sezione più o meno grande di un insieme di* [*computer*](https://it.wikipedia.org/wiki/Computer)[*host*](https://it.wikipedia.org/wiki/Host) *locali.*[*[5]*](https://it.wikipedia.org/wiki/Firewall#cite_note-5)

*In alcuni casi può essere utile creare una terza sottorete, detta* [*DMZ*](https://it.wikipedia.org/wiki/Demilitarized_zone) *(o zona demilitarizzata), adatta a contenere quei sistemi che devono essere isolati dalla rete interna, ma che devono comunque essere protetti dal firewall ed essere raggiungibili dall'esterno (*[*server*](https://it.wikipedia.org/wiki/Server) *pubblici) )*

*Un* ***firewall*** *viene generalmente predisposto all’ingresso della rete da proteggere.*

*Sui pacchetti dall’esterno verso l’interno, il firewall può essere configurato in modo che solo certi host siano raggiungibili su certe porte (in modo da riservarli a certi servizi, e proteggerli da altri attacchi).*

*I servizi raggiungibili dall’esterno (ad esempio un web server) vengono anche detti* ***zona smilitarizzata*** *(DMZ), ad indicare il fatto che fanno da cuscinetto tra l’esterno (Internet) e l’interno (la rete aziendale).*

*Sui pacchetti dall’interno verso l’esterno, il firewall può decidere di bloccare le connessioni dirette verso siti considerati pericolosi o inappropriati.*

***I pacchetti vengono filtrati principalmente (ma non solo) sulla base degli indirizzi IP, delle porte, e delle interfacce di rete utilizzate****.*

***Chiavi di cifratura (encryption)***

*Lo strumento primario per la sicurezza dei contenuti in Internet sono le chiavi di cifratura.*

*Le chiavi vengono utilizzate per trasformare il contenuto originario in modo che sia incomprensibile a chi non possiede la chiave. Inoltre le chiavi possono essere utilizzate anche per autenticare la provenienza di un contenuto, o di un interlocutore. Esistono due tipi di chiavi, entrambi di largo uso****: chiavi***

***simmetriche e chiavi asimmetriche***

***Chiavi simmetriche***

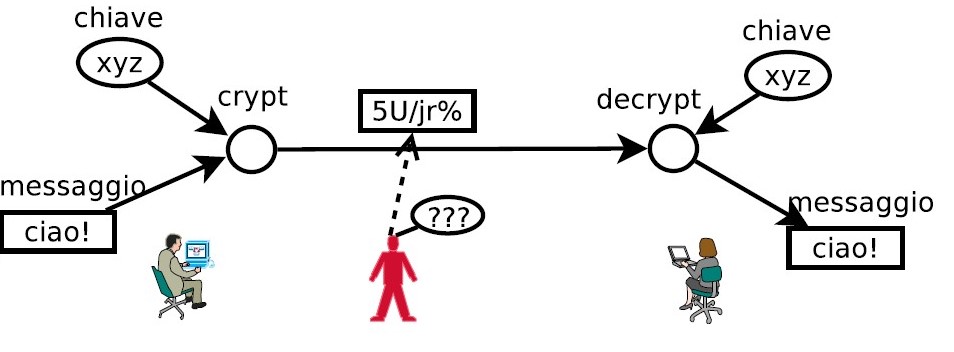
*La chiave simmetrica è una chiave conosciuta agli interlocutori: viene usata sia per cifrare che per decifrare*

*Il principale difetto della chiave simmetrica è che deve essere comunicata, e la comunicazione può essere spiata*

*Se spedisco la chiave via mail...*

*Per avere una chiave simmetrica moderatamente sicura sono necessari 8 byte (64 bit)*

***Proprietà delle funzioni di cifratura simmetriche***

**

*Funzione di cifratura:*

***f (messaggio; chiave) = cifrato***

*Funzione di decifratura*

***g(cifrato; chiave) = messaggio***

*Conoscendo un messaggio ed il cifrato corrispondente è complicato risalire alla chiave.*

***Crittografia asimmetrica (1976)***

*La crittografia asimmetrica è composta da una coppia di chiavi legate da una formula matematica:*

* *una chiave* ***privata****, nota ad un solo utente*
* *una chiave* ***pubblica****, prodotta usando la chiave privata*

*La chiave pubblica è capace di cifrare il messaggio.*

*Solo chi ha la chiave privata può decifrare un messaggio cifrato con la chiave pubblica corrispondente.*

*..ma si può anche cifrare un messaggio con la chiave privata, e sarà decifrabile solo con la pubblica*

*Quindi la crittografia asimmetrica non richiede lo scambio di segreti attraverso altri canali*

*...la chiave pubblica non è segreta.*

*Per essere moderatamente sicura, la chiave pubblica deve essere lunga almeno 128 Byte (1024 bit), contro gli 8 della chiave simmetrica.*

*In genere viene usata una chiave pubblica di 2048 bit.*

***Proprietà delle funzioni di cifratura asimmetriche***

*Funzione di generazione della coppia di chiavi*

***k(chiave privata) = chiave pubblica***

*Il procedimento inverso è troppo costoso, pensate a far girare una chiave spingendo il ferro di una serratura*

*Procedimento di cifratura:*

***f1(messaggio; chiave pubblica) = cifrato***

***g1(cifrato; chiave privata) = messaggio***

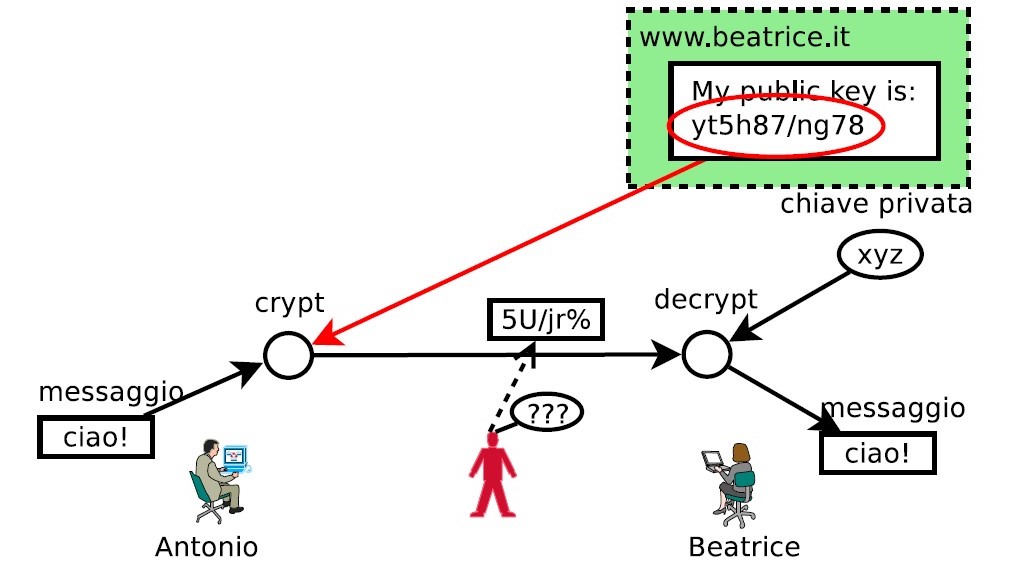
*Procedimento di autenticazione*

***f2(messaggio; chiave privata) = cifrato***

***g2(cifrato; chiave pubblica) = messaggio***

*Procedimento di firma digitale (che usa g2):*

***s (messaggio; cifrato; chiave pubblica) = {vero, falso};***

******

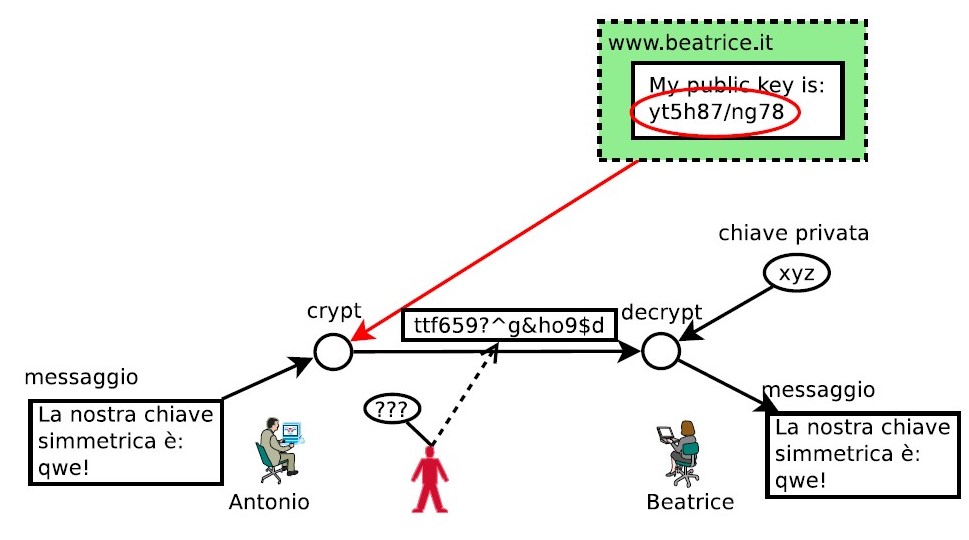
***Creazione di una chiave simmetrica sicura***

*È possibile per due partner X e Y concordare una* ***chiave simmetrica S comune, il segreto****,* ***utilizzando chiavi asimmetriche***

1. *Nel primo passaggio X rende nota la propria chiave pubblica a Y* 
   1. *la chiave può essere contenuta in un certificato digitale, che vedremo dopo.*
2. *Nel secondo passaggio Y crea una chiave simmetrica S, e la spedisce a X cifrata con la chiave pubblica di X (conf1)*
3. *Per finire X utilizza la propria chiave privata per decifrare la chiave simmetrica S (con g1)*

*Ora la chiave* ***S è nota ad entrambi****, ed è stata trasferita in modo che solo il destinatario ha potuto decifrarla*

*La chiave simmetrica è notevolmente più corta della chiave pubblica, e il suo uso è molto più efficiente.*

******

***La firma digitale***

*La crittografia asimmetrica può essere utilizzata anche per* ***firmare*** *(o autenticare) un messaggio.*

*Usando le funzioni f2 e g2 un messaggio cifrato usando la chiave privata può essere decifrato solo usando la chiave pubblica.*

***In questo caso il contenuto del messaggio è pubblico****: cioè, può essere decifrato con la chiave pubblica*

*La proprietà importante è che, conoscendo:*

*il messaggio*

*la cifratura con il chiave privata (con f2)*

***...chiunque (usando g2) può accertare che l’operazione di cifratura è stata operata da chi possiede la chiave privata.***

*Quindi la presenza della cifratura vale come una firma: visto ed approvato.*

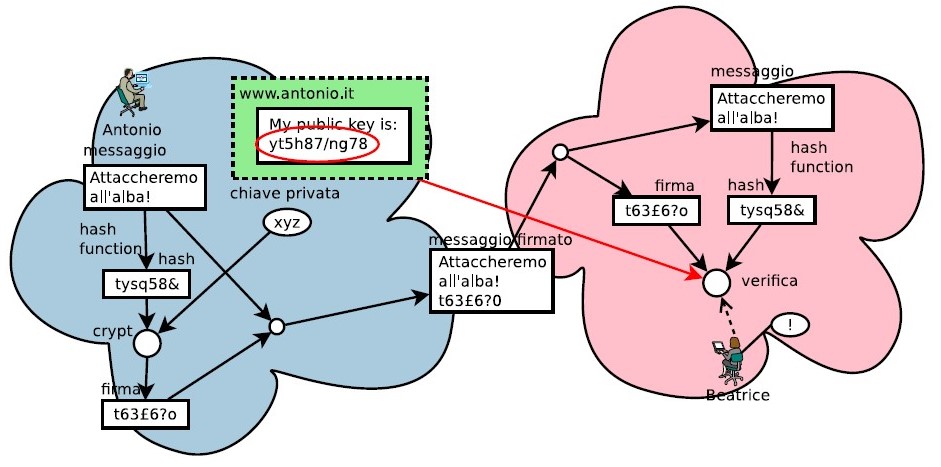
***La funzione di hashing***

*Per autenticare un messaggio di solito gli si allega la cifratura (la firma) di una sua versione estremamente compressa (detta hash).*

*...un documento in pdf di qualche MByte viene compresso in un hash di pochi Kbyte.*

*Se il mittente e il destinatario condividono la funzione di hashing:*

* *il mittente usa la funzione di* ***hashing*** *sul messaggio e produce la hash*
* *il mittente cifra la* ***hash*** *usando la propria chiave privata, e produce la firma del messaggio*
* *il mittente invia insieme il messaggio e la firma*
* *il destinatario produce la hash del messaggio ricevuto*
* *il destinatario verifica se la hash e la firma sono compatibili con la chiave pubblica del mittente*

**

***Il certificato digitale***

*Per il funzionamento affidabile della autenticazione è necessario* ***accertare la provenienza della chiave pubblica****.*

*Per questo si usa un certificato digitale: è un documento pubblico che associa una chiave pubblica ad una*

*certa entità amministrativa, l’entità può essere descritta con la sua ragione sociale, il codice fiscale, l’indirizzo e quant’altro.*

*Per garantire l’autenticità dell’associazione, il certificato contiene la firma generata da una entità di* ***certificazione autorevole*** *(Certification Authority, CA) cioè****, l’hash del documento****, crittografato con la chiave privata*

*della CA.*

***La chiave pubblica della CA è nota, la chiave privata è blindata.***

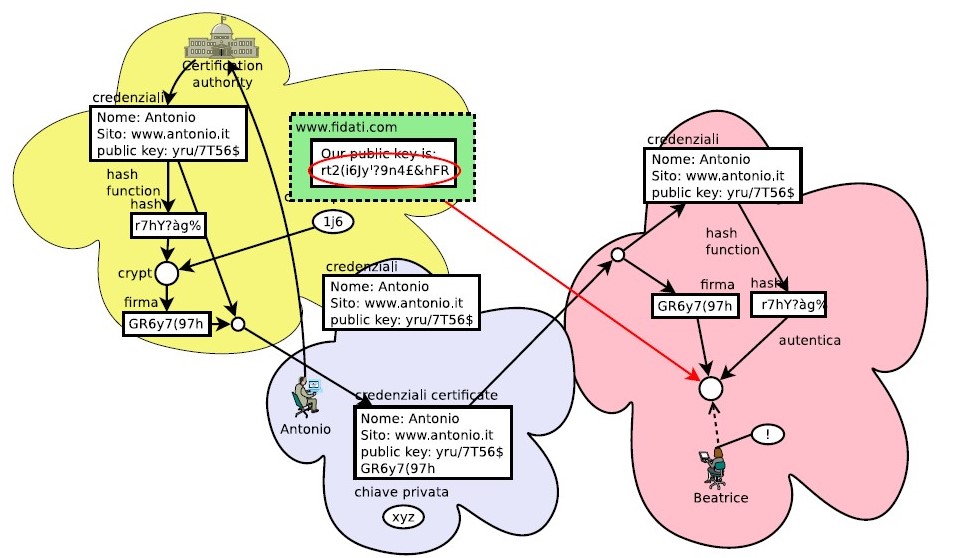
***Proprietà di un certificato digitale***

*Falsificare il certificato è impossibile senza conoscere la chiave privata della CA.*

*Le CA offrono il servizio di generazione del certificato solitamente a pagamento, e devono garantire assoluta affidabilità.*

*Il guasto della CA non rende inutilizzabili i certificati. La compromissione della segretezza della chiave privata della CA*

*rende inefficaci i certificati*

**

***Applicazioni: il protocollo TLS (Transport Layer Security)***

*Quando Internet era disponibile ad un piccolo gruppo di persone, si usava direttamente il protocollo TCP per lo scambio di flussi bidirezionali.*

*Con la crescita di Internet le informazioni trasferite sono diventate più delicate, mentre la presenza di malintenzionati inevitabile.*

*La comunicazione tramite TCP avviene "in chiaro": quindi le informazioni trasferite non sono confidenziali (vedi wireshark).*

*Quindi per molte applicazioni è necessario cifrare le comunicazioni.*

***Il protocollo TLS***

*Il protocollo TLS (successore di SSL)* ***introduce la cifratura della connessione TCP.***

*In apertura della connessione viene scambiato un certificato per:*

* *certificare l’identità del server*
* *concordare la chiave simmetrica segreta*

***Ciascun pacchetto viene poi cifrato con la chiave simmetrica****.*

*Il protocollo TLS è utilizzato per rendere sicuri altri protocolli, come HTTP e FTP.*

***Sviluppo del protocollo TLS***

***negoziazione****: client e server si accordano per determinare versione del protocollo, modi di cifratura eccetera*

*il* ***server*** *comunica il proprio certificato: il pacchetto contiene la chiave pubblica del server*

*il* ***cliente*** *crea una propria chiave preliminare (pre-master) e la invia al server, cifrata con la chiave pubblica del server*

*ora ambedue conoscono la chiave pre-master, che usano per* ***generare*** *le chiavi usate per cifrare e verificare la successiva comunicazione.*

***Accesso ad una macchina remota con chiavi asimmetriche***

*Siamo abituati ad accedere ad un computer* ***utilizzando username e password***

*Tuttavia questa modalità non è sicura quando il computer viene acceduto tramite Internet*

* *lo scambio di pacchetti contenenti le credenziali può essere intercettato*

*Una modalità alternativa, predominante quando si usano* ***macchine remote utilizza una coppia di chiavi asimmetriche.***

*Sulla macchina remota è in funzione un Server SSH a cui l’utente accede tramite un Client SSH.*

***Uso della coppia di chiavi in SSH***

*Il cliente ha (altrimenti crea) una propria coppia di chiavi pubblica/privata*

*Sulla macchina remota viene copiata la chiave pubblica del cliente che vuole accedere*

*Al momento di accedere, il cliente invia la propria identità firmata usando la chiave privata*

*Il server verifica il documento usando la chiave pubblica, ed invia una chiave simmetrica cifrata con la chiave pubblica del cliente.*

*Il resto della comunicazione viene gestito con la chiave simmetrica*

*Lo schema è quello già visto prima.*

***Cloud computing***

***Cloud computing significa utilizzare risorse di calcolo remote in modo semplice***

***Realizza un obiettivo antico: la condivisione delle risorse***

*Lo sviluppo di tecnologie di comunicazione avanzate la rende possibile*

*La condivisione delle risorse incontra l’interesse di:*

* *Le imprese che hanno a disposizione risorse e infrastrutture solo parzialmente utilizzate*
* *utenti che possono utilizzare risorse senza necessità di investire in logistica e manutenzione*
* *fornitori di servizi che possono realizzarli senza necessariamente acquisire l’infrastruttura*

*Contrario di cloud: on premises*

***Private e Public Cloud***

*Le risorse possono essere cedute a vantaggio della propria o di altre organizzazioni. Una distinzione molto significativa, non solo dal punto di vista amministrativo*

***Nel private*** *cloud l’utente è anche proprietario dell’infrastruttura*

* *I l’utente ottimizza, realizzando risorse specifiche utilizzando risorse grezze*
* *I l’utente adatta rapidamente le risorse ai cambiamenti organizzativi*
* *I l’utente adegua l’infrastruttura senza riorganizzare le risorse offerte*

***Nel public*** *cloud l’utente non è il proprietario dell’infrastruttura*

*(fornitore)*

* *I l’utente delega al fornitore la gestione dell’infrastruttura*
  + *come gli aggiornamenti software, o la riparazione dei guasti*
* *I il fornitore si impegna a fornire servizi di un livello determinato contro un corrispettivo economico*

***Il cloud e la virtualizzazione***

*Esiste una fortissima relazione tra tecnologie di virtualizzazione e cloud*

* *I Il meccanismo che sta alla base della fornitura di una risorsa è la sua virtualizzazione*
* *I Tramite la virtualizzazione sono in grado di fornire servizi a richiesta utilizzando risorse grezze*

*La virtualizzazione interessa tutte le risorse di calcolo:*

1. *il processore, può essere virtualizzato con tecnologie del genere hypervisor supportate dall’hardware*
2. *la memoria può essere virtualizzata realizzando hard-disk virtuali*
3. *l’interconnessione può essere virtualizzata utilizzando link virtuali*

*In laboratorio abbiamo usato tutti questi tipi di virtualizzazione per il nostro private cloud*

***Il cloud e i web services***

*I web services incontrano la richiesta di semplicità del cloud. Un web service offre una interfaccia grafica (realizzata nel browser) per accedere al controllo delle risorse*

*Lo sviluppo di modalità di comunicazione attraverso l’infrastruttura http (lo web) fornisce gli strumenti per realizzarlo*

*ad esempio XMPP e WebSocket*

*Quindi la risorsa si presenta attraverso un servizio implementato nel web*

*L’acronimo diffuso per la tecnologia cloud è \*aaS (anything as a service)*

*I principali generi di cloud sono SaaS, PaaS, IaaS*

***SaaS: Software as a Service***

*Si tratta di una applicazione accessibile tramite una interfaccia web*

*Ad esempio un servizio di webmail, agenda, software repository...*

*Il servizio può eventualmente manipolare dati persistenti, che restano nell’ambiente offerto dal fornitore*

*All’utente vengono offerti anche gli strumenti per garantire un accesso esclusivo*

*Molto spesso questo genere di servizio è a sua volta utente di un servizio di cloud*

*Ad esempio, Google Mail, BitBucket.*

***PaaS: Platform as a Service***

*Viene offerta una piattaforma di calcolo completa di sistema operativo: l’utente è in grado di configurarla aggiungendo nuove funzionalità*

*Di solito l’utente ha a disposizione uno storage dove tenere dati o risultati intermedi*

*L’utente può avere la possibilità di produrre una interfaccia alle funzionalità che realizza*

*Ad esempio, Google App Engine, OpenShift, Heroku*

***IaaS: Infrastructure as a Service***

*Vengono offerti i componenti di una infrastruttura di calcolo, compute, storage e network*

*L’utente può configurarne le capacità e l’interconnessione*

*I componenti possono essere accessibili con un indirizzo IP pubblico, o essere vincolati in un’intranet*

*Le capacità di calcolo possono crescere automaticamente in risposta a carichi variabili*

*Ad esempio Amazon EC2, OpenStack*

***Il punto di vista di chi possiede una infrastruttura***

*Si tratta di imprese che posseggono una importante infrastruttura*

* *I ad esempio Amazon, o Google*
* *I anche Grid per la ricerca scientifica (European Grid Infrastructure, EGI)*

*L’uso di questa può essere poco efficiente*

*Un fattore di scala può rendere attraenti ulteriori investimenti in nfrastruttura*

*Le tecniche di virtualizzazione rendono possibile la produzione di risorse a richiesta*

*Utilizzare l’infrastruttura per offrire servizi a pagamento risulta vantaggioso*

*La gestione è più efficiente anche da un punto di vista energetico (lo hype è Green Cloud)*

***Il punto di vista di chi implementa nuovi servizi***

*Il problema di chi offre servizi è gestire l’infrastruttura necessaria a realizzarli o svilupparli*

*ad esempio, un istituto bancario, o un supermercato, problemi sono di varia natura:*

* *I Logistica - una infrastruttura ha bisogno di locali attrezzati ...*
* *I Amministrazione - riparazioni, aggiornamenti ...*
* *I Scalabilità - fare fronte a un incremento di attività senza tempi di ammortamento lunghi*
* *Problemi di sicurezza*
* *Localizzazione dei dati e delle elaborazioni*

***Il punto di vista dell’utente***

*L’utente cerca soprattutto un supporto affidabile ed economico*

*Possibilità di accrescere rapidamente le capacità di calcolo, e ridurle successivamente*

*Accessibilità da piattaforme/dispositivi diversi*

*Distinguere una parte di cloud locale, con risorse proprie, ed una esterna*

*Private/public cloud possono integrarsi in un hybrid cloud: uso le risorse pubbliche quando esaurisco quelle private: ottimizza in presenza di un carico fisso (ad es. gestione dei servizi) ed uno variabile (ad es. risorse destinate ai clienti).*

***Esempi di SaaS: Google***

*Comunicazione e social: Gmail, GoogleTalk, Google+*

*Storage: Google Drive*

*Suite office: presentazioni, foglio di calcolo, testi*

*CMS: Google Sites*

*Geographic Information System (GIS): Google maps*

***Esempi di SaaS: Google***

*Comunicazione e social: Gmail, GoogleTalk, Google+*

*Storage: Google Drive*

*Suite office: presentazioni, foglio di calcolo, testi*

*CMS: Google Sites*

*Geographic Information System (GIS): Google maps*

***Esempi di IaaS: Amazon AWS***

*Elastic Compute: si tratta di nodi virtuali di varie capacità Equipaggiati con vari sistemi operativi*

*Elastic Block: memoria persistente (come HD)*

1. *localizzabile, replicabile, condivisibile ecc.*

*Virtual Private Cloud: organizzare i nodi in una rete compresa la creazione di sottoreti, routing, servizi interni ecc.*

*Auto scaling: configura la capacità di incrementare le risorse sui picchi di carico in modo automatico*

*Service Level Agreement (SLA): garanzie sulla funzionalità degli host*

***Esempi di software per il private cloud***

*Eucalyptus*

* *Le risorse compute, storage, network elastiche*
* *L’interfaccia con public cloud AWS (cloud ibrida)*

*OpenNebula*

* *I Indicato soprattutto per la gestione di dati*
* *I Attenzione alla gestione degli utenti e riservatezza*

*OpenStack*

* *I Open source: il codice è di pubblico dominio*
* *I Sistema controllato tramite un web app dashboard*

*OpenShift*

* *I Open Source*
* *I Sistema di PaaS per i private cloud*

***Interazione con il cloud: il problema dell’interfaccia***

*Il modo in cui l’utente interagisce con la gestione del cloud è determinante per l’interoperabilità*

*È ancora una questione chiave: l’utente di un servizio di cloud teme il lock in più di chiunque altro*

* *lock in, restare intrappolato in una tecnologia che mi lega ad uno specifico fornitore*

*L’esistenza di uno standard per l’interazione con il cloud manager fornisce la base per poter utilizzare la stessa interfaccia per più fornitori*

*Esiste una proposta di standard open (OCCI - Open Cloud Computing Interface) ma esistono pure un certo numero di*

*iniziative chiuse (ad es. DMTF)*

***Lo standard OCCI***

*Lo standard propone un modo di interagire con il componente dell’infrastruttura cloud che controlla le risorse*

*Specifica in modo semplice le interazioni con questo componente*

*Lo standard si basa sul paradigma REST*

*Le risorse nel cloud sono rappresentate come risorse HTTP, su cui è possibile operare con i verb di http*

*I PUT: creazione di una risorsa*

*I GET: ottiene la rappresentazione della risorsa*

*I POST: altera la configurazione della risorsa*

*I DELETE: rimuove la risorsa*

***Il cloud dei dati: servizi NoSQL***

*In breve, le caratteristiche di un database NoSQL:*

* *I Un database è composto da collezioni:*

*ad es., uno per l’anagrafica degli impiegati, uno per gli uffici*

* *I Ogni collezione è composta da documenti, ad esempio per descrivere un certo impiegato*
* *I Un documento è descritto come un dato JSON*

*Ad esempio {nome:"Pio”, cognome:"Po”, livello:4}*

***Interesse per i database NoSQL***

*I DB NoSQL sono particolarmente apprezzati per applicazioni che coinvolgono grandi quantità di dati (big data)*

*...ma non solo: sono estremamente più semplici da usare*

*Alcune delle ragioni*

* *Scalabilità: un DB NoSQL riesce a gestire più informazioni di un relazionale*
* *I Fuzzy query: le richieste approssimative sono gestite in modo efficiente*
* *I Eterogeneità: i documenti in una collezione non devono necessariamente avere tutti lo stesso formato*
* *I Agilità: database non legato al formato dei documenti*

***MongoDB e mlab.com***

*MongoDB deriva da humongous, enorme: è il nome di un server NoSQL (open source)*

*Utilizziamo un fornitore che mette a disposizione un database*

*MongoDB: http://mlab.com*

*Creiamo un nuovo utente e facciamo login*

*Accanto a "MongoDB deployments" selezioniamo Create new*

*Selezioniamo il Cloud Provider (quello che preferite, ad es. AWS)*

*Selezioniamo il Plan type (SANDBOX)*

*I avete a disposizione 500Mb di dati gratis*

*Nella schermata successiva scegliete una regione vicina (Irlanda*

*per AWS)*

*Nella schermata successiva definite il nome del DB (ad esempio*

*"example")*

*Inviate l’ordine (submit) e dopo una decina di secondi il DB è*

*disponibile*

***Accesso al database***

*È necessario definire almeno un utente per il nuovo database:*

* *I nella schermata del database selezionare la linguetta Users*
* *I aggiungete un nuovo utente con Add database user*
* *Il definire username e password*

*I dati in No-SQL sono organizzati in collezioni (collections)*

*Torniamo sulla linguetta Collections e selezionare Add collections e creiamo una collezione dati*

***Abilitazione interfaccia REST***

*In alto a destra cliccate sul vostro user (non sull’account)*

*Scorrete la finestra verso il basso per trovare API Key*

*Abilitate l’accesso tramite API (tasto in basso)*

*Annotate la API key (generatela se necessario)*

*Create una nuova finestra nel browser e visitate la URL*

*(sostituendo la vostra API key):*

*https://api.mlab.com/api/1/databases?apiKey=XXXX*

*Sono tutti i vostri database (uno, example)*

*Poi, le collezioni nel database example (cioè dati)*

*https://api.mlab.com/api/1/databases/example/collections?apiKey=XXXX*

*E infine i documenti nella collezione dati*

*https:// ... /databases/example/collections/dati?apiKey=XXXX*

***Usare una API REST da programma***

*La caratteristica importante di una API REST è che è possibile*

*utilizzarla da programma*

* *quindi, senza accedere tramite browser*

*Il programma Python seguente accede al server e restituisce il dato (JSON) corrispondente a tutti i database*

***GET in Python***

***import*** *httplib, json*

*api\_key="la vostra apiKey"*

*conn = httplib.HTTPSConnection("api.mlab.com") # Apro la connessione*

*# Invio la request*

*conn.request(*

*"GET", # metodo HTTP*

*"/api/1/databases?apiKey="+api\_key # HTTP header*

*)*

*response = conn.getresponse()*

***print*** *response.status, response.reason*

***print*** *response.read()*

*conn.close()*

*Provate a modificare il programma per ottenere le collezioni, ed i*

*dati nelle collezioni*

***Aggiungere dati ad una collezione***

*Per aggiungere un nuovo documento nella nostra collezione possiamo usare una POST*

* *la URL corrisponde a quella che abbiamo già visto nell’ultimo esempio*
* *è necessario specificare uno header che indichi la codifica del contenuto application/json*
* *I il corpo del messaggio contiene il documento che si vuole che sia registrato nella collezione*

*Il programma Python seguente utilizza una POST*

***POST in Python***

***import*** *httplib, urllib, json*

*api\_key="la vostra apiKey"*

*# Il documento da inserire come oggetto Python (insieme di coppie chiave􀀀valore)*

*document = {*

*"text":"Hallo!",*

*"author":"augusto"*

*}*

*document\_json=json.JSONEncoder().encode(document) # Lo codifico in JSON*

*conn = httplib.HTTPSConnection("api.mlab.com") # Apro la connessione*

*conn.request(*

*"POST", # metodo HTTP*

*"/api/1/databases/example/collections/dati?apiKey="+api\_key, # URL*

*document\_json, # HTTP body*

*{"Content-type": "application/json"} # HTTP header*

*)*

*response = conn.getresponse()*

***print*** *response.status, response.reason*

***print*** *response.read()*

*conn.close()*

***Query su un database***

*Una query viene richiesta con una GET al solito path*

*Insieme alla apiKey va specificata la query,*

*Nella forma più semplice, per selezionare i documenti con un certo valore in un certo campo*

*I nome:valore, ...*

*Ad esempio, nel browser, possiamo impostare:*

*https:// ... /dati?apiKey=XXXX&q={author:"augusto"}*

*Attenzione: se fate la stessa prova con curl, sostituite { con %7b e }*

*con %7d, e racchiudete la URL tra apici singoli ’*

*I la prima sostituzione è per l’urlencoding*

*I la seconda per proteggere i caratteri dall’interpretazione della shell Unix*