

Lez 1 - Reti e modelli stratificati -13/09/2021

lunedì 13 settembre 2021 11:28

Slide introduzione al corso

RICEVIMENTO: venerdì 14:00 - 16:00

Rete: interconnessione di dispositivi in grado di scambiarsi informazioni, quali sistemi, terminali, router, switch e modem

I sistemi terminali sono chiamati host:

- Macchina dell'utente finale
- Server

Dispositivi di interconnessione:

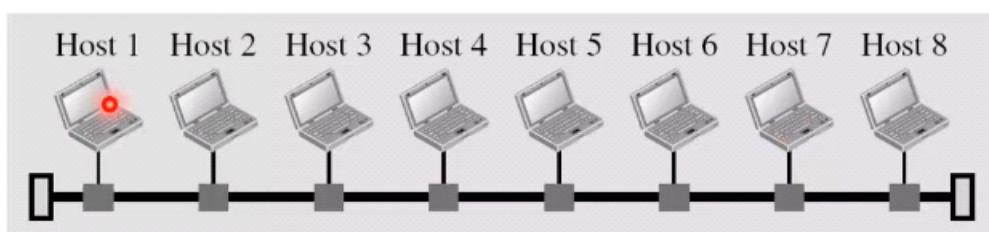
- Router
- Switch

Collegamenti(link): mezzi trasmissivi (cablati, wireless)

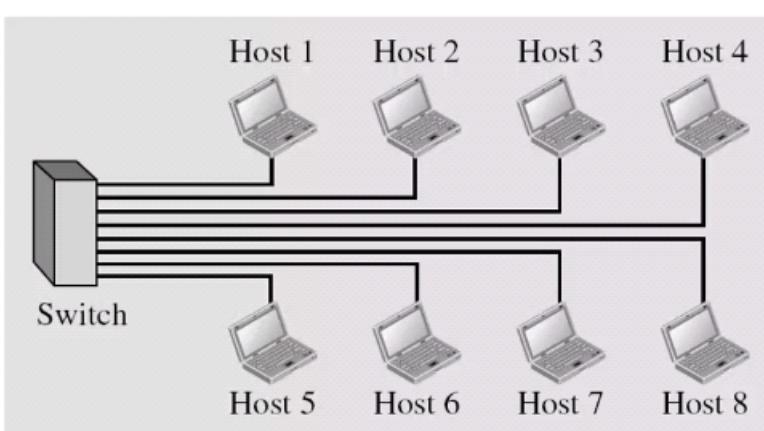
LAN

Reti di computer circoscritte ad un'area limitata, sono di proprietà di un'organizzazione (reti private)
Hanno un'estensione che arriva tipicamente fino a qualche km

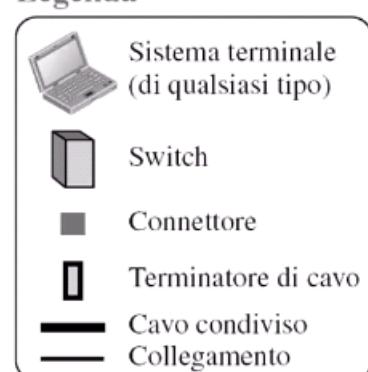
Vengono usate per connettere sistemi terminali personali tramite cavo in rame o wireless
principalmente



a. LAN con cavo condiviso (obsoleta)



Legenda



b. LAN con switch (moderna)

Tecnologia a bus decisamente obsoleta e orrida, adesso si usano degli switch

WAN (Wide Area Network)

Una WAN (o rete geografica) è una rete il cui compito è interconnettere LAN o singoli host separati da distanze geografiche

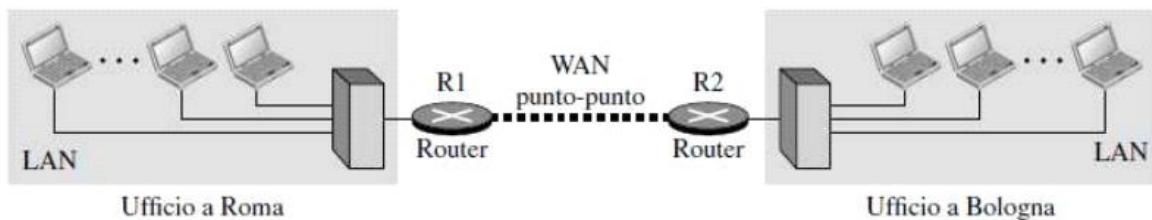
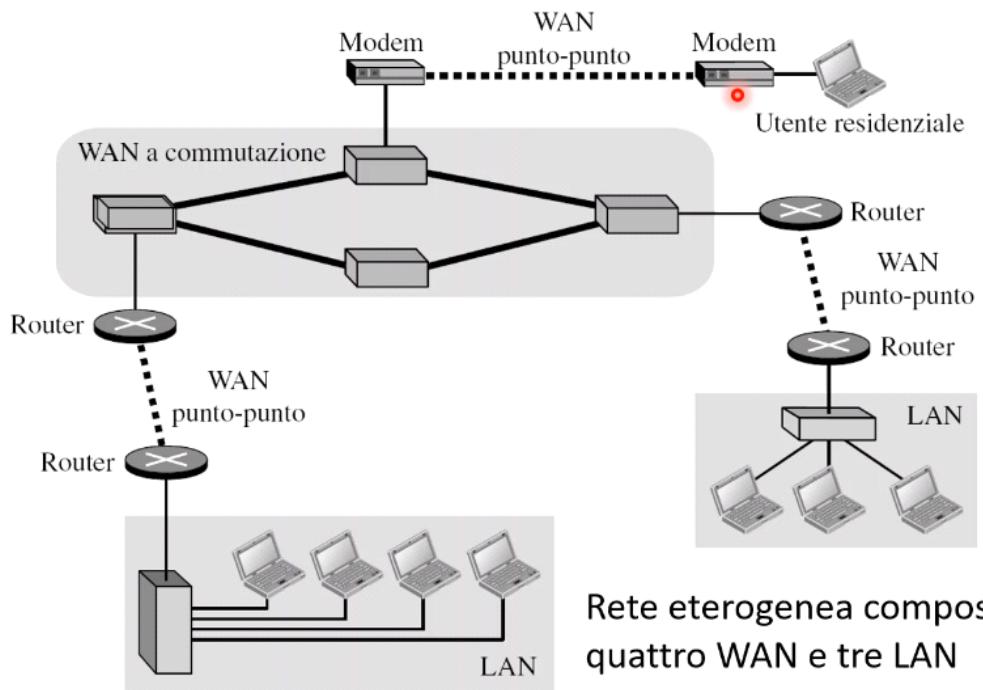
WAN punto-punto: collega due dispositivi tramite un mezzo trasmissivo

WAN a commutazione

Collega più di due punti di terminazione (es dorsali internet)

Elementi di commutazione: elaboratori specializzati utilizzati per connettere fra loro due o più linee di trasmissione.

Si collegano in genere due reti locali (LAN) con una rete WAN punto-punto (es uffici della stessa organizzazione)



Commutazione

Una internet è data dall'interconnessione di reti, composte da link e dispositivi capaci di scambiarsi informazioni

Dispositivi involved:

- Sistemi terminali (host)
- Dispositivi di interconnessione che si trovano nel percorso tra sorgente e destinazione

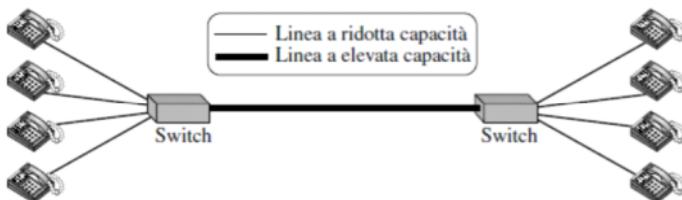
Tecniche di commutazione: (modalità con cui viene determinato il percorso sorgente-destinatario)

- Circuit switched network - reti a commutazione di circuito
- Packet-switched network - reti a commutazione di pacchetto

Commutazione di circuito

- Instaura un cammino dedicato tra i due dispositivi che vogliono comunicare: una serie di linee di trasmissione e nodi di commutazione che collegano sorg. E dest.
- Sul percorso vengono dedicate risorse alla comunicazione (canale logico o circuito)
 - o Banda di frequenza o slot di trasmissione sui collegamenti
 - o Capacità commutative dei nodi
- Le risorse allocate sono garantite per tutta la durata della comunicazione

Vantaggi	<ul style="list-style-type: none"> - Performance (garantita) es. rete telefonica fissa tradizionale
Svantaggi	<ul style="list-style-type: none"> - Necessaria una fase di instaurazione della comunicazione (setup) - Le risorse rimangono inattive se non utilizzate



Circuito dedicato per l'intera durata della connessione

Problema: poca flessibilità nell'utilizzo delle risorse

Commutazione di pacchetto

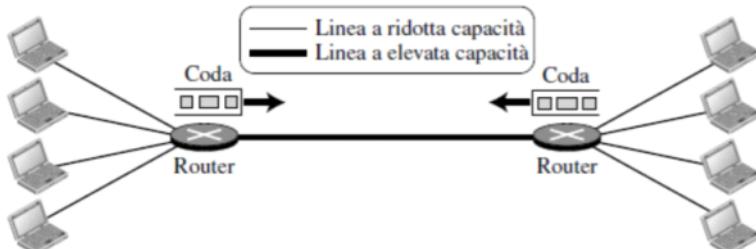
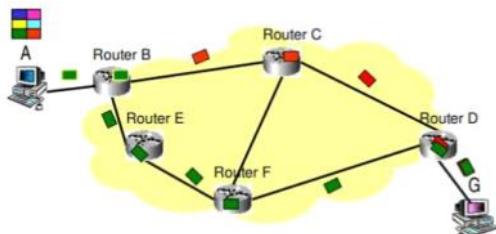
Sfrutta le risorse della rete in modo diverso.

Il flusso di dati punto punto viene suddiviso in pacchetti.

I pacchetti degli utenti A e G condividono le risorse di rete

I pacchetti sono indipendenti l'uno dall'altro della stessa comunicazione, possono seguire lo stesso percorso o percorsi diversi, sia pacchetti della stessa comunicazione che quelle diverse.

Le risorse vengono usate a seconda delle necessità



Il router riceve un pacchetto e lo inoltra sul collegamento a elevata capacità, se il collegamento è pieno i pacchetti vengono messi in un buffer.

L'introduzione del buffer crea congestione e potenziale ritardo nell'arrivo dei pacchetti.

Il buffer ha capacità limitata quindi in casi gravi di congestione vengono persi dei pacchetti

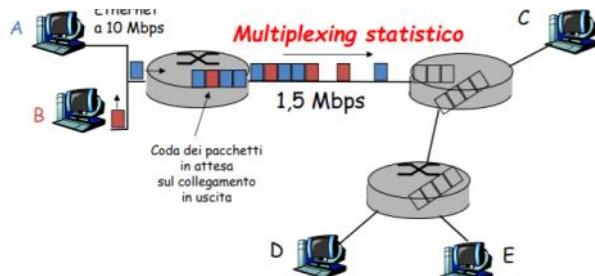
Non c'è un canale dedicato, gli host comunicano scambiandosi pacchetti

I router possono memorizzare i pacchetti nelle code (buffer) se il collegamento tra i due router è usato alla massima capacità

La richiesta di risorse può eccedere il quantitativo disponibile
Congestione: accodamento dei pacchetti, attesa per l'utilizzo del collegamento

Trasmissione store and forward

Un router aspetta che gli arrivi tutto il pacchetto poi lo analizza e lo spedisce
 Aspetta che gli arrivi tutto il pacchetto prima di poter trasmettere-> *ritardo di store and forward*
 Attesa dei pacchetti in code di output (buffer) -> *ritardi di coda*
 I buffer hanno dimensione finita -> *perdita di pacchetti*



In questo caso ho una connessione ethernet a 10 mbps e un collegamento a 1.5, quindi al router arriveranno un tot di pacchetti che non riuscirà a instradare immediatamente, ma dovrà metterli nel buffer.

La sequenza dei pacchetti non segue un ordine prefissato: *condivisione di risorse su richiesta* multiplexing statistico delle risorse)

Circuit vs packet switching

- esempio
 - N=35 utenti condividono un link 1 Mbps
 - Ogni utente genera 100kbps quando è "attivo"
 - Ogni utente è attivo 10% del tempo
- Circuit Switching

Con la commutazione di circuito, occorre riservare 100kbps per ogni utente, in ogni istante. Il link di output può quindi supportare simultaneamente al massimo $1\text{Mbps}/100\text{kbps} = 10$ utenti
- Packet Switching
 - 10 o meno utenti simultaneamente attivi -> banda richiesta ≤ 1 Mbps, nessun ritardo
 - Più di 10 utenti attivi simultaneamente -> frequenza aggregata di arrivo dei dati supera la capacità del collegamento in uscita e quindi si ha ritardo di accodamento
 - N.B. la probabilità che ci siano 10 o meno utenti attivi contemporaneamente è 0.9996
 - con alta probabilità la tecnica del packet switching supporta tutti i 35 utenti senza introdurre alcun ritardo!

25

Circuit Switching

- Vantaggi
 - Prestazioni garantite
 - Overhead limitato
 - Tecnologie di switching efficienti
- Svantaggi
 - E' richiesta la segnalazione per instaurare il circuito (configurare le tabelle di switching)
 - Sottoutilizzo delle risorse in presenza di traffico a raffica (burst) e rate di traffic variabile

Packet Switching

- Vantaggi
 - Risorse trasmissive usate solo se necessario
 - Segnalazione non richiesta
- Svantaggi
 - Overhead
 - Tecnologie di inoltro non efficienti (necessità di selezionare l'uscita per ogni pacchetto)
 - Tempo di elaborazione ai router (routing table lookup)
 - Accodamento ai router

26

Internet

Una internet è costituita da due o più reti interconnesse, la internet più famosa è **Internet**, composta da migliaia di reti interconnesse, ogni rete connessa a Internet deve usare IP (internet protocol) e rispettare certe convenzioni su nomi e indirizzi

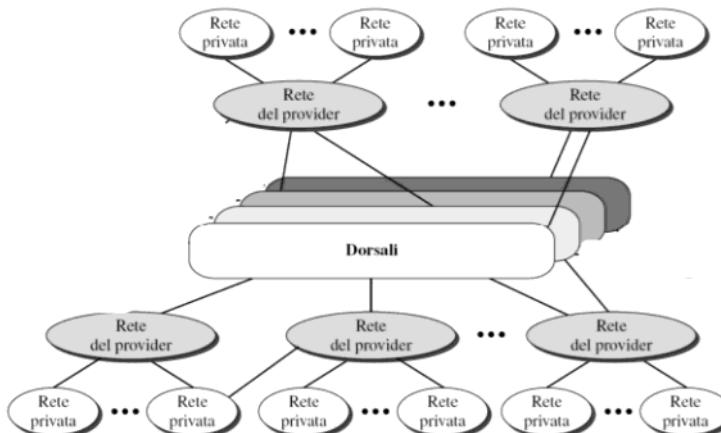
Infrastruttura che fornisce servizi di comunicazione alle applicazioni

I tipi di servizi che possono essere utilizzati sono:

- Senza connessione: senza garanzia di consegna (**connectionless**)
- Orientati alla connessione: garantiti in integrità, completezza e ordine (**connection-oriented**)

Entità software:

Applicazioni, protocolli (regolamentano le trasmissioni), interfacce (come uno strat dello stack si interfaccia con l'altro)



Le reti degli host sono cono connesso a Internet attraverso una gerarchia di ISP le dorsali sono ISP di livello 1 (sono poche e interconnesse) ISP di secondo e terzo livello (regionali e di accesso)

Peering point: accordo tra due ISP di accettare e inoltrare il traffico che ricevono dall'altro

- IXP Internet eXchange Point: punto d'incontro per il peering tra due o più ISP

Reti di accesso

Il collegamento che connette l'utente al primo router di internet è detto rete di accesso:

- Accesso via rete telefonica (dial-up, dsl, fibra ottica)
- Accesso tramite reti wireless
- Collegamento diretto

Metriche

Misura delle prestazioni di rete

- Ampiezza di banda e bitrate
- Throughput
- Latenza
- Perdita di pacchetti

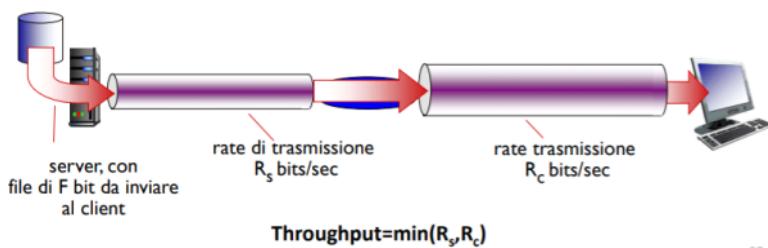
La banda/ampiezza di banda larghezza dell'intervallo di frequenze utilizzato dal sistema trasmissivo. (Hz)

La velocità di trasmissione è la quantità di bit che possono essere trasmessi nell'unità di tempo su un certo collegamento. (bits/secondo o bps)

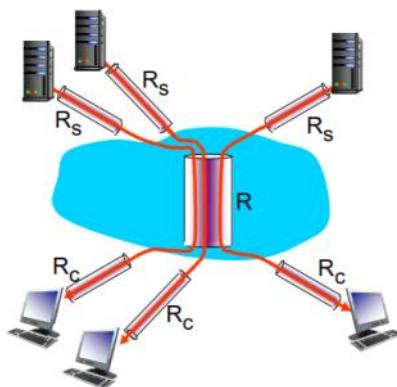
Il bit rate dipende dalla larghezza di banda e dalla tecnica trasmissiva usata

Throughput: quantità di dati utili che possono essere trasmessi tra un nodo sorgente e un nodo destinazione in un certo intervallo di tempo. Indica la velocità con cui trasferiamo i dati, al netto di perdite sulla rete, duplicazioni, protocolli...

Si intende da un nodo terminale all'altro, il bitrate è la capacità trasmissiva di un collegamento, nel mio percorso posso avere più collegamenti con più bitrate diversi.



Non sono solo nel collegamento, il collegamento è condiviso per più comunicazioni



End-to-end throughput per connessione: $\min(R_c, R_s, R / 10)$ (con 10 collegamenti alla rete sopra)

Il throughput di questo collegamento non può essere più grande del minimo tra questi valori.

In rete mando sia i dati del mio file che i dati relativi ai protocolli per viaggiare in internet.

- Latenza: tempo richiesto affinché un messaggio arrivi a destinazione dal momento in cui il primo bit parte dalla sorgente

Latenza = ritardo di propagazione + ritardo di trasmissione + ritardo di accodamento + ritardo di elaborazione

<Lez 3 20/09/2021>

Come si verificano ritardi e perdite?

I pacchetti si accodano nei buffer dei router

Il tasso di arrivo dei pacchetti sul collegamento eccede la capacità del collegamento in uscita di evaderli; i pacchetti si accodano, in attesa del loro turno

- Pacchetti in attesa di essere trasmessi **ritardo**
- Pacchetti accodati **ritardo**
- Buffer liberi (disponibili): se non ci sono buffer liberi i pacchetti in arrivo vengono scartati **perdita**

Cause di ritardo:

Ritardo di elaborazione del nodo	Ritardo di accodamento	Ritardo di trasmissione (L/R)	Ritardo di propagazione (d/s)
<ul style="list-style-type: none"> - Controllo errori sui bit - Determinazione del canale di uscita 	<ul style="list-style-type: none"> - Attesa di trasmissione - Dipende da intensità e tipo di traffico (che influiscono sul numero di pacchetti in coda nel buffer) 	<i>Tempo impiegato per trasmettere un pacchetto su un link.</i> R = rate di trasmissione del collegamento (bps) L = lunghezza del pacchetto (bit)	<i>Tempo impiegato da 1 bit per essere propagato da un nodo all'altro</i> d = lunghezza del collegamento fisico (m) s = velocità di propagazione del mezzo ($3-2 \times 10^8$ m/sec)

Ritardo end-to-end:

N-1 router tra origine e destinazione

Ritardo(nodo) = ritardo(el) + Ritardo (tr) + Ritardo (pr)

Ritardo(end-to-end) = N * Ritardo(nodo)

$$\text{Ritardo}_{\text{nodo}} = \text{Ritardo}_{\text{el}} + \text{Ritardo}_{\text{tr}} + \text{Ritardo}_{\text{pr}}$$

$$\text{Ritardo}_{\text{end-to-end}} = N * \text{Ritardo}_{\text{nodo}}$$

nel caso generale di ritardi eterogenei -> sommatoria

$$\text{Ritardo}_{\text{end-to-end}} = \text{Ritardo}_{\text{nodo1}} + \text{Ritardo}_{\text{nodo2}} + \dots + \text{Ritardo}_{\text{nodoN}}$$

Traceroute: comando che traccia un pacchetto dal tuo computer all'host mostrando il numero di salti per raggiungerlo e al ritardo dal mittente per ogni passaggio

Lez 3 - 20/09/2021

lunedì 20 settembre 2021 11:17

Protocollo

Insieme di regole che permettono a due entità di comunicare.

Il modello usato nei modelli di comunicazioni è un *modello stratificato*:

Stratificare per semplificare sistemi complessi.

La struttura esplicita permette l'identificazione delle relazioni tra gli elementi di un sistema complesso

- Modello di riferimento stratificato
- Suddivisione di funzioni e attori

La modularizzazione facilita la manutenzione e l'aggiornamento dei dati

- Un modulo (**livello/strato**) svolge un insieme delimitato di compiti e nel sistema appare come una black box (input/output)
- Ogni livello offre servizi allo stato superiore
 - o Implementa azioni all'interno del livello stesso
 - o Utilizza servizi del livello inferiore

Separazione tra servizi offerti (interfaccia) e implementazione: il cambiamento dell'implementazione di un servizio in un livello è sostanzialmente trasparente per il resto del sistema.

Principi base:

Separation of Concern: separazione delle responsabilità

Information hiding: nascondere le informazioni che non sono indispensabili per il committente.

- **Modello stratificato**
 - costituito da sistemi di consumatori/produttori
 - sistemi organizzati in strati funzionali (livelli)
 - ogni strato fornisce servizi allo strato superiori e usa i servizi di quello inferiore
 - ogni strato scambia informazioni direttamente solo con gli strati adiacenti
 - in ogni comunicazione i due strati omologhi svolgono funzioni reciproche
 - esistono sistemi (intermedi) che implementano solo alcune funzioni
- **Requisiti**
 - efficiente: minimizzare lo sforzo globale di consegnare le lettere (non la singola!)
 - efficace: consegnare la maggior quota possibile di lettere

Strati omologhi svolgono funzioni reciproche.

I livelli possono evolvere con la tecnologia senza dover cambiare gli altri strati, perché sono indipendenti

Vantaggi del modello stratificato:

- Scomponi il problema in sottoproblemi più semplici da trattare -> il singolo strato è più semplice del sistema nel suo complesso (semplifica progettazione, implementazione e manutenzione del software)
- Rende i vari livelli indipendenti
I servizi forniti dagli strati inferiori possono essere usati da più entità negli strati adiacenti superiori

I livelli diversi possono essere sviluppati da soggetti diversi

Ogni livello logico di astrazione è realizzato in un apposito strato; un livello viene creato quando si rende necessario un diverso grado di astrazione
Ogni strato svolge una sola e ben definita funzione
Il flusso dati attraverso le interfacce di ogni strato deve essere minimizzato
Il numero degli strati deve essere minimizzato compatibilmente con la loro complessità

Modello ISO/OSI. Open system interconnection/international organization for standards

Modello di riferimento per i sistemi di comunicazione in rete

Obiettivo:

Definire uno standard per cui definendo delle regole comuni fosse possibile far parlare qualsiasi dispositivo tra una rete e l'altra.

Definizione di uno standard APERTO:

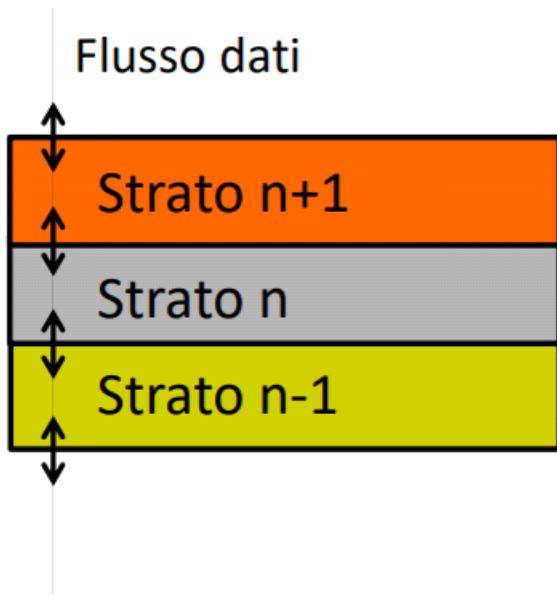
- I dettagli del protocollo sono disponibili pubblicamente
- I cambiamenti sono gestiti da un'organizzazione la cui partecipazione è aperta al pubblico

Un sistema che implementa protocolli aperti è un **sistema aperto**

L'organizzazione che ha definito questo standard è la ISO, che ha definito il modello di riferimento Open Systems Interconnection(OSI). Diventato standard internazionale nel 1983.

Modello stratificato, ogni strato ha una propria funzionalità.

La comunicazione tra i vari livelli è assicurata da chiamate standard; ogni livello è tenuto a rispondere in maniera corretta alle chiamate che gli competono e che verranno generate dai due livelli a esso adiacenti



Strato n offre un servizio a n+1 e chiede un servizio a n-1. Il flusso dati va in verticale. Lo strato n-esimo di una entità comunica con lo strato ennesimo dell'altra entità (es destinazione)

Strato/livello:

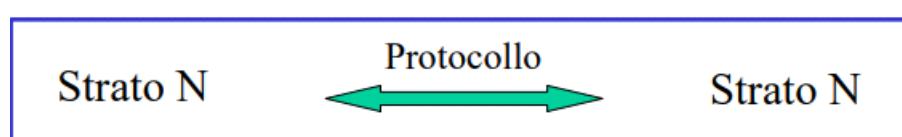
Modulo definito attraverso i servizi, protocolli e interfacce che lo caratterizzano

Servizio:

Insieme di primitive (operazione che uno strato fornisce ad uno strato soprastante)

Interfaccia:

Insieme di regole che governano il formato e il significato dei dati che vengono scambiati tra *due strati adiacenti della stessa entità*



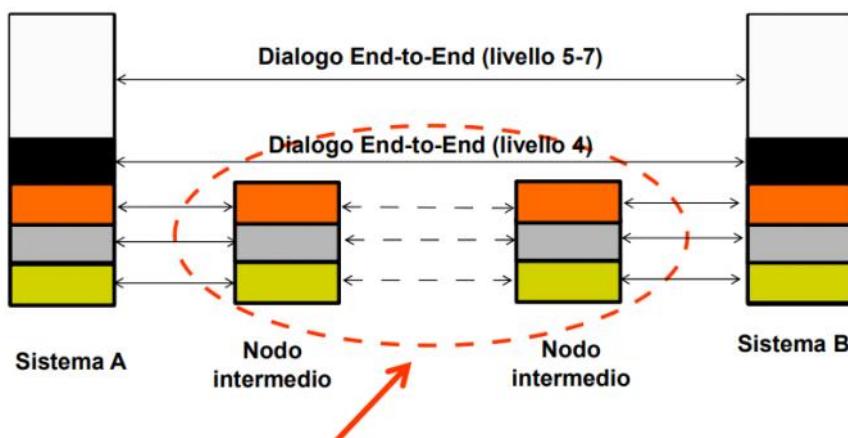
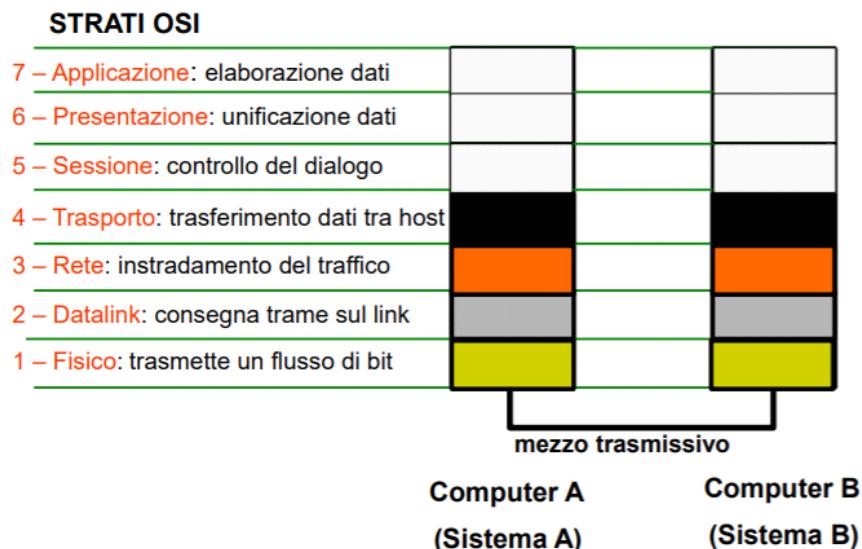
Protocollo:

Insieme di regole che:

- Permettono a due entità omologhe (stesso strato) uno scambio efficace ed efficiente delle informazioni
- Definiscono il formato e l'ordine dei messaggi inviati e ricevuti tra entità omologhe della rete e le azioni che vengono fatte per la trasmissione e ricezione dei messaggi
- *Efficace*: un sistema che riesce a raggiungere lo scopo prefissato con la maggior frequenza possibile
- *Efficiente*: un sistema che riesce a raggiungere lo scopo prefissato con il minor sforzo possibile

Specifica la sintassi, la semantica e le azioni da intraprendere dopo la ricezione di un msg

ISO/OSI è stratificato in 7 livelli



Tipicamente i primi 3 livelli sono intermedi (1-2-3)

Lez 4 - Livello Applicativo 22/09/2021

mercoledì 22 settembre 2021 11:04

Da slide 24 intro 3

I nodi intermedi non implementano tutti gli strati, implementano funzioni chiamate end-to-end.

I servizi trasporto, rete e collegamento possono offrire allo strato adiacente e superiore un servizio *connection oriented e un servizio connectionless*

Connection oriented stabilisce una connessione, instrada i dati e chiude la connessione

Connectionless instrada i dati senza stabilire una connessione.

Flusso dell'informazione

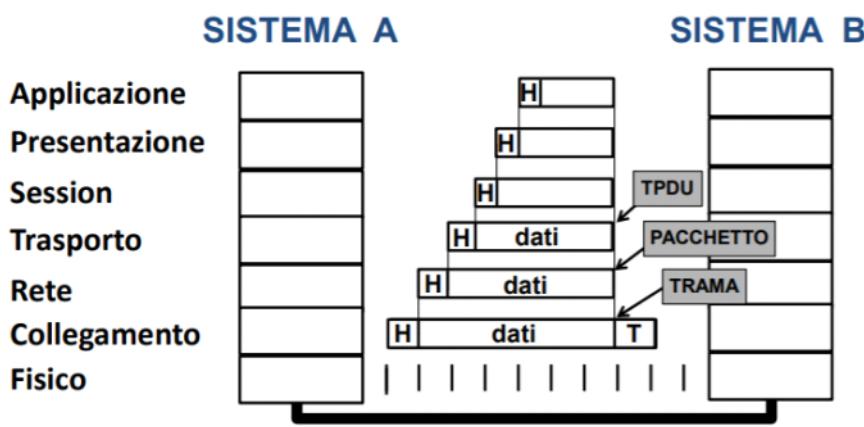
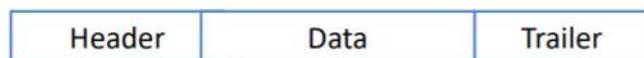
incapsulamento

Tipicamente l'informazione, a livello di dati, ha origine al livello applicativo, l'informazione discende i livelli fino al canale fisico. Ogni livello aggiunge delle informazioni, un'intestazione che racchiude le informazioni di protocollo che servono per implementare i protocolli di quel livello.

Per i dati ricevuti si segue il cammino inverso, si guarda l'intestazione che era stata aggiunta dall'altro nodo a livello omologo

Ogni livello esegue una operazione di incapsulamento sui dati già incapsulati a livello precedente, al nodo opposto vengono de-capsulati per estrarre i dati.

- Header
 - Qualificazione del pacchetto dati per questo livello
- DATA
 - Payload proveniente dal livello superiore
- Trailer
 - generalmente usato in funzione di trattamento dell'errore (rivelazione, correzione)



La TPDU è l'insieme dei dati e header del livello di trasporto

Stack TCP/IP

Il modello TCPI/IP è molto simile, ma su 5 livelli (senza sessione e presentazione)



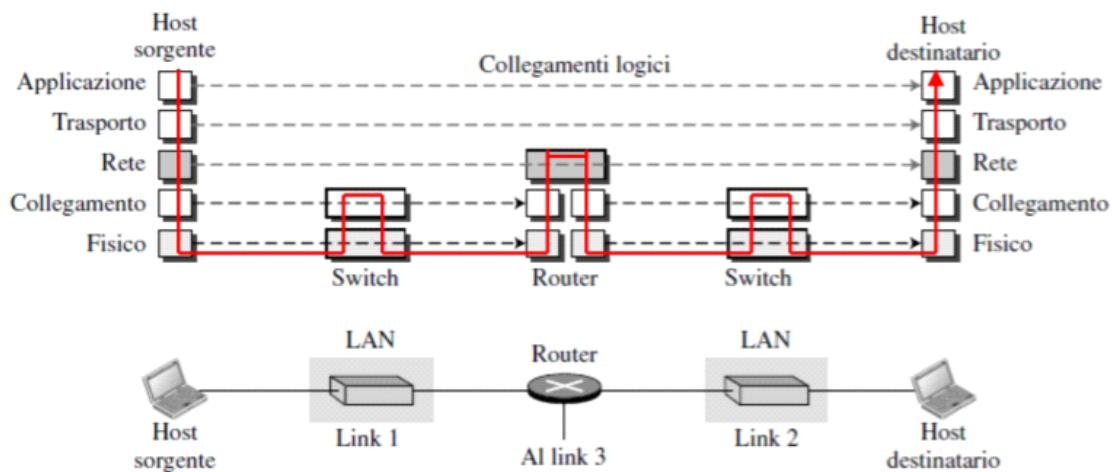
Applicazione: supporta le applicazioni di rete

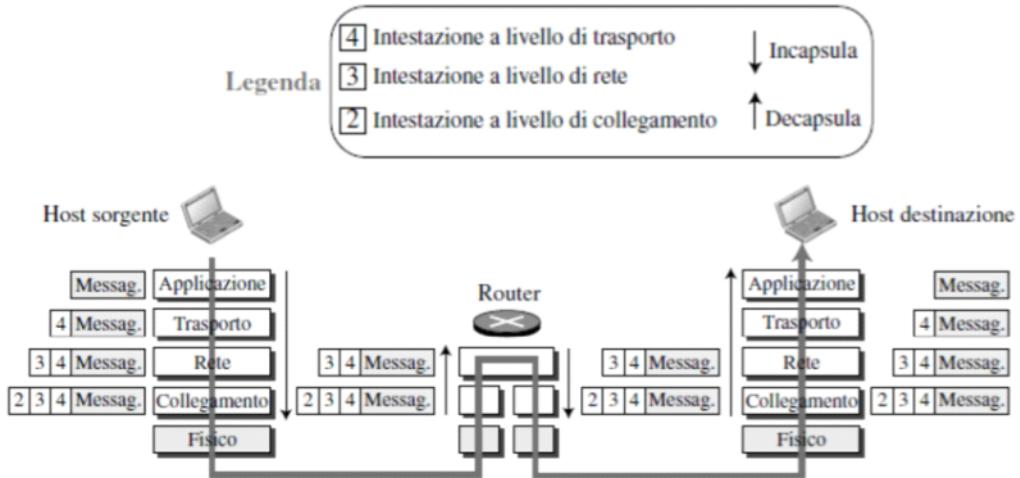
Trasporto: trasferimento dati end-to-end (tcp, udp)

Rete: instradamento dei dati tra host e destinazione (IP), protocolli di routing

Link: trasferimento dei dati in frame attraverso il collegamento che unisce degli elementi di rete vicini (ethernet, ppp, ...)

Fisico: trasferimento die bit tramite il mezzo trasmittivo





ISO/OSI usato come modello di riferimento, TCPI/IP standard de facto

Livello Applicativo URI-HTTP

Abbiamo bisogno di far comunicare processi che stanno su host diversi, abbiamo bisogno di regole per farlo senza essere legati alle tecnologie specifiche dei nostri host

Due o più processi girano su ciascun host e si scambiano messaggi.

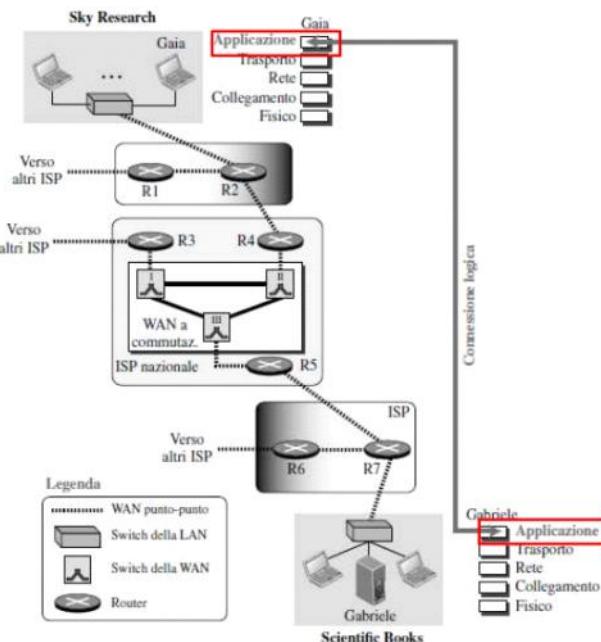
Si ha un collegamento logico tra i due processi.

Il protocollo:

- Definisce i tipi di messaggi scambiati a livello applicazione (es. richiesta, risposta)
- Sintassi dei vari tipi di messaggio
- La semantica dei campi
- Le regole per determinare quando e come un processo invia o risponde ai messaggi

Livello applicazione

I livelli applicazione nei due lati della comunicazione agiscono come se esistesse un collegamento diretto attraverso il quale poter inviare e ricevere messaggi



Ci sono due paradigmi fondamentali:

Client-Server: numero limitato di processi server che offrono un servizio e sono in esecuzione, in attesa di ricevere richieste

Client: programma che richiede un servizio, inizia la comunicazione col server

Peer to Peer: peer che possono offrire servizi e inviare richieste
misto

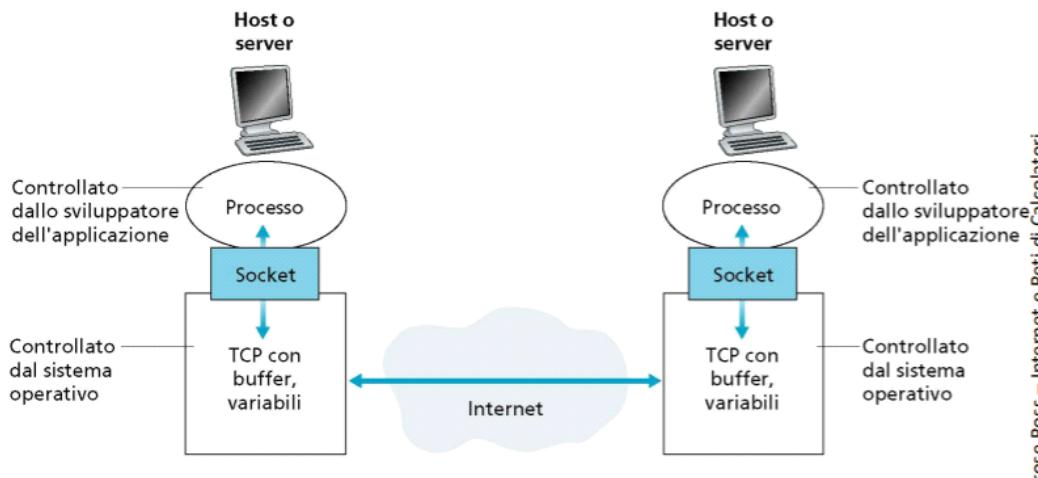
Terminologia

API: application programming interface:

Insieme di regole che un programmatore deve rispettare per utilizzare delle risorse

Interfaccia socket:

API che funge da interfaccia tra gli strati di applicazione e di trasporto.



Quello che fa il processo quando impacchetta il messaggio, vuole che venga recapitato all'altro socket

Interfaccia socket

Il client impacchetta il messaggio e ha un suo identificativo locale di socket, il destinatario sarà la socket corrispondente del processo destinatario.

È una struttura dati che viene utilizzata dal programma applicativo, endpoint della comunicazione. La comunicazione tra un processo client e server è realizzata attraverso la comunicazione tra le due socket

Il client considera la socket come entità che riceve le richieste e fornisce le risposte. (il server pure al contrario)

Per ricevere messaggi, un processo deve avere un identificatore -> IP a 32 bit.

Per identificare il processo non è sufficiente avere solo l'indirizzo IP.

L'identificativo include l'indirizzo IP e il numero di porta associato al processo

Uso dei servizi di trasporto

Una coppia di processi deve utilizzare i servizi offerti dal livello di trasporto per la comunicazione

I principali protocolli offerti da TCP/IP sono:

- UDP
- TCP

Servizio TCP

- Connection-oriented: setup richiesto tra client e server
- Trasporto affidabile tra processo mittente e destinatario
- Controllo del flusso: il mittente non "inonderà" di dati il destinatario
- Controllo di congestione: "strangola" il mittente quando la rete è sovraccarica
- Non offre garanzie di timing né di ampiezza minima di banda

Servizio UDP

- Non orientato alle connessioni
- Trasporto NON affidabile
- NO controllo di flusso
- NO controllo congestione
- No garanzie timing né ampiezza minima di banda

Requisiti più importanti

Throughput: tasso al quale il processo mittente può inviare i bit al processo ricevente, importante per i trasporti multimediali.

Ritardi: nella commutazione di pacchetto abbiamo tante componenti di ritardo, non costante. Lo stack TCP/IP non ci garantisce che i dati vengano consegnati (protocollo best effort)

Ci sono applicazioni che possono tollerare perdite di dati e alcune che non possono. Possono:
streaming, telefonata, non possono: browser

applicazione	Tolleranza alla Perdita di dati	throughput	Sensibilità al tempo
file transfer	no	elastico	no
e-mail	no	elastico	no
Documenti Web	no	elastico	no
Telefonia su Internet/ videoconferenza	si	audio: 5kbps-1Mbps video:10kbps-5Mbps	Si, centinaia di msec
audio/video memorizzato	si	come sopra	Si, pochi secondi
Giochi interattivi	si	Pochi kbps	Si, centinaia di msec
Messaggistica istantanea	no	elastico	Si e no

Mappati i requisiti posso assegnare i protocolli

applicazione	protocollo di livello applicazione	Protocollo di trasporto
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

Che cos'è il WEB?

Ad oggi enorme collezione di informazioni nella quale le risorse sono distribuite e collegate l'una all'altra, tramite gli hyperlink, collegamenti. Qualsiasi server web nel mondo può aggiungere risorse.

Un sito web è un insieme di risorse web.

Pagina web formata da:

- File HTML di base
- Diversi oggetti referenziati (altre pagine, immagini, file)
- Ciascun oggetto è indirizzabile tramite **URL** (uniform resource locator)

URI, URL E URN

URI (Uniform Resource Identifier)

Forma generale per identificare una risorsa. "a uniform resource identifier is a compact string of characters for identifying an abstract or physical resource"

Uniform: uniformità della sintassi dell'identificatore

Resource: qualsiasi cosa abbia un'identità

Identifier: informazioni che permettono di distinguere un oggetto da un altro

Due tipi di URI: URL e URN

URL:

Sottoinsieme di uri che identifica le risorse attraverso un meccanismo di accesso

URN: uniform resource name

Identificativi che vogliono essere globalmente unici, persistente anche quando la risorsa non è più accessibile

Es. urn:oasis:names:specification:docbook:dtd:xml:4.1.2

Sintassi URL:

<scheme>://<user>:<password>@<host>:<port>/<path>

Schema: meccanismo di accesso a questa risorsa (https, http, ftp)

User e password: deprecato, nella sintassi della url c'è, opzionale

Host: nome di dominio o indirizzo IP

Porta: numero di porta del server, alcuni protocolli lo hanno di default (opzionale)

Path: percorso. Identifica, dentro il sito a una risorsa specifica con un certo identificativo, consiste in una sequenza di segmenti separati da '/'. Ad es path nel file system del server

HTTP URL

http://<host>:<port>/<path>?<query>

: query: stringa di informazioni che deve essere interpretata dal server

Le url di distinguono in absolute e relative:

- *Absolute*: identifica una risorsa indipendentemente dal contesto in cui è usata
- *Relative*: informazioni che possono identificare una risorsa in base ad un'altra URL

Le url relative non viaggiano sulla rete, sono interpretate dal browser in base al documento di partenza. (file locali)(?)

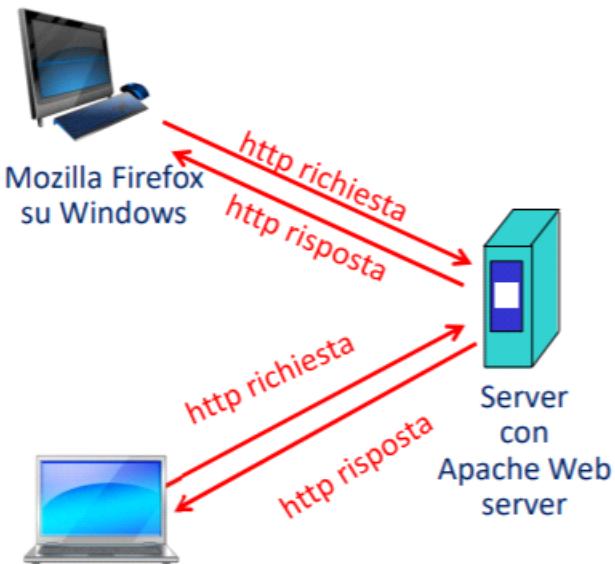
Se cambi parte del percorso/host non lo trovi più :p

HTTP (Hypertext Transfer Protocol)

Usato dal 1990 come protocollo di trasferimento per il WWW

Protocollo generico stateless che può essere usato per ipertesto, nome di server, oggetti distribuiti, estensibile tramite i suoi metodi.

Usato per tipizzazione e negoziazione dei formati di rappresentazione dei dati



Chrome su S.O. Ubuntu

Il protocollo http è di tipo richiesta/risposta

Protocollo stateless:

Le coppie r/r sono indipendente, ogni richiesta viene eseguita indipendentemente da quelle che l'hanno preceduta

Usa il protocollo di trasporto TCP (connection-oriented)

Il client richiede l'instaurazione di una connessione TCP, i processi si scambiano messaggi attraverso le rispettive socket

Connessione: un circuito logico di livello trasporto stabilito tra due programmi applicativi per comunicare tra loro

Connessione non persistente

Viene stabilita una connessione tcp separata per recuperare ciascuna URL

Connessione persistente

A meno che non sia indicato il client assume che il server tenga la connessione aperta

Dopo la chiusura il client non deve più inviar richieste su quella connessione

Supponiamo che l'utente digitò la URL
 www.someSchool.edu/someDepartment/home.index

(file html contenente te
 e riferimenti a
 10 immagini jpg)

- 1a. Il client http inizia la connessione TCP verso il server http al www.someSchool.edu. Socket Porta 80 è default per il server http.
- 1b. Il server http dell'host www.someSchool.edu aspetta le richieste di connessione TCP connection alla porta 80. "Accetta" la connessione, e lo notifica al client
- 2. Il client http invia un *messaggio di richiesta http* (che contiene la URL) alla socket di connessione con lo strato di trasporto (TCP)
- 3. Il server http riceve il msg di richiesta, compila un *messaggio di risposta* che contiene l'oggetto richiesto (someDepartment/home.index), manda il messaggio alla socket

tempo ↓

- tempo**
4. Il server http chiude la connessione TCP
 5. Il client http riceve il messaggio di risposta che contiene il file html e lo visualizza. Percorrendo il file trova il riferimento a 10 oggetti jpg

6. Ripete i passaggi da 1-5 per ognuno dei 10 oggetti jpg

Ipotesi: connessione non persistente

Il protocollo http è “stateless”

ogni risposta è collegata solo alla richiesta che l'ha generata, indipendentemente dalle richieste precedenti

Con una connessione persistente, la stessa connessione http può essere utilizzata per ricevere ulteriori richieste; il server chiude la connessione quando viene specificato nell'header del messaggio o per timeout

Pipelining

Serve per migliorare ulteriormente le prestazioni

Consiste nell'invio da parte del client di molteplici richieste senza aspettare la ricezione di ciascuna risposta

Il server DEVE inviare le risposte nello stesso ordine in cui sono state ricevute le richieste

Il client non può inviare in pipeline richieste che usano metodi http non idem-potenti

Non è molto usato

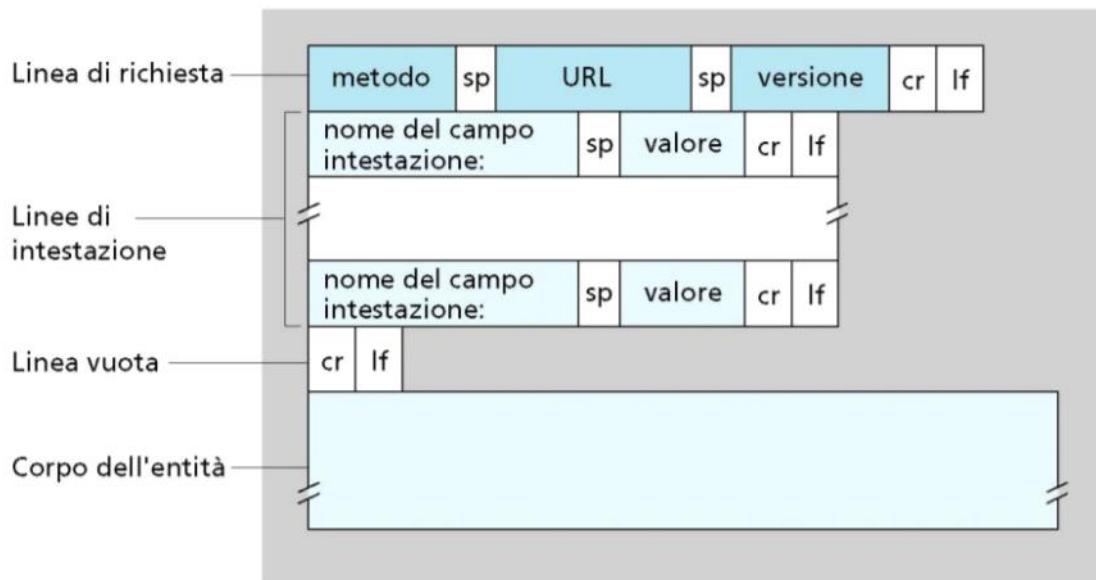
Messaggi HTTP

Può essere di tipo **request** o **response**

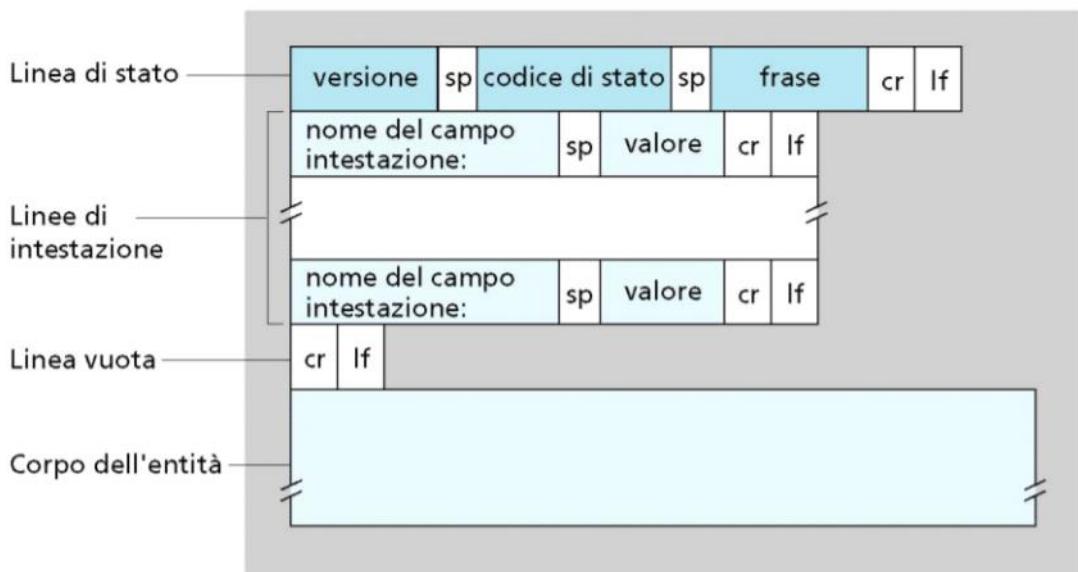
La riga iniziale distingue la richiesta dalla risposta (start - line)

Seguono una serie di HEADER (coppie di nome-valore) e poi il corpo del messaggio (non sempre)

HTTP Request message



HTTP Response message



Metodi principali del protocollo HTTP

HTTP request

```
Request = Request-Line
         * ( general-header
           | request-header
           | entity-header )
         CRLF
         [ message-body ]
```

Esempio

```
GET /pub/WWW/TheProject.html HTTP/1.1
Host: www.w3.org
Connection: close
User Agent: Mozilla/4.0
Accept-language: it
```

```
Request-Line = Method SP
               Request-URI SP
               HTTP-Version CRLF
```

Header

Ci sono tante tipologie di header, anche i valori via via sono stati estesi.

Insieme di coppie nome-valore

- *General header*: relativi alla trasmissione es data, codifica, connessione (connection close, connection keepAlive)
- *Entity header*: contenuto del corpo del messaggio: tipo, lunghezza, data di scadenza...
- *Response header*: caratterizza il messaggio di risposta: informazioni sul server, autorizzazione richiesta...
- *Request Header*: intestazioni specifiche per la richiesta: tipo di software, chi sta facendo la richiesta, caratteristiche del client, di autorizzazione.

Chunked: modo per far mandare ai server degli aggiornamenti successivi, mantenersi la connessione aperta per mandare un po' di info ora e un po' dopo(?), escamotage per tenere aperta per un po' la connessione.

General Headers

```
Date: Tue, 15 Nov 1994 08:12:31 GMT
Connection: close
Transfer-Encoding: chunked
```

Request Headers

```
Accept: text/plain; q=0.5, text/html,
        image/png, application/json
Accept-Charset: iso-8859-5,
                unicode-1-1;q=0.8
Accept-Encoding: compress, gzip
```

'q' indica una preferenza default: 1

Accept charset: set di caratteri accettabile per la risposta

Accept encoding: tipo di codifica accettabile per la risposta

HTTP response

```
Response = Status-Line
           *( general-header
             | response-header
             | entity-header )
           CRLF
           [ message-body ]
```

```
HTTP/1.1 200 OK
Date: Sun, 14 May 2000 23:49:39 GMT
Server: Apache/1.3.9 (Unix) (Red Hat/Linux)
Last-Modified: Tue, 21 Sep 1999 14:46:36 GMT
```

Status Line

```
Status-Line
  HTTP-Version SP
  Status-Code SP
  Reason-Phrase CRLF
```

Esempio: `HTTP/1.1 200 OK`

Status line: prima riga di un messaggio di risposta

Status code: intero a 3 cifre che indica il risultato dell'operazione richiesta

Reason-phrase: descrizione testuale dello status code

Tipi di status code:

```
1xx: Informational - Request received, continuing process
2xx: Success - The action was successfully received,
understood, and accepted
3xx: Redirection - Further action must be taken in order
to complete the request
4xx: Client Error - The request contains bad syntax or
cannot be fulfilled
5xx: Server Error - The server failed to fulfill an
apparently valid request
```

"100" - Continue	"101" - Switching Protocols
"200" - OK	"201" - Created
"202" - Accepted	"203" - Non-Authoritative Information
"204" - No Content	"205" - Reset Content
"206" - Partial Content	
"300" - Multiple Choices	"301" - Moved Permanently
"302" - Moved Temporarily	"303" - See Other
"304" - Not Modified	"305" - Use Proxy
"400" - Bad Request	"401" - Unauthorized
"402" - Payment Required	"403" - Forbidden
"404" - Not Found	"405" - Method Not Allowed
"406" - Not Acceptable	"407" - Proxy Authentication Required
"408" - Request Time-out	"409" - Conflict
"410" - Gone	"411" - Length Required
"412" - Precondition Failed	"413" - Request Entity Too Large
"414" - Request-URI Too Large	"415" - Unsupported Media Type
"500" - Internal Server Error	"501" - Not Implemented
"502" - Bad Gateway	"503" - Service Unavailable
"504" - Gateway Time-out	"505" - HTTP Version not supported

<http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>

101 - switching protocols: succede quando per eseguire una richiesta devo cambiare il protocollo
 201 - con i metodi http possiamo chiedere al server di creare una nuova risorsa, se l'operazione va a buon fine il server risponde con un 201. usato nella API REST (usare http per far parlare due programmi)

Response headers

Intestazioni che ci danno informazioni sulla risposta che non possono essere inserite nello status line. Ad es programmi e librerie usate dal server per la risposta

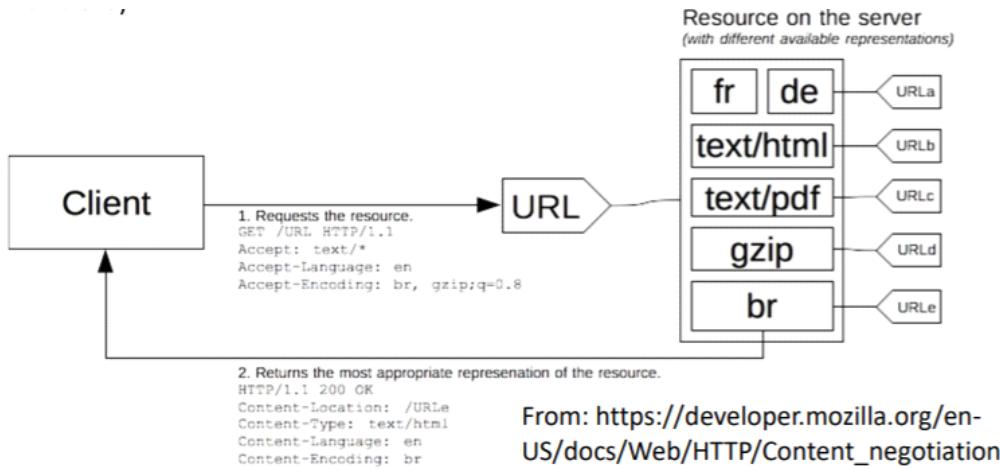
Location: usato per ridirezionare il ricevente a una URI differente dalla request URI per completare la richiesta o accedere alla risorsa

Entity headers

```

Content-Base
  URI assoluta da usare per risolvere le URL relative
  contenute nell'entity body
Content-Encoding
  codifica dell'entity body (es: gzip)
Content-Language
  lingua dell'entity body (es: en, it)
Content-Type
  tipo dell'entity body (es: text/html)
Expires
  (utile per caching) validità temporale del contenuto
Last-Modified
  (utile per caching) data dell'ultima modifica sul server
  
```

Content negotiation



Il client invia una prima richiesta al server, il server può rappresentare la risposta in vari modi (lingue, formati), ha pronte le risorse in varie *rappresentazioni* il client dice che accetta text, en, br e gzip con preferenza minore. Risponde con URL-e, gli rimanda il contenuto e descrive cosa ha mandato.

Content negotiation: meccanismo per selezionare la rappresentazione appropriata quando viene servita una richiesta

Options:

Option è una request URI e il server risponde quali operazioni sono disponibili su quella request URI

```

OPTIONS http://192.168.11.66/manual/index.html
        HTTP/1.1
host: 192.168.11.66
Connection: close

HTTP/1.1 200 OK
Date: Sun, 14 May 2000 19:52:12 GMT
Server: Apache/1.3.9 (Unix)  (Red Hat/Linux)
Content-Length: 0
Allow: GET, HEAD, OPTIONS, TRACE
Connection: close

```

GET - REQUEST

Metodo che richiede il trasferimento di una risorsa identificata da una URL o operazioni associate all'URL stessa

Sono possibili conditional get (header "if-...")

Header: if-Modified-Since, If-Unmodified-Since (date) ...

Partial get: Header:Range

GET - RESPONSE

```

HTTP/1.1 200 OK
Date: Sun, 14 May 2000 19:57:13 GMT
Server: Apache/1.3.9 (Unix) (Red Hat/Linux)
Last-Modified: Tue, 21 Sep 1999 14:46:36 GMT
ETag: "f2fc-799-37e79a4c"
Accept-Ranges: bytes
Content-Length: 1945
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2
Final//EN">
<HTML> ...

```

Get condizionale:

```

GET http://192.168.11.66 HTTP/1.1
Host: 192.168.11.66
If-Modified-Since: Tue, 21 Sep 1999 14:46:36 GMT

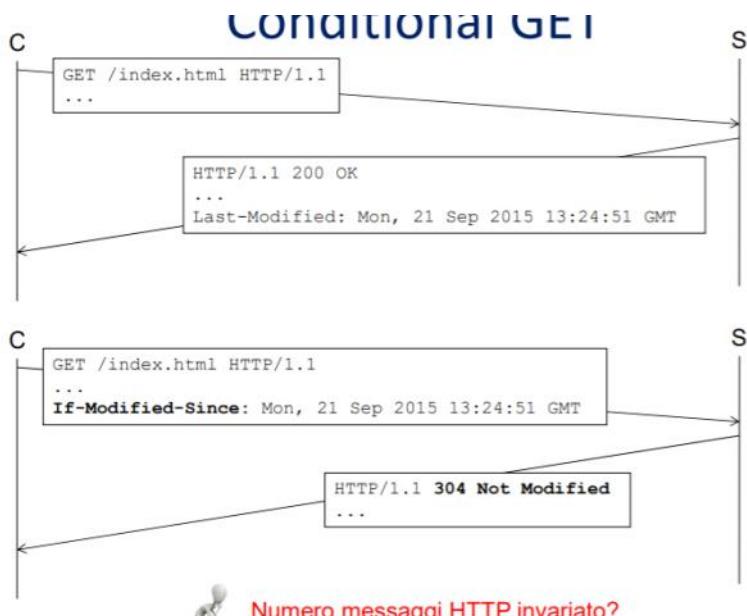
```

```

HTTP/1.1 304 Not Modified
Date: Wed, 22 Sep 1999 15:06:36 GMT
Server: Apache/1.3.9 (Unix) (Red Hat/Linux)

```

Vuole la risorsa se è stata modificata dal giorno indicato, il server risponde con codice 304 - not modified



Mando comunque la risposta col messaggio HTTP, ma non mandando la risorsa risparmio.

Request method - HEAD

```

HEAD http://192.168.11.66 HTTP/1.1
host: 192.168.11.66
Connection: close

```

```
HTTP/1.1 200 OK
Date: Sun, 14 May 2000 20:02:41 GMT
Server: Apache/1.3.9 (Unix) (Red Hat/Linux)
Last-Modified: Tue, 21 Sep 1999 14:46:36 GMT
ETag: "f2fc-799-37e79a4c"
Accept-Ranges: bytes
Content-Length: 1945
Connection: close
Content-Type: text/html
```

Fa una richiesta analoga alla get, ma in risposta voglio solo l'intestazione a quella che sarebbe stata la risposta della get non voglio il corpo. Utile per controllare lo stato dei documenti

Request method - POST

Con POST il client può mandare nel corpo del messaggio informazioni al server (compilare un form, postare un messaggio, scrittura su database)

Request method - PUT & DELETE

Con la **put** il client può chiedere di modificare una risorsa, specifica nella request URI l'identificativo della risorsa. (recupero una risorsa con GET)

Con la **delete** il client chiede di cancellare una risorsa identificata dalla request URI

Metodi safe e idempotenti

Metodi che non hanno effetti collaterali, non vanno a modificare la risorsa: GET, HEAD, OPTIONS, TRACE (safe)

I metodi idempotenti possono modificare la risorsa, ma se fai la richiesta più volte l'effetto non cambia. Se fai una put la risorsa viene creata, se la rifai uguale non succede nulla: Get, HEAD, PUT, DELETE, OPTIONS, TRACE

Pipeline

La post non è un metodo idempotente

Web caching

Soddisfare il cliente senza contattare il server

Memorizzare copie temporanee di risorse Web e riservirle al client per ridurre l'uso di risorse e diminuire il tempo di risposta

User Agent Cache: il browser (agent) mantiene una copia delle risorse visitate dall'utente

Proxy Cache: dispositivo intermedio, intercetta il traffico e mette in cache le risposte. La richiesta quando ho un proxy nel mezzo viene prima presa dal proxy che controlla se ha quella risorse, altrimenti la chiede al server, che la manda al proxy che la inoltra e si mantiene una copia.

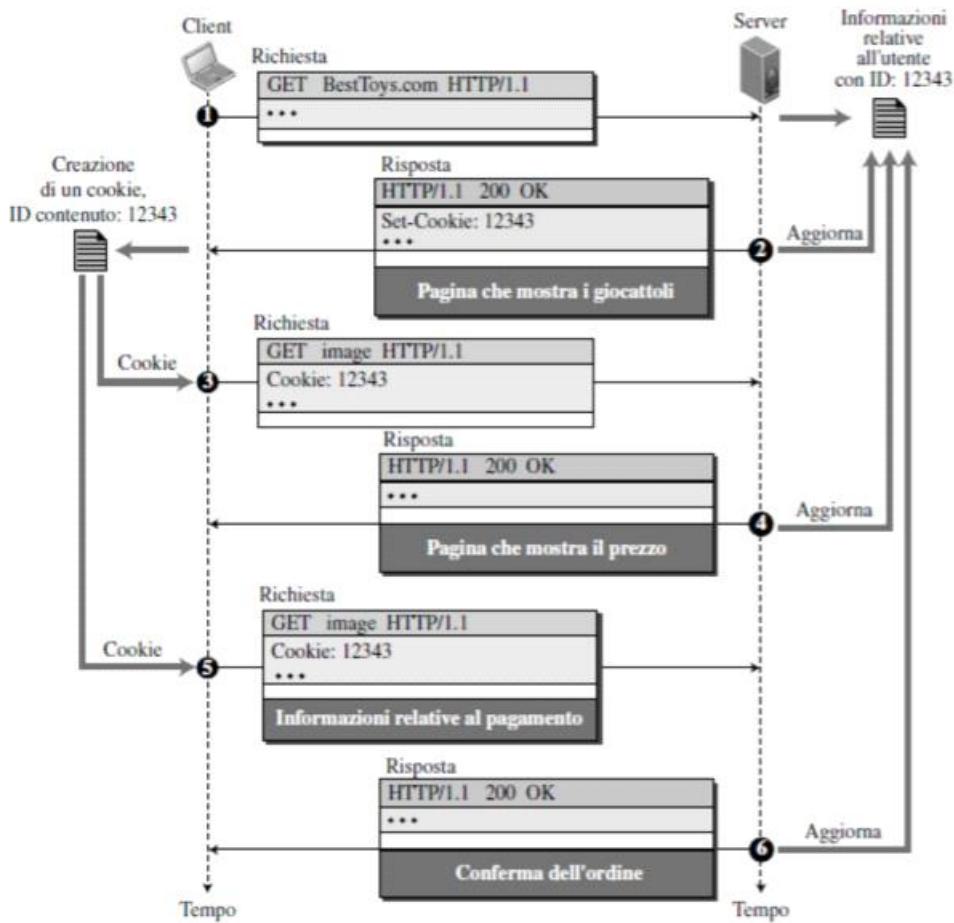
L'uso del proxy è trasparente, non inoltri te la richiesta al proxy. Il server origine non sa chi a fatto la richiesta e risponde al proxy.

I cookie

Http è stateless

Per poter gestire il concetto di sessione e risorse che possono cambiare stato con l'interazione, su un protocollo stateless si possono usare vari meccanismi, il principale sono i cookie.

Servono per dare al server delle informazioni sull'identità dell'utente, sue preferenze e eventuali azioni che lui può aver fatto in passato. Non ho le informazioni di ip e porta che posso associare a un certo utente perché mi collego magari da dispositivi e reti diverse.



1. Il client fa una richiesta al server
2. Il server vede la richiesta, crea un *identificativo* e chiede al browser di memorizzarsi un *cookie con questa informazione*
3. Da questo punto in poi nella navigazione il client manderà nella richiesta oltre alle varie informazioni anche il cookie
4. Il server a questo punto sa come interpretarlo, *con la richiesta e l'identificativo può aggiornarsi delle informazioni della sessione o della risorsa che sta gestendo*, mantenendo il protocollo stateless.

Ci sono cookie di vario tipo:

Alcuni avvantaggiano l'utente nell'uso di servizi di vario tipo e altre di tracciamento dell'utilizzo dell'utente.

Vengono ancora molto usati in modo più normato. (accettazione dei cookie)

Per vedere i messaggi http chrome -> strumenti -> altri strumenti -> strumenti per sviluppatori

TELNET

Protocollo per l'accesso remoto a delle macchine, servizio di terminale remoto su reti TCP/IP
ancestor di SSH

Sta per terminal network.

È un protocollo per interazione client server di login remoto generale.

Mi permette, da una macchina di accedere a un'altra che può avere caratteristiche diverse, utilizzo dei comandi che vengono interpretati dall'altra parte.

Telnet permette a un utente su una macchina di stabilire una connessione con un login remoto

I comandi inseriti sulla macchina locale vengono inviate sulla macchina remota, quella produrrà un output che verrà mostrato sulla macchina locale

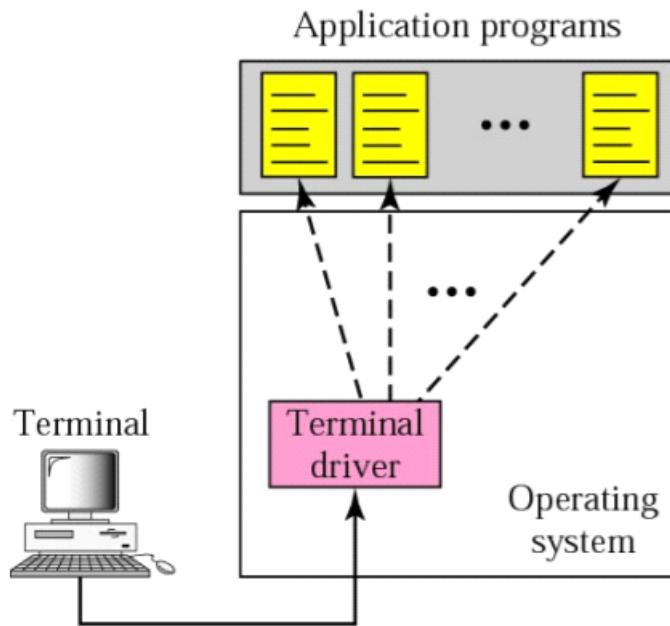
Modello di telnet

Programma server che accetta le richieste

Programma client che effettua le richieste

Come standard il client con telnet si connette alla porta 23 tramite connessione TCP

Come funziona in locale

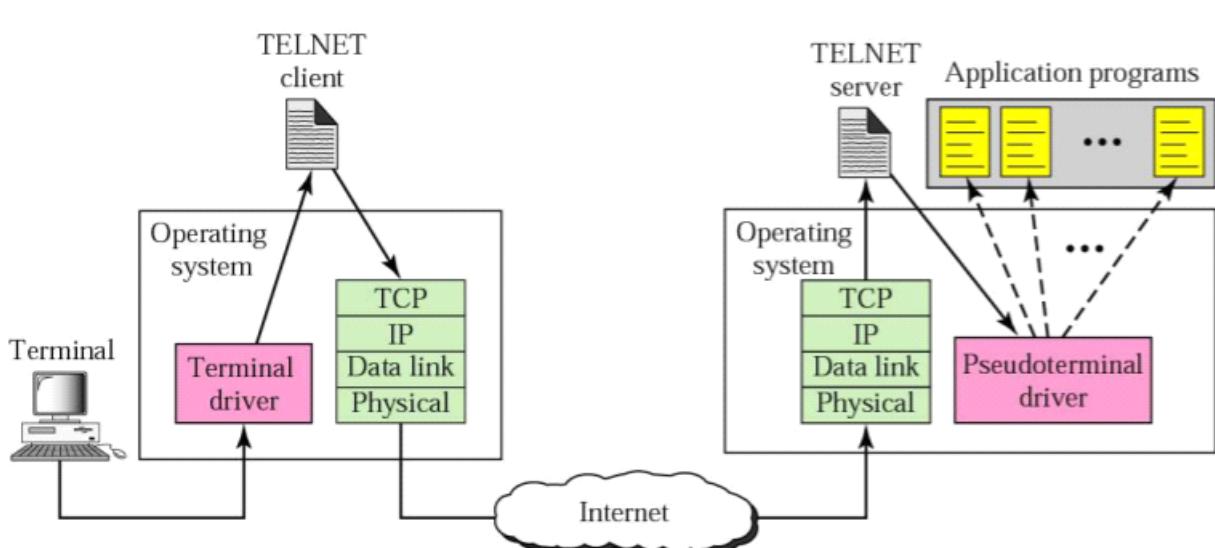


Terminale, driver di terminale

Il sistema operativo assume che gli input ad un processo vengano forniti da tastiera e che gli output siano inviati al monitor

Lato client ho il terminale dell'utente che manda il comando che viene acquisito tramite il driver di terminale e tramite il programma telnet client incapsula il messaggio e lo trasmette.

Sulla macchina in remoto il messaggio viene decapsulato e mandato al telnet server che li manda allo pseudoterminale driver che interagisce con le applicazioni



Pseudoterminale driver: entry point del s.o. che consente di trasferire caratteri a un processo come se provenissero dal terminale. Accetta i caratteri dal server telnet e li trasmette al s.o. che li trasmette all'opportuna app

Telnet: NVT

Problema: TELNET deve poter operare con il numero massimo di sistemi e quindi gestire dettagli di sistemi operativi eterogenei.

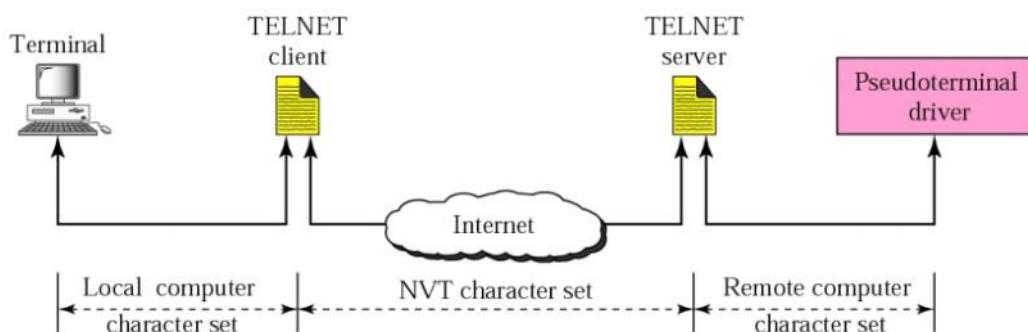
I terminali possono differire gli uni dagli altri per:

- il set e codifica di caratteri
- la larghezza della linea e lunghezza della pagina
- i tasti funzione individuati da diverse sequenze di caratteri (escape sequence)
 - es. diversa combinazione di tasti per interrompere un processo (CTRL-C, ESC), caratteri ASCII diversi per terminazione righe di testo

Soluzione: definizione di un terminale virtuale. Sulla rete si considera un unico terminale standard e in corrispondenza di ogni host, si effettuano la conversione da terminale locale a terminale virtuale e viceversa

Telnet assume che sui due host sia in esecuzione un Network Virtual Terminal (NVT); la connessione TCP è instaurata tra questi due terminali NVT

Il NVT definisce un set di caratteri e di comandi universale.



Protocolli per implementare servizio di mail

Trasferimento di un messaggio tra un mittente e un destinatario

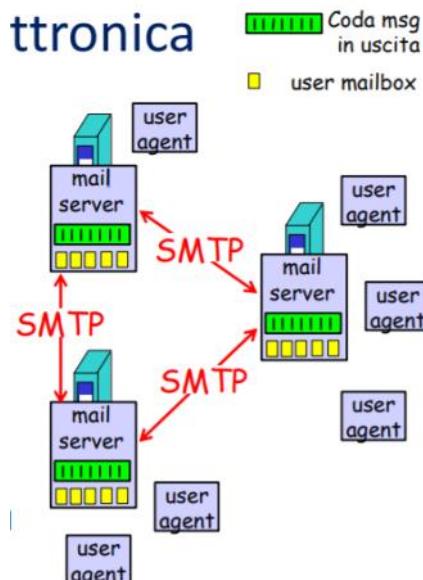
Servizio asincrono.

Offre delle garanzie sulla consegna dei messaggi. Messaggi di errore se un messaggio non viene consegnato

Componenti per realizzare il servizio:

- User agent: componenti che permettono all'utente di comporre un messaggio, richiedere l'invio, riceverlo e visualizzarlo (outlook, gmail)
- Intermediari: Mail Server
Questi mail server comprendono aree in cui vengono archiviati i messaggi in entrata (messaggi che un server ha ricevuto e che deve trasferire a altri mail server) e mailbox, (cassetta di posta), coda di messaggi in uscita

Protocollo SMTP, protocollo con cui i mail server dialogano per gestire il trasferimento di mail



5

User agent: per composizione editing e lettura messaggi

Mail server:

- i messaggi in entrata ed uscita vengono archiviati sul server
- Le mailbox contengono messaggi in ingresso diretti all'utente
- Una coda di messaggi in uscita

Come fa un mail server a sapere dove recapitare la mail?

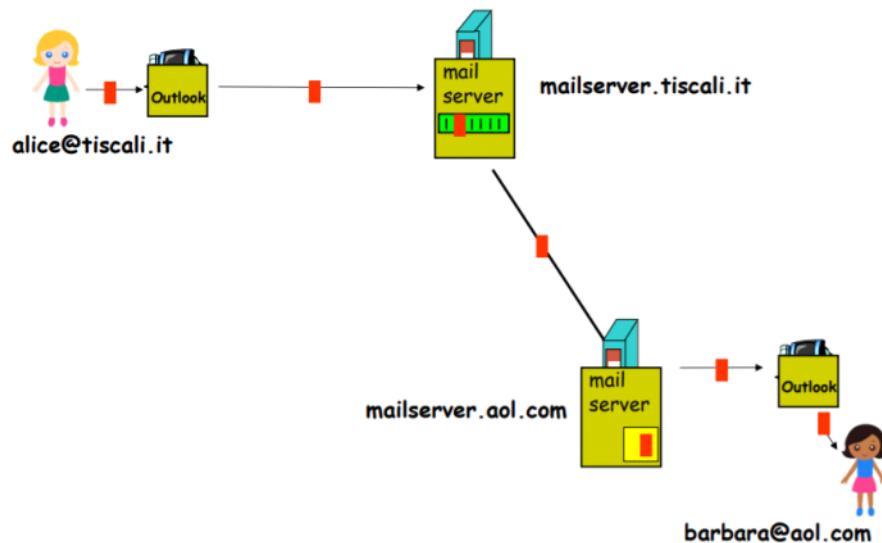
E quello di destinazione a quale casella di posta?

Indirizzo email

local-part @ domain-name

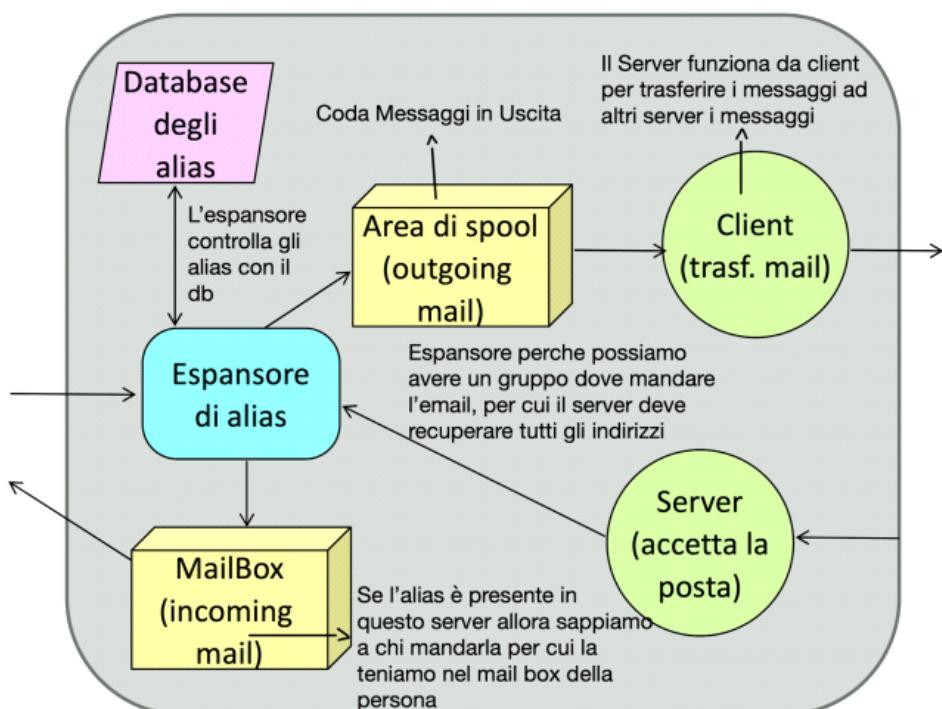
Domain-name: mail server destinazione

Local-part: cassetta di posta nel mail server



Come funziona all'interno un mail server

----- **COMPONENTI SERVER POSTALE:**



Il mail server è in ascolto di messaggi che gli vengono inviati da user agent o altri mail server, le mail se non sono dirette ai proprio utenti devono essere trasferiti ad altri utenti quindi vengono messi in **area di spool (outgoing mail)** quindi il mail server per ciascuna mail guarda il nome di dominio e come un client chiede una connessione tcp e trasferisce al nuovo mail server, si tiene una copia finché il trasferimento non va a buon fine (o fino a scadenza messaggio).

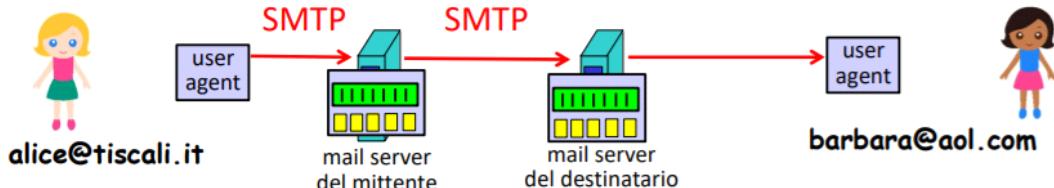
Se invece la mail che riceve è per uno dei propri utenti lo mette in **mailBox**.

Alias:

Non necessariamente c'è una corrispondenza 1 a 1 tra utente e indirizzo di mail.

Magari indirizzo **direttore@unipi.it** corrisponde a **mario.rossi@unipi.it** e quindi con le solite credenziali riceve anche le mail di direttore

Le stesse cose possono essere fatte anche con nome di dominio diversi.



- Servizio di trasferimento posta:
 - Intermediari, no consegna diretta
- **SMTP [RFC 2821]**: distribuisce/archivia nel mail server del destinatario
 - **Protocollo di tipo «push»**

Protocollo push:

Il client riceve una mail la deve trasferire e richiede una connessione tcp per delegarla all'altro
Arriva mail -> attiva connessione -> trasferisce

SMTP

L'obiettivo di SMTP è un trasferimento affidabile e efficiente di mail. Protocollo applicativo che usa TCP. Mail affidabile -> servizio di trasporto affidabile

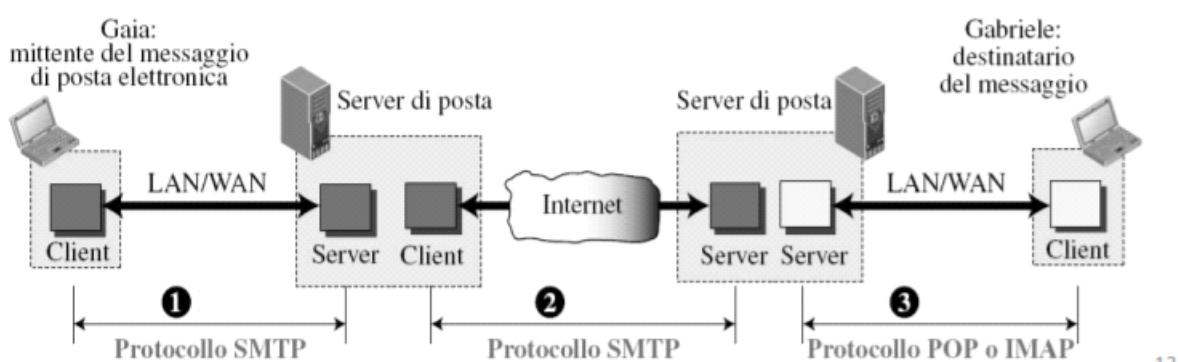
SMTP richiede solo un trasferimento di dati ordinato e affidabile.

Porta mail attraverso intermediari tra mittente e destinatario

Quando un client SMTP vuole trasferire un messaggio, stabilisce un canale di trasmissione bidirezionale con un server SMTP. La responsabilità di un client è di trasferire la mail a un server SMTP, o comunicare un eventuale insuccesso (**scambio formale di responsabilità**).

Un client SMTP determina l'indirizzo di un host appropriato che ospita un server SMTP risolvendo il nome della destinazione in un mail server destinazione

- risoluzione di nome in indirizzo IP attraverso il DNS).





.. non ho ricevuto il tuo messaggio

Possibili problemi:

- connessione con mailserver del mittente (server inesistente o irraggiungibile)
- connessione con mailserver destinatario (server inesistente o irraggiungibile)
- inserimento in mailbox destinatario (user unknown, mailbox full)
... ma in tutti questi i casi **il mittente riceve una notifica!**

Destinatario può non ricevere (senza che mittente sia avvisato)
solo se qualcuno (intruso, filtro antispam) rimuove messaggio



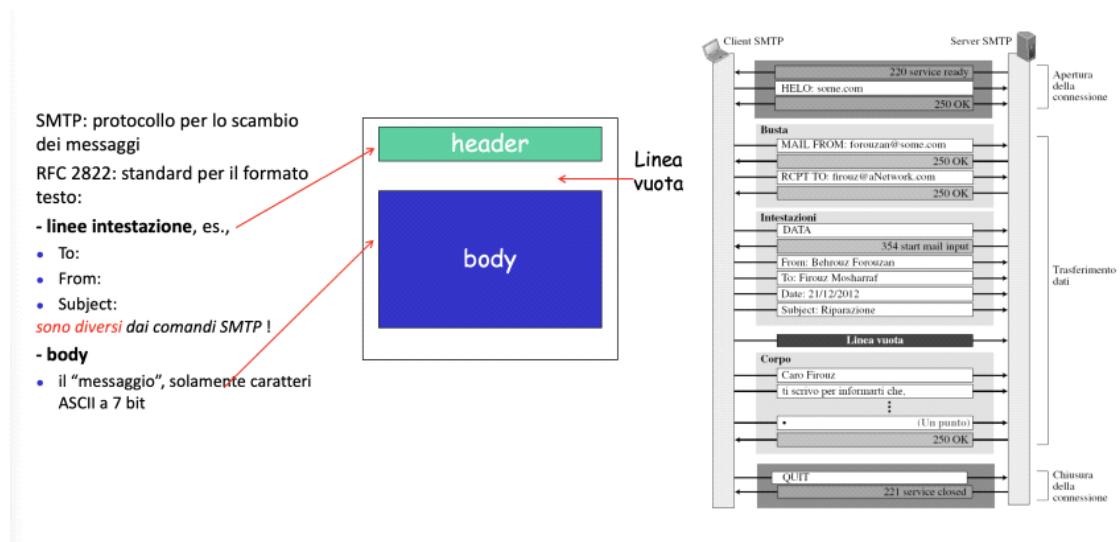
Come funziona il trasferimento di mail

Si usa protocollo TCP porta di default 25.

Fasi di trasmissione:

- Handshaking:
client e server di "presentano"
- Trasferimento del messaggio
- Chiusura della connessione

L'interazione avviene con uno scambio di comandi/risposte ASCII 7 bit



Estensioni multimediali

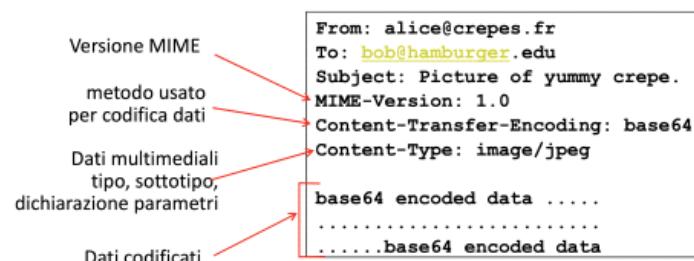
Tipi MIME (Multipurpose Internet Mail Extension):

MIME è un'estensione, non un protocollo, codifica tutti i contenuti a monte come ASCII così da poter mandare tutto via SMTP, fu fatto così in modo da utilizzare tutti i protocolli esistenti

Supporta:

- Testo in set di caratteri diversi da ASCII
- Allegati non testuali
- Corpo del messaggio in più parti (multi-part)
- Header in set di caratteri non ASCII

MIME (RFC-2045) definisce un insieme di metodi per rappresentare dati binari in ASCII



- Client e server SMTP si aspettano messaggi ASCII (caratteri ASCII che usano 7 degli 8 bit di un byte). I dati binari usano invece tutti gli 8 bit di un byte (es. immagini, eseguibili, set estesi di caratteri).
- MIME fornisce vari schemi di transfer encoding, tra cui:
 - ASCII encoding of binary data: base 64 encoding
 - Gruppi di 24 bit sono divisi in 4 unità da 6 bit e ciascuna unità viene inviata come un carattere ASCII
 - Quoted-printable encoding: per messaggi testuali con pochi caratteri non-ASCII, più efficiente
- **Oggigiorno i mail server possono negoziare l'invio di dati in codifica binaria (8 bit), se la negoziazione non ha successo si usano i caratteri ASCII**

Tipi MIME

Content-Type: type/subtype; parameters

Specificano la natura dei dati nel corpo di un'entità MIME

Text

esempi di subtypes: **plain**,
html

Video

esempi di subtypes: **mpeg**,

Image

esempi di subtypes: **jpeg**, **gif**

Application

altri dati che devono essere
processati da un'applicazione
prima di essere visualizzabili

Audio

esempi di subtypes: **basic** (8-bit
mu-law encoded), **32kadpcm** (32
kbps coding)

esempi di subtypes: **msword**,
octet-stream (dati
arbitrari binari)

Mail multipart:

Parti del testo codificate in modo diverso.

--98766789 1° parte

Content-Transfer-Encoding: quoted-printable

Content-Type: text/plain

Dear Bob,

Please find a picture of a crepe.

--98766789 2° parte

Content-Transfer-Encoding: base64

Content-Type: image/jpeg

base64 encoded data

.....

.....base64 encoded data

--98766789--

Si usa una riga codificata per distinguere dove iniziano le varie parti

Protocolli di accesso alla mail

Problema del trasferimento allo user agent

Viene usato un altro protocollo.

Perché non SMTP? Per mandare la mail allo user agent del destinatario

Se usassi SMTP il mail server del destinatario agirebbe da client perché dovrebbe fare la richiesta e quindi avremmo bisogno di un server SMTP per la connessione TCP.

Abbiamo quindi bisogno di un protocollo di tipo pull.

L'user agent fa la richiesta di "se ci sono mail per l'utente"

Accesso alla posta: POP3

Usiamo TCP anche qui (porta 110) verso il server di posta.

Fasi:

Autorizzazione

Aggiornamento

Transazione

Accesso alla posta: IMAP

Consente di organizzare la posta in cartelle. Manipola i messaggi sul server, comandi per estrarre solo alcuni componenti dei messaggi

Punti deboli del servizio:

SPAM

Messaggi forged

Messaggi letti da intrusi

STATO APPLICATIVO

DNS

A livello applicativo si parla di *nomi*

Serie di caratteri scelti da un alfabeto finito

Identificativo applicativo disaccoppiato dall'indirizzo ip della macchina

HO BISOGNODI UN IDENTIFICATIVO A QUEL LIVELLO DI ASTRAZIONE

E faccio un associazione poi tra il nome del livello applicativo e l'ip della macchina

IP: usato per instradare datagrammi in rete

Il DNS ha come obiettivo trovare associazione tra nome a livello applicativo e l'indirizzo ip

Quando internet era "piccolo" era facile trovare associazione, tutti i nomi logici e i relativi IP erano contenuti in un file

Periodicamente tutti gli host prelevavano una versione aggiornata del file

Adesso non è fattibile così

Si ha quindi un sistema DNS (domain name system) con regole e organizzazione gerarchica con una delega delle autorità

DNS è posizionato a livello applicativo -> gira sui sistemi terminali

Si basa sul paradigma client-server e usa i servizi del livello di trasporto per mandare le richieste di risoluzione dei nomi e le risposte (associazione)

Non interagisce con gli utenti (a meno di messaggi di errore nella risoluzione)

I servizi complessi vengono messi alle estremità della rete

1. Scrivi un url nel browser
2. Il browser chiede al DNS l'indirizzo ip di quel nome
3. Il DNS manda la traduzione
4. Con la traduzione potrà chiedere al livello di trasporto di stabilire la connessione
5. Dalla connessione farà la sua richiesta e riceverà la risposta

IL PROTOCOLLO:

- Specifica sintassi dei nomi e modo per gestirli
- Consente la conversione da nomi a indirizzi e viceversa

1. Schema di assegnazione dei nomi gerarchi basato su domini
2. Database distribuito contenenti i nomi e i corrispondenti indirizzi con gerarchia di name server
3. Protocollo per la distribuzione delle informazioni sui nomi tra nome e server
 - a. Host, router, name server comunicano per **risolvere** i nomi
 - b. Utilizzando udp (porta 53) oppure TCP

Servizi offerti dal DNS

1. Risolvere nomi in indirizzi IP: **traduzione**
2. Mappare il fatto che un host possa avere più nomi (**host aliasing**) (www.enterprise.com / enterprise.com)
3. Distribuzione carico: associare al nome canonico di una macchina, più indirizzi IP per bilanciare il carico
4. **Mail server aliasing:** con il nome del dominio prendiamo anche il server mail

Struttura gerarchica

Un nome è costituito da diverse parti

- Nome organizzazione, dipartimento, ufficio ecc.

Assegnazione del nome delegabile -> sistema (in buona parte) decentralizzato

Delega dell'autorità per l'assegnazione delle varie parti dello spazio dei nomi

Distribuzione responsabilità della conversione tra nomi e indirizzi

Nomi di dominio

Spazio dei nomi -> struttura gerarchica, rappresentabile con un albero

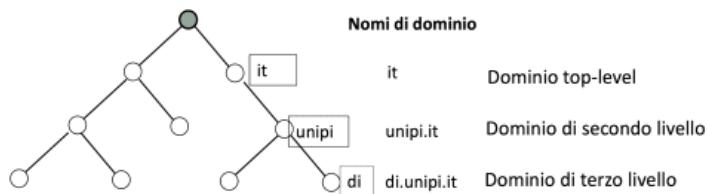
Ogni nodo dell'albero ha un nome di dominio,

Top level: *.it*

Secondo livello: *unipi.it*

Terzo livello: *di.unipi.it*

Dominio: sottoalbero nello spazio dei nomi di dominio che viene identificato dal nome di dominio del nodo radice del sottoalbero.



Internet, spazio dei nomi in internet può essere suddiviso in centinaia di domini con i loro sottodomini

I nomi gerarchici delle macchine sono assegnati in base alla struttura delle organizzazioni che ottengono l'autorità per porzioni nello spazio dei nomi

La struttura gerarchica permette autonomia nella scelta dei nomi all'interno del dominio

Esempio: server1.di.unipi.it != da server1.cs.cornell.edu

Top level domain

.com .edu .Mil .gov .net .org e codici geografici (.it .Uk .Fr ..)

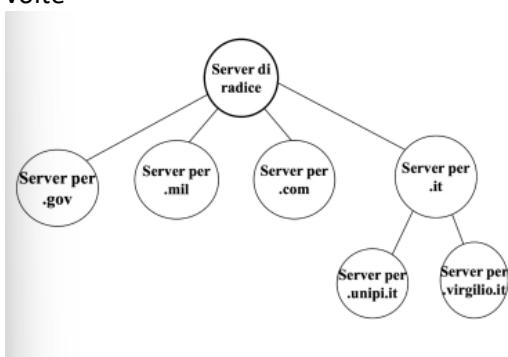
Mantenuti da IANA (internet assigned numbers authority)

NAME SERVER

Struttura con cui dislochiamo la traduzione all'interno degli effettivi server

- DNS: database distribuito implementato in una gerarchia di più name server
- Name Server: programma che gestisce la conversione da nome di dominio a indirizzo IP

Di solito vengono distribuiti in base alla zona della regione dove il dominio deve operare il più delle volte



GERARCHIA SERVER

- Server Radice (root name server): responsabile dei record della zona radice, restituisce sui TLD, distribuiti in tutto il mondo
- Server Top-Level-Domain (TLD): mantiene le informazioni di un certo top level domain (.it, .com...)
- Server di Competenza (authoritative name server): memorizza nome e ip di un insieme di host, effettua la traduzione per quegli host, sono divisi per zone e ce ne possono essere di primari e secondari

Grazie alla gerarchia dei name server io posso distribuire le info sui domini su più name server
Zona: regione di cui è responsabile un certo name server (non necessariamente coincide col dominio)

Il server immagazzina le informazioni relative alla propria zona, inclusi i riferimenti ai server di domini di livello inferiore

Server radice

Responsabili delle informazioni di top-level domain

Centinaia di root name server in tutto il mondo

Server top level domain

Mantengono le informazioni di quel livello più restituiscono le info sui name server di competenza

dei sottodomini

Server di competenza

Autorità per una certa zona

Memorizzare nome e ip di un insieme di host

Può effettuare traduzioni per quegli host

Per una certa zona ci possono essere server di competenza primari e secondari

P: file di zona

S: ricevono i file di zona e li offrono al servizio di traduzione

Local name server

Non appartiene strettamente alla gerarchia dei server

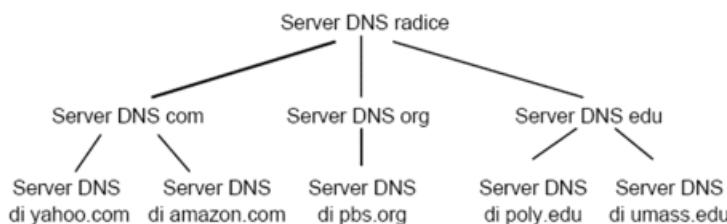
Ogni ISP ha il suo

Le query DNS vengono prima rivolte al name server locale

Quando un programma deve trasformare un nome in ip chiama un programma detto **resolver** passando il nome come parametro di ingresso

Il quale se non ha le associazioni richieste in cache chiama il Local Name Server, un server di cui conosce l'indirizzo IP, se questo Local Name Server ha la traduzione bene, sennò inoltra la query alla gerarchia dei DNS

È un server che si può implementare in locale o affidarsi ad altri, non appartiene alla gerarchia.



Il client vuole l'IP di www.amazon.com; 1^a approssimazione:

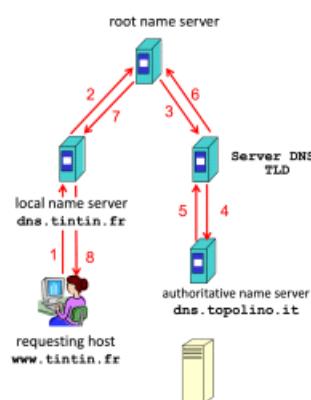
- 1 Il client interroga il server radice per trovare il server DNS com
- 2 Il client interroga il server DNS com per ottenere il server DNS amazon.com
- 3 Il client interroga il server DNS amazon.com per ottenere l'indirizzo IP di www.amazon.com

QUERY

Ricorsive

Non vengono più molto usate. Si richiede una conversione completa.

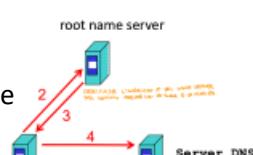
- 1- l'utente chiama il suo DNS locale
- 2- Il DNS locale contatta il root name server
- 3- Il root name server contatta tutta la scaletta fino a ritrovare l'indirizzo



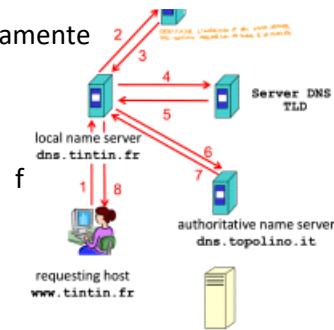
Iterative

Servizio minimo che i server devono erogare

Le risposte si restituiscono direttamente al client, e viene restituito in caso non si trovi ancora la risposta il reindirizzamento del server da contattare successivamente



il reindirizzamento del server da contattare successivamente



DNS CHACHING

Una volta che un name server ha appreso un'associazione la mette nella cache, i record nella cache vengono cancellati dopo un certo tempo. Migliora il ritardo e riduce il numero dei messaggi DNS

RECORD DNS

Database DNS: contiene resource records (RR, request)

RR format: (name, value, type, ttl)

Name e value dipendono da type,

Ttl indica quando la risorsa dovrà essere rimossa dalla cache

Type

- **A** : (AAAA per ipv6)
Name = HostName
Value = Indirizzo IP
- **CNAME** :
Name = HostName
Value = Nome canonico dell'host
- **NS** :
Name = nome di dominio ([unipi.it](#))
Value = Hostname dell'autoritative name server per quel dominio
- **CNAME** : (per server di Posta)
Name = nome di dominio
Value = Nome canonico del server di posta associato a name

(Dominio, Nome)

(cli.di.unip

[* il messaggio d
(nameserver.cli.c

(Alias, HostNa

(cli.di.unip

[* il messaggio d
(mailserver.cli.di.

Messaggi DNS

- UDP porta 53
- Permette TCP di solito tra zone molto distanti
- Identification : 16 bit
- Flag : Query 0 e Reply 1

Domande:

- campi per il nome richiesto e il tipo di query.

Risposte:

- RR nella risposta alla domanda.

Competenza:

- Record relativi ai serer di competenza.

Esempio:

12 bytes la parte gialla

DIG

dig (domain information groper) is a flexible tool for interrogating DNS name servers. It performs DNS lookups and displays the answers that are returned from the name server(s) that were queried. Most DNS administrators use dig to troubleshoot DNS problems because of its flexibility, ease of use and clarity of output

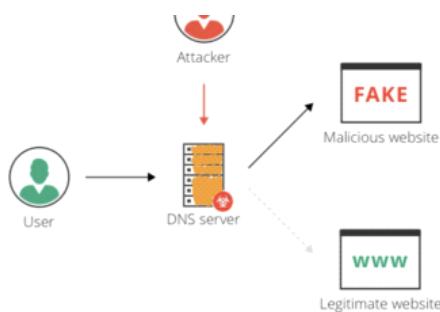
Dig [www.unipi.it](#) (MX/ANY (type)) @8.8.8.8

identification	flags
Quante query se ci sono number of questions	number of answer RRs
number of authority RRs	number of additional RRs
questions (variable number of questions)	
answers (variable number of resource records)	
authority (variable number of resource records)	
additional information (variable number of resource records)	

Il DNS è altamente decentralizzato per cui la query viaggia lungo una catena di server DNS prima di ottenere il risultato

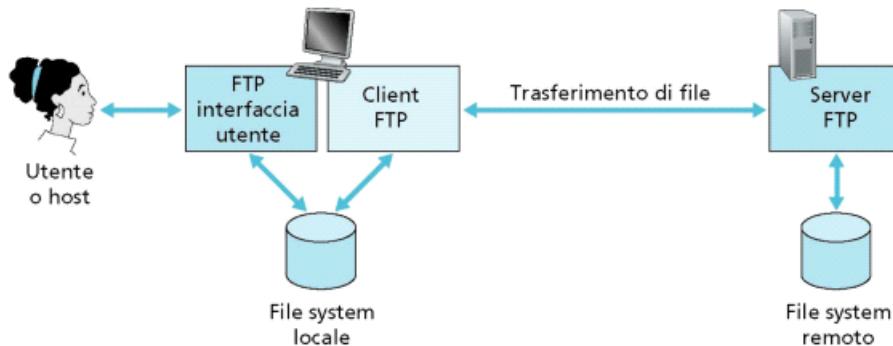
Il DNS hijacking è la pratica di restituire risposte non corrette alla query DNS reindirizzando il client verso siti malevoli

- Local Hijacking
- Router Hijacking
- Rogue Hijacking
- Man-in-the-Middle Attack



FTP

File Transfer Protocol, da un host remoto alla macchina locale, modello Client Server



Oltre al trasferimento di File (ovvero fare una copia in locale del file da trasferire, in caso di modifiche si ritrasferisce il file modificato) offre servizi di:

- Accesso interattivo: l'utente può navigare e modificare l'albero di directory nel file system remoto
- Specifica del formato dei dati da trasferire: file di testo, binari ecc
- Autenticazione: il client può specificare user e passw

Modello

Opera con due tipi di connessione (TCP)

Control connection: si scambiano i comandi e risposte tra client e server, utilizzando telnet

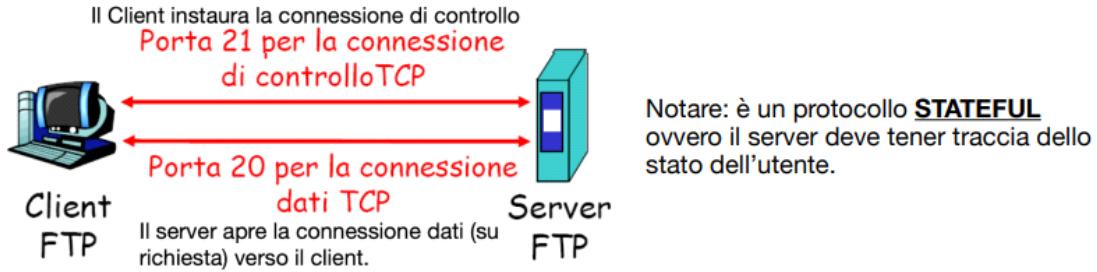
1. Il client contatta il server ftp sulla porta 21
2. Il client ottiene l'autorizzazione sulla connessione di controllo
3. Il client invia i comandi sulla connessione

La connessione di controllo è persistente, ovvero vengono mandati più comandi prima dell'interruzione della connessione

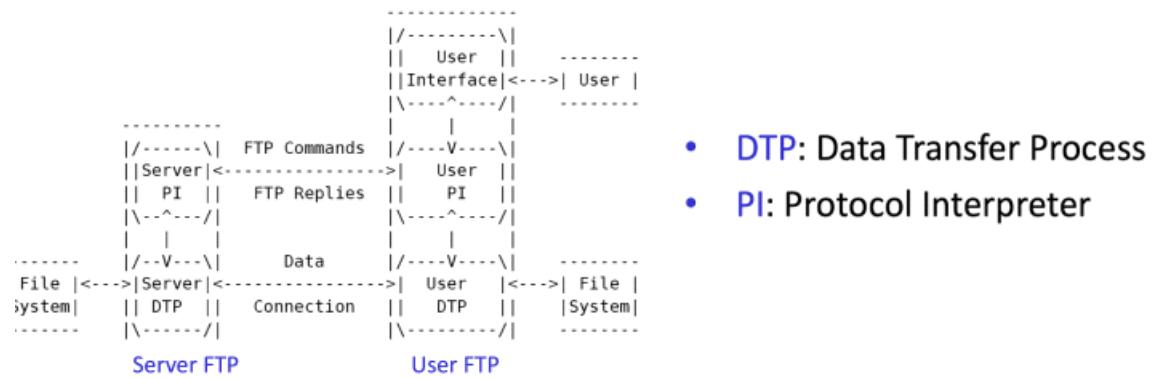
Data connection: si trasferiscono i dati con le specifiche della control connection

1. Il client chiede con un comando al server di trasferire un file
2. Il server apre una connessione dati TCP con il client (active mode)
3. Trasferisce il file
4. Dopo il completamento il server chiude la connessione

La connessione non è persistente



- La porta di uscita con cui il client attiva la connessione di controllo con il server è casuale mentre quella sul server è la 21.
- FTP usa la connessione di controllo per sincronizzare le porte assegnate dinamicamente tra client e server.
- Sulla connessione di controllo si parla per caratteri con una codifica NVT ASCII, sia per i comandi che per le risposte.
- I dispositivi dove risiedono il client e server ftp sono diversi (so, strutture file, diversi formati dei file), queste informazioni vanno tutte trasferite attraverso la porta di controllo.



- **DTP: Data Transfer Process**
- **PI: Protocol Interpreter**

- Active mode:
Il server apre una connessione dati TCP con il client conoscendo il numero di porta lato client, il client gliela comunica sulla connessione di controllo
- Passive mode
Il client chiede al server una porta per attivarci sopra la connessione dati così da far fare tutta la sincronizzazione al client

Modalità di trasmissione:

- Stream mode: ftp invia i dati a tcp con un flusso continuo di bit
- Block mode: ftp invia i dati a tcp suddivisi in blocchi, ogni blocco preceduto da un header.
- Compressed mode: si trasmette il file compresso

----- COMANDI :

Comandi

- USER username
- PASS password
- LIST elenca i file della directory corrente
- NLST richiede elenco file e directory (ls)
- RETR filename: recupera (get) un file dalla directory corrente
- STOR filename: memorizza (put) un file nell'host remoto

- ABOR interrompe l'ultimo comando ed i trasferimenti in corso
- PORT indirizzo e numero di porta del client
- SYST il server restituisce il tipo di sistema
- QUIT (quit): chiude la connessione

Codici di ritorno

- Codice di stato ed espressione (come in HTTP)
- 331 Username OK, password required
- 425 Can't open data connection
- 452 Error writing file
- 200 Comando OK

- 125 data connection already open; transfer starting
- 225 Data connection open
- 226 Closing data connection. Requested file action successful (for example, file transfer or file abort).
- 426 Connection closed; transfer aborted.
- 227 Entering Passive Mode (h1,h2,h3,h4,p1,p2).

Anonymous FTP

- server che supportano connessioni FTP senza autenticazione
- Tipicamente consentono di accedere solo ad una parte del file system e permettono solo un subset di operazioni (es. no PUT)
 - login anonymous
 - password guest (o e-mail)

Securing FTP with TLS (FTPS) RFC 4217

- Meccanismo che può essere usato da client e server FTP per implementare sicurezza e autenticazione usando il protocollo TLS (RFC 2246)

```
ftp
ftp> open ftp.ed.ac.uk
220 FTP Server
Utente (luther.is.ed.ac.uk:(none)):ftp
331 Anonymous login ok, send your complete email address as
your password
Password:
230 Anonymous access granted, restrictions apply
ftp> ls
200 PORT command successful
150 Opening ASCII mode data connection for file list
incoming
pub
INSTRUCTIONS-FOR-USING-THIS-SERVICE
edupload
226 Transfer complete
ftp: 65 bytes received in 0.03secondi 2.60Kbyte/sec)
```

TRASPORTO

mercoledì 13 ottobre 2021 11:09

LO STRATO DI TRASPORTO

Realizza una comunicazione logica fra processi residenti in host system diversi

- Logico: i processi si comportano come se gli host fossero direttamente collegati

Il compito dello strato di trasporto è quello di fornire servizi a quello di applicazione

L'applicazione sceglie lo stile di trasporto:

- Sequenza di messaggi singoli UDP
- Stream di byte TCP

Il livello di trasporto utilizza servizi del livello di rete

È un servizio fondamentale, da un'idea di comunicazione diretta tra gli host, è implementato solo sui sistemi terminali

Prende i dati, li formatta in segmenti o user datagram e li passa al livello di rete

- Servizio privo di connessione UDP

il livello di trasporto si occupa di mandare **messaggi** (al liv trasp destinatario), ogni messaggio viene incapsulato in un datagram e non è correlato con i messaggi prec e succ, non ha garanzia di consegna, se perdo un messaggio non lo recupero. I messaggi possono non arrivare in ordine, possono non fare lo stesso percorso. Il liv trasp li manda al livello applicativo così come gli arrivano

- Servizio orientato alla connessione TCP

stabilisce una connessione logica tra client e server

- **Servizi offerti:**

- Multiplexing/demultiplexing
 - Controllo degli errori (header + dati)

- **Il Protocollo TCP [RFC 793]**

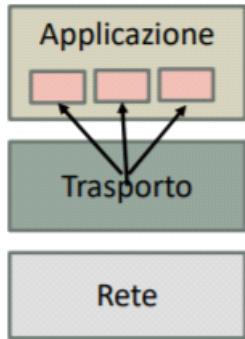
- Gestione della connessione
 - Consegnà affidabile (priva di errori, completezza e ordine)
 - Flow Control
 - Controllo di congestione

- **Il Protocollo UDP [RFC 768]**

- Senza connessione
 - Non affidabile, consegne senza ordine
 - Estensione “senza fronzoli” del servizio di consegna “host to host” di IP

Demultiplexing

Lo stato di trasporto provvede allo smistamento dei pacchetti tra la rete e le applicazioni

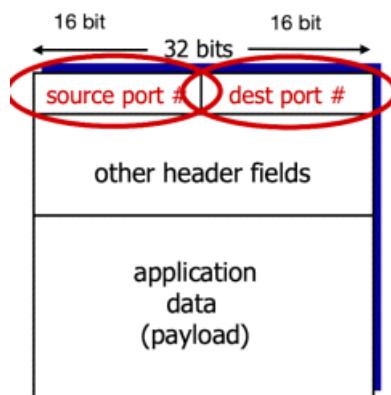


Multiplexing

Accorpamento dei flussi dati dai processi alla rete. "imbusta" i dati ricevuti con un preambolo

Le operazioni di multi e demultiplexing si basano sui socket address dei processi

Socket address: indirizzo ip e numero di porta



TCP/UDP segment format

l'host riceve il datagramma IP dal livello sottostante.

- Ogni datagramma ha un indirizzo IP sorgente e un IP destinatario
- Ogni datagramma trasporta un segmento di livello trasporto
- Ogni segmento contiene nell'header un numero di porto sorgente e un porta dest

Demultiplexing

nell'host ricevente:

consegnare i segmenti ricevuti
alla socket appropriata

Multiplexing

nell'host mittente:

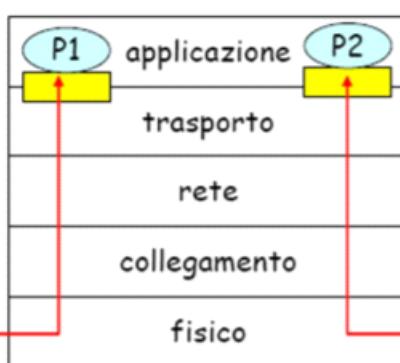
raccogliere i dati da varie
socket, incapsularli con
l'intestazione (utilizzati poi
per il demultiplexing)

= socket

= processo



host 1



host 2



host 3

Copyright 1996-2005 J.F Kurose and K.W. Ross

Porta:

Ogni comunicazione di trasporto viene identificata univocamente dalla coppia IP/porta degli host
La porta è un numero da 16 bit.

L'indirizzo IP è da 32 bit.

Il range di porte è organizzato con determinate regole:

- *System ports*: (Well Known Ports) 0 - 1023 porte assegnate dallo IANA a determinati servizi (es porta HTTP 80) identificano processi server
- *User ports (registered ports)* 1024 - 49151 porte registrate per un servizio (es porta MySQL è in questo range) assegnate da IANA
- *Dynamic Ports (Private o Ephemeral Ports)* porte per l'indirizzo del mio client, non assegnate da IANA

Il SO assegna dinamicamente le porte ai processi che ne fanno richiesta

SERVICE NAME	PORt/PROTOCOL	DESCRIPTION
ftp-data	20/tcp	File Transfer [Default Data]
ftp	21/tcp	File Transfer [Control]
ssh	22/tcp	SSH Remote Login Protocol
telnet	23/tcp	Telnet
smtp	25/tcp	Simple Mail Transfer
domain	53/tcp	Domain Name Server
www-http	80/tcp	World Wide Web HTTP
pop3	110/tcp	Post Office Protocol - Version 3
nntp	119/tcp	Network News Transfer Protocol
imap3	220/tcp	Interactive Mail Access Protocol v3
mysql	3306/tcp	MySQL server
tftp	69/udp	Trivial File Transfer Protocol
Domain	53/udp	Domain Name Server
RIP	520/udp	Routing Information Protocol
snmp	161/udp	Simple Network Manag. Prot.

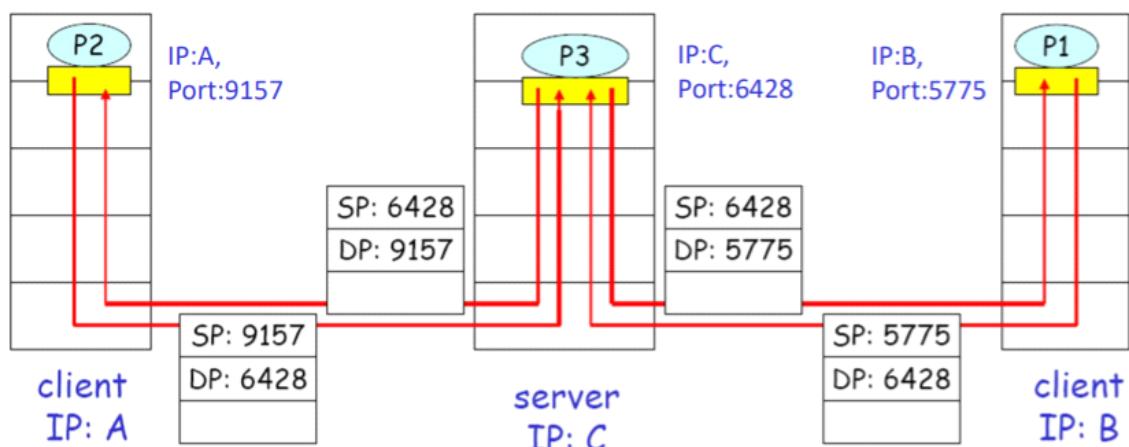
Connectionless Demultiplexing UDP

Creo una socket (DatagramSocket mySocket = new DatagramSocket())

Socket UDP identificata da coppia (IP, porta)

Lo stato di trasporto dell'host ricevente consegna il segmento UDP alla socket

I datagram con stesso IP e porta destinatario uguale vengono consegnati alla stessa porta
(anche con mittente diverso)



Demultiplexing orientato alla connessione (TCP)

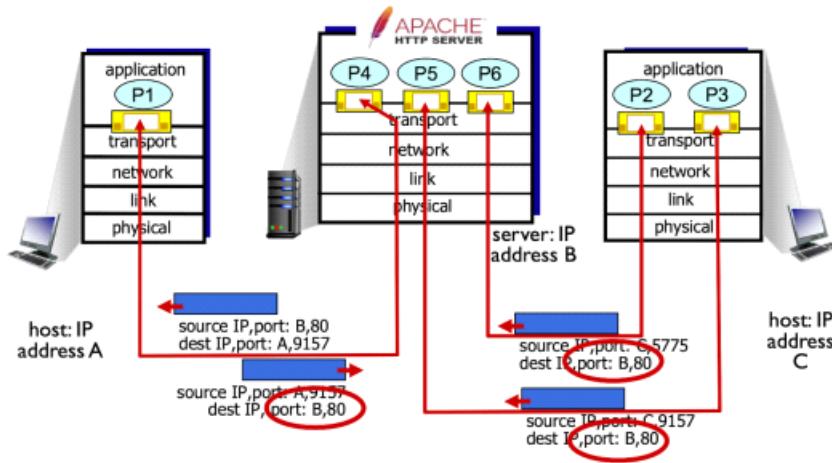
Due processi che comunicano con una connessione logica instaurata dal livello di trasporto

4 parametri:

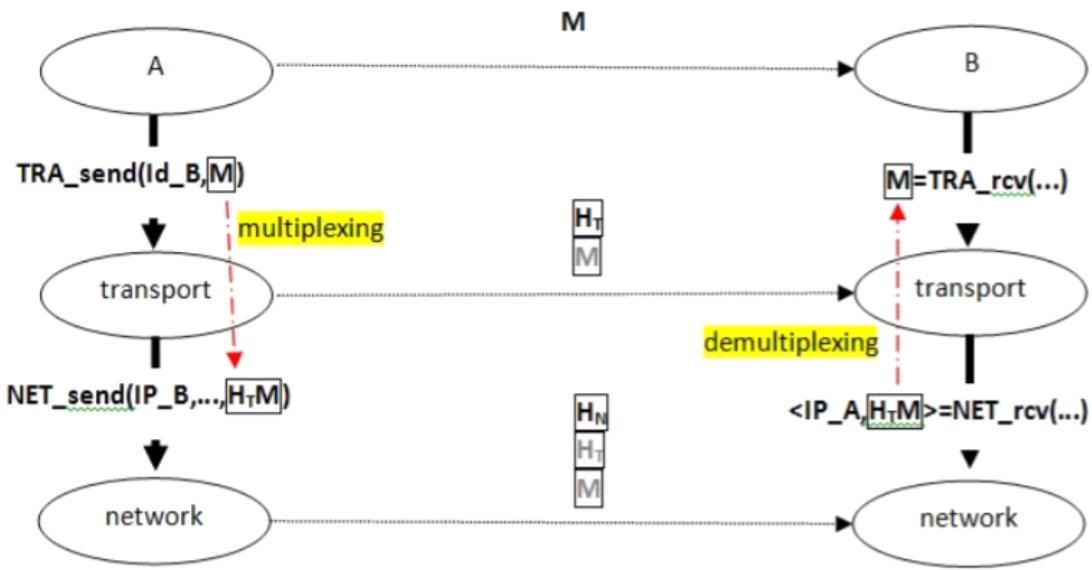
- IP origine
- Numero porta origine
- IP destinatario
- Numero porta destinatario

L'host ricevente usa i 4 parametri per inviare il segmento alla socket appropriata

Un host può supportare più socket contemporanee



Interfaccia tra i livelli dello stack



TCP

Quello che fa il TCP è prendere un flusso di byte indefinito a priori e mandarlo al destinatario che lo manda al processo destinatario

Flusso di byte ordinati ma non strutturati

Cosa c'è nel flusso di byte non interessa a TCP

Il processo applicativo poi mette in ordine i byte e ci opera riorganizza e raggruppa i byte.

- Orientato alla connessione

I nodi intermedi non sono informati della connessione.

La connessione viene instaurata tramite **handshake** e si preparano strutture dati per gestire la connessione

Il client è quello che per primo stabilisce la connessione, non è detto che sia lui che inizia a mandare i dati

- Connessione full duplex

I dati possono viaggiare in entrambe le direzioni, contemporaneamente .

Le due direzioni sono slegate. La connessione, grazie al demultiplexing è punto-punto (abbiamo IP e

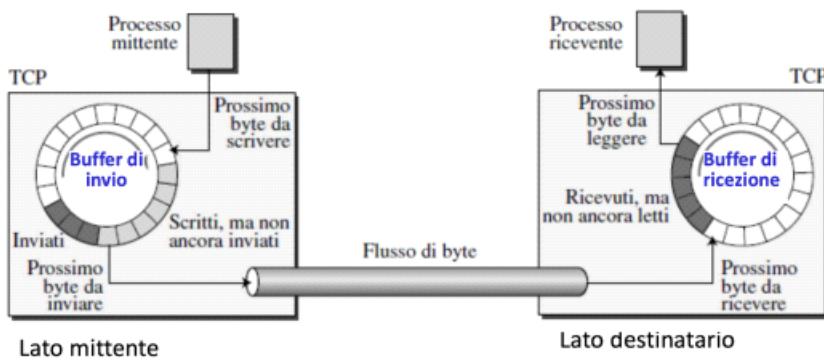
porta)

Connessione punto-punto, 1 a 1

Servizi del TCP:

- Flusso continuo di dati
- Trasferimento bidirezionale
- Multiplexing/demultiplexing
- Garantisce un trasferimento dati ordinato e affidabile, offre un servizio di controllo degli errori (dati corrotti, segmenti persi, duplicati, fuori sequenza)
Meccanismi di inizio e fine trasmissione (*controllo di sessione*)
- Controllo di flusso: non manda più dati di quelli su cui il processo dest riesca a operare
- Controllo di congestione: ha lo scopo di recuperare situazioni di sovraccarico della rete

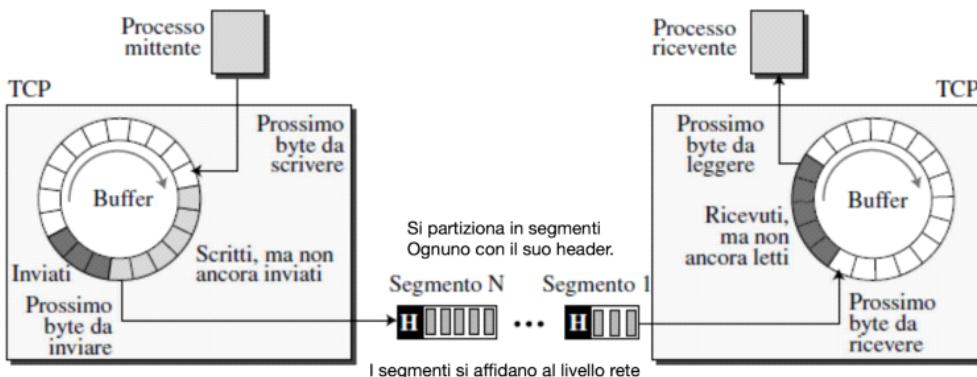
Per il trasferimento dati si utilizzano dei buffer, i byte che arrivano vengono messi in un buffer e quando ritiene opportuno invia i byte in rete (in un segmento), ottimizza il trasferimento



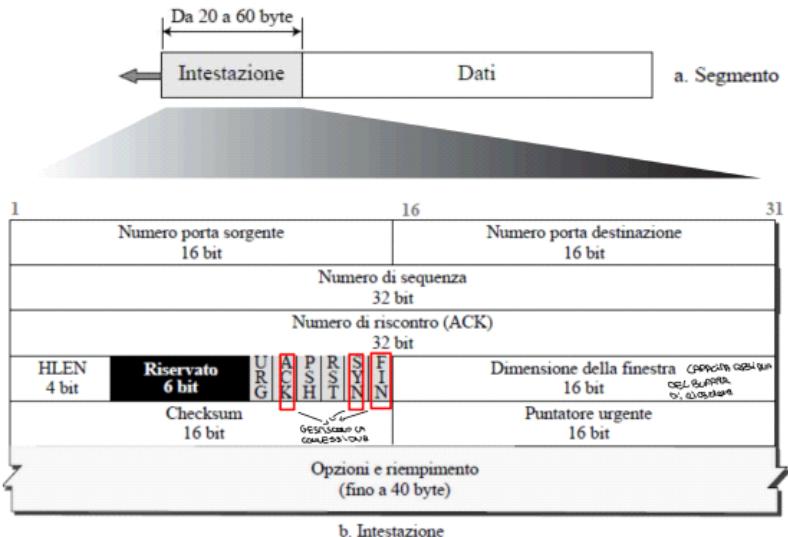
SEGMENTI TCP

Il flusso viene ordinato in segmenti in cui:

- Ogni segmento ha il suo header
- Ogni segmento viene consegnato al livello IP



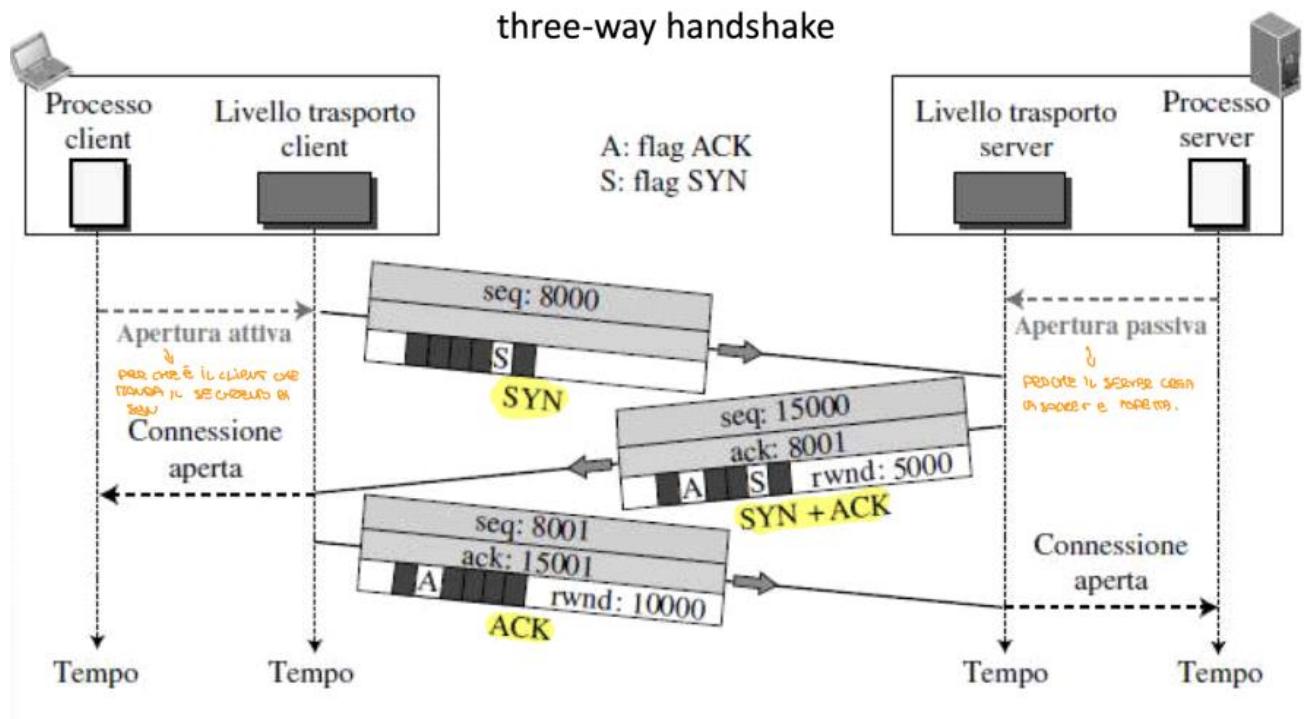
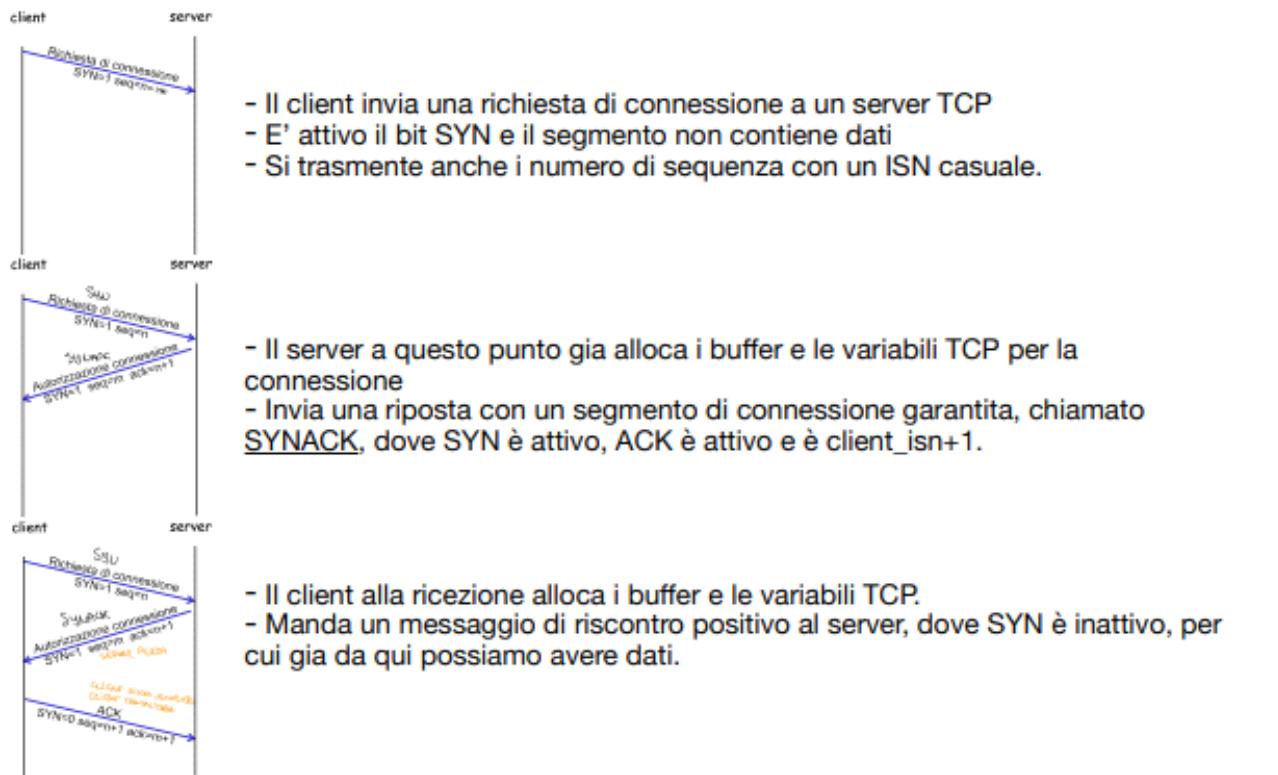
FORMATO SEGMENTI



- Numero di sequenza, numero di riscontro, finestra: flow control, ritrasmissione e riordino dei pacchetti
- Porta 16 bit: numero di porta della comunicazione
- Numero di sequenza (32bit): numero nello stream del primo byte di dati di questo segmento, se SYN è settato il numero è ISN (Initial Sequence Number) e il primo byte di dati è ISN + 1
- Numero di riscontro (32bit): se ACK è settato, contiene il valore del prossimo numero di sequenza che il mittente si aspetta di ricevere dall'altro host
- HLEN (4bit): la lunghezza dell'header TCB espressa in parola da 4 byte
- Bit codice: 6 flag per:
 - o URG: il campo di puntatore urgente contiene dati da trasferire in via prioritaria
 - o ACK: il campo numero di riscontro contiene dati significativi
 - o PSH: funzione Push (trasferimento di dati dal segmento al livello applicativo immediato)
 - o RST: reset della connessione
 - o SYN: sincronizza il numero di sequenza
 - o FIN: non ci sono altri dati dal mittente, chiusura della connessione
- Finestra di ricezione (16 bit): il numero di byte di dati a partire da quello indicato dal campo num riscontro che il mettente è in grado di accettare, serve per il controllo del flusso
- Checksum (16b): checksum per l'intero segmento per rilevare errori
- Opzioni (facoltativo, lunghezza max 40 byte): negoziazione di vari parametri, ad es. dimensione massima segmento (MSS), selective acknowledgement supportato e blocchi di dati riscontrati selettivamente. Le opzioni sono sempre multipli di 8 bit e il loro valore è considerato per il calcolo del checksum

HANDSHAKE

Three way handshake



CHIUSURA DELLA CONNESSIONE

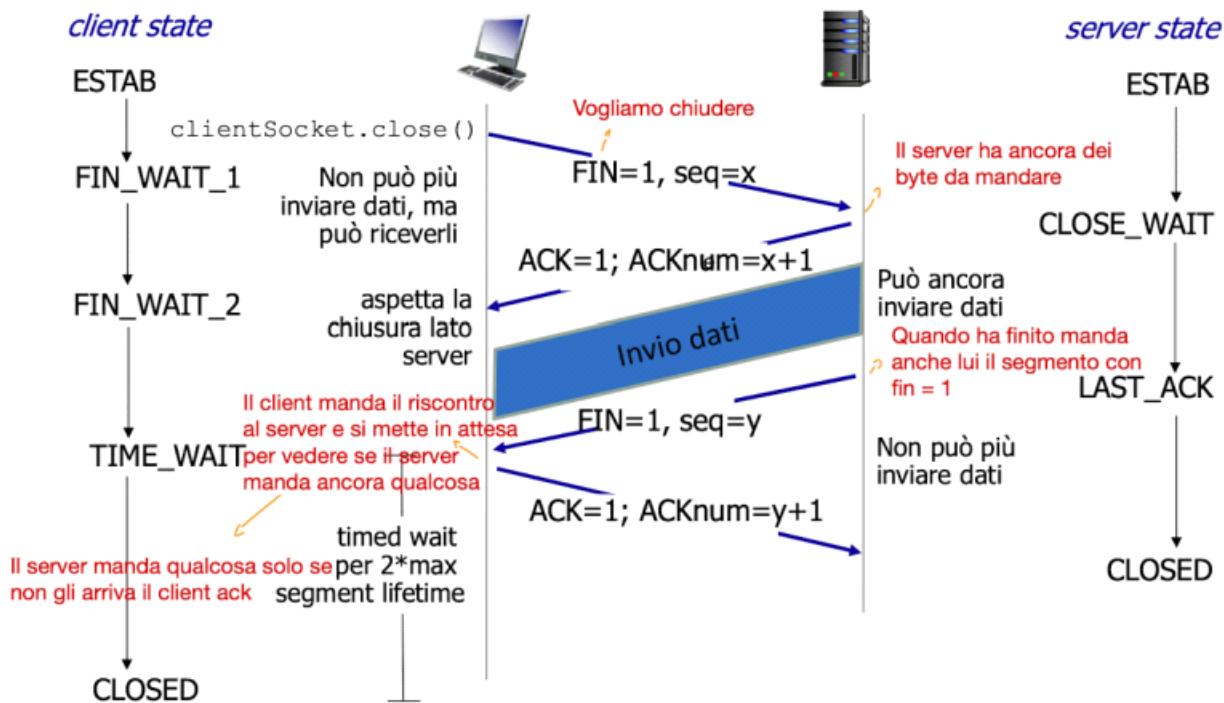
Sia il client che il server chiudono il loro lato della connessione visto che è bidirezionale.

Si chiude inviando il segmento con FIN = 1

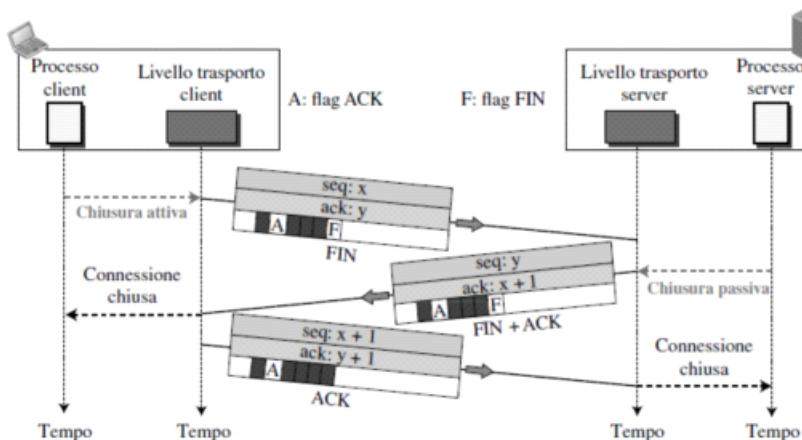
Si risponde al FIN con ACK

Time wait è lo stato finale dove prima di chiudere la connessione si aspetta per due volte MSL (maximum segment lifetime) ovvero la stima del massimo periodo di tempo che un pacchetto IP può vivere sulla rete.

Viene usato dal protocollo per implementare in maniera affidabile la terminazione della connessione, se ACK viene perso chi esegue la chiusura passiva manderà un ulteriore FIN, mentre chi ha perso il pacchetto rimanda ACK

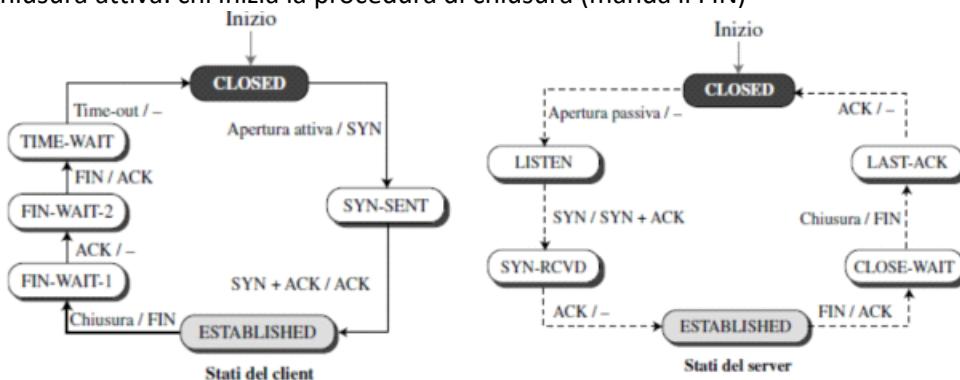


NB. MSL è 2 minuti secondo RFC, questo valore può variare
Per verificare su sistemi Linux `svsctl net.inet.tcp.msl`



- Un FIN che non trasporta dati consuma comunque un numero di sequenza
- Idem per FIN+ACK

Chiusura attiva: chi inizia la procedura di chiusura (manda il FIN)



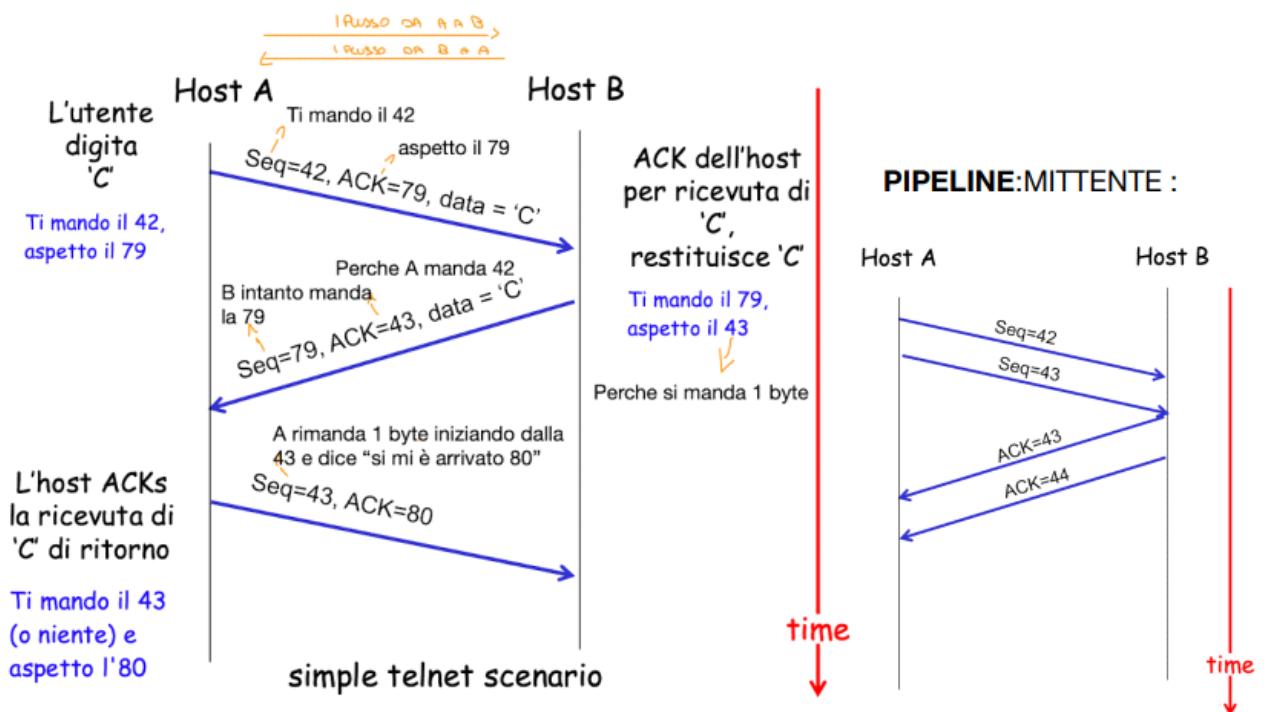
- LISTEN - represents waiting for a connection request from any remote TCP and port
- SYN-SENT - represents waiting for a matching connection request after having sent a connection request.
- SYN-RECEIVED - represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request.

- ESTABLISHED - represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.
- FIN-WAIT-1 - represents waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.
- FIN-WAIT-2 - represents waiting for a connection termination request from the remote TCP.
- CLOSE_WAIT - represents waiting for a connection termination request from the local user.
- CLOSING represents waiting for a connection termination request acknowledgment from the remote TCP.
- LAST-ACK - represents waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).
- TIME-WAIT - represents waiting for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request.
- CLOSED - represents no connection state at all.

TRASFERIMENTO AFFIDABILE

Una volta instaurata la connessione può accadere che alcuni segmenti vengano smarriti o corrotti. Il checksum serve per scartare i segmenti corrotti

- *Numero di sequenza*: numero del primo byte del segmento nel flusso di byte
- *Numero di riscontro*: numero di sequenza del byte che l'host attende dall'altro
- *Riscontro cumulativo*: si fa il riscontro dei byte fino al primo byte mancante nel flusso
- *Timer*: per capire quando c'è stata una perdita

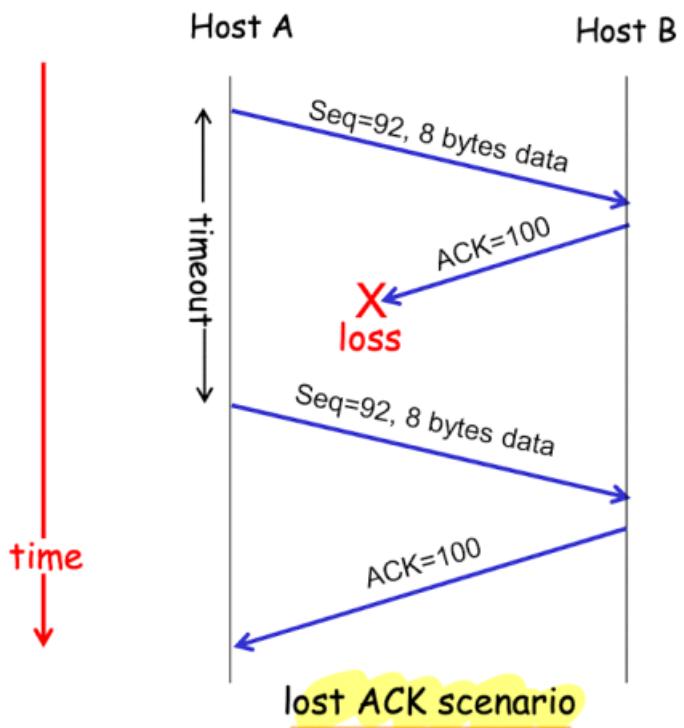


EVENTI MITTENTE

1. TCP riceve i dati dell'applicazione
2. Incapsula i dati in uno o più segmenti e assegna il numero di sequenza
3. Avvia il timer di ritrasmissione RTO

Si ritrasmette in caso di:

- Timeout: ritrasmette il segmento del quale non è arrivato l'ack che ha causato il timeout
- Ricezione di 3 ACK duplicati: si ritrasmette prima della scadenza del rimer perché significa che il segmento dopo l'ultimo riscontrato è andato perso (ritrasmissione veloce)



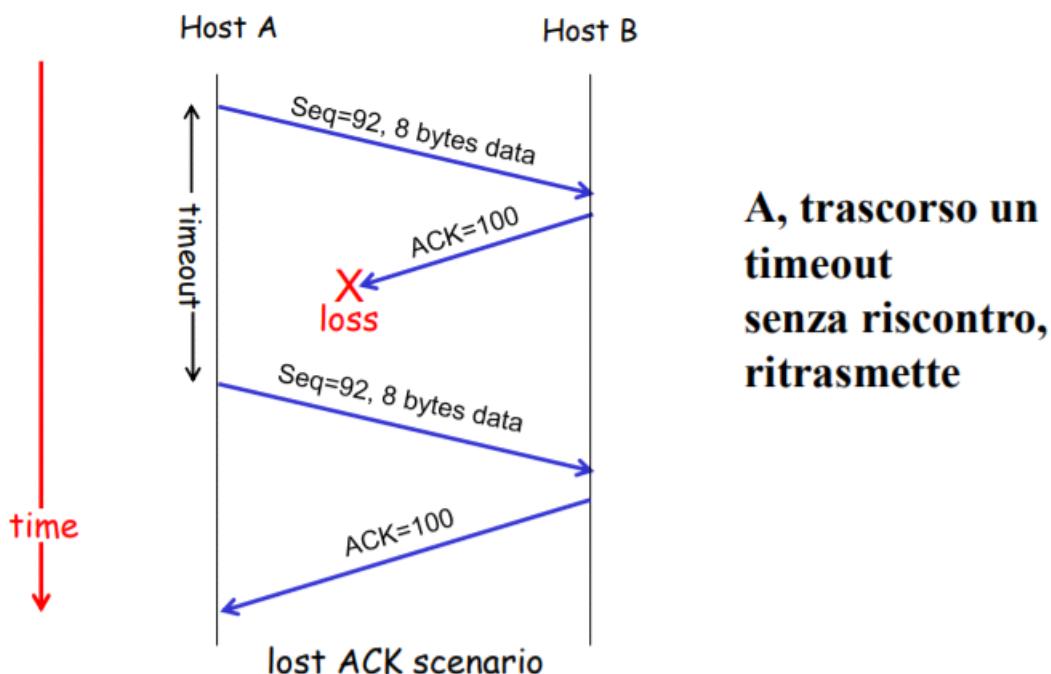
Timeout: in caso di timeout si ritrasmette il segmento che ha causato il timeout e si riavvia il timer
 Ack duplicato: ricevo una volta un ack con numero di riscontro es. 500, lo ricevo altre 2 volte,
 significa che ho perso il segmento successivo a quello riscontrato, senza aspettare che scada il timer,
 eseguo una **ritrasmissione veloce** (fast retransmission)

Segmenti fuori sequenza:

I dati possono arrivare fuori sequenza ed essere temporaneamente memorizzati dall'entità TCP destinataria.

Il TCP non dice come il destinatario deve gestire i pacchetti fuori sequenza, dipende dall'implementazione

Ritrasmissione dovuta a riscontro perso



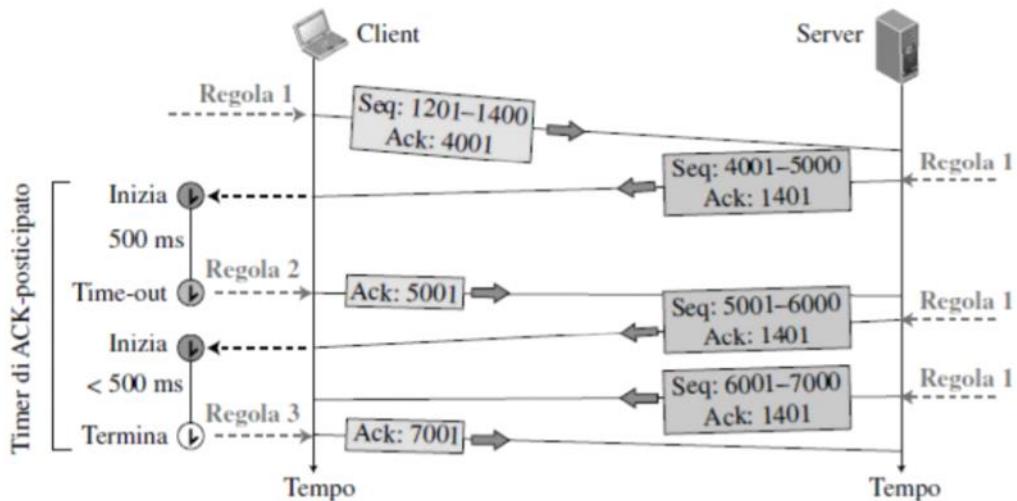
43

LATO DESTINATARIO

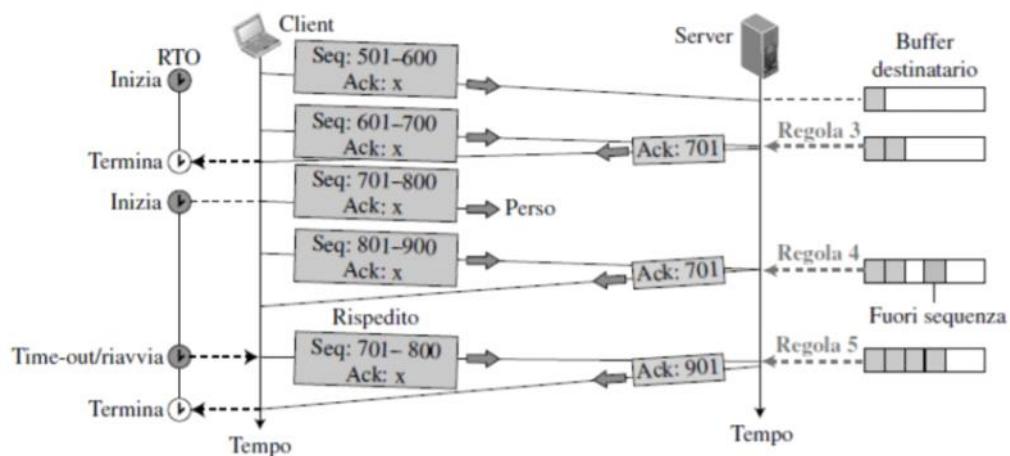
1. Se il destinatario ha dei dati da trasmettere può trasmettere dati e mandare anche l'ACK di quello che ha ricevuto
2. Se il destinatario deve riscontrare dei segmenti, ma non ha nulla da mandare e riceve segmenti in ordine, prima di mandare l'ack ritarda 500ms, se entro quel tempo arriva un altro segmento può poi mandare l'ack, così manda l'ack di più segmenti
3. Se il destinatario riceve un segmento atteso e quello precedente non è arrivato allora invia subito l'ack
4. Se dest riceve un segmento fuori sequenza oppure mancante o duplicato manda subito ack

Operatività normale

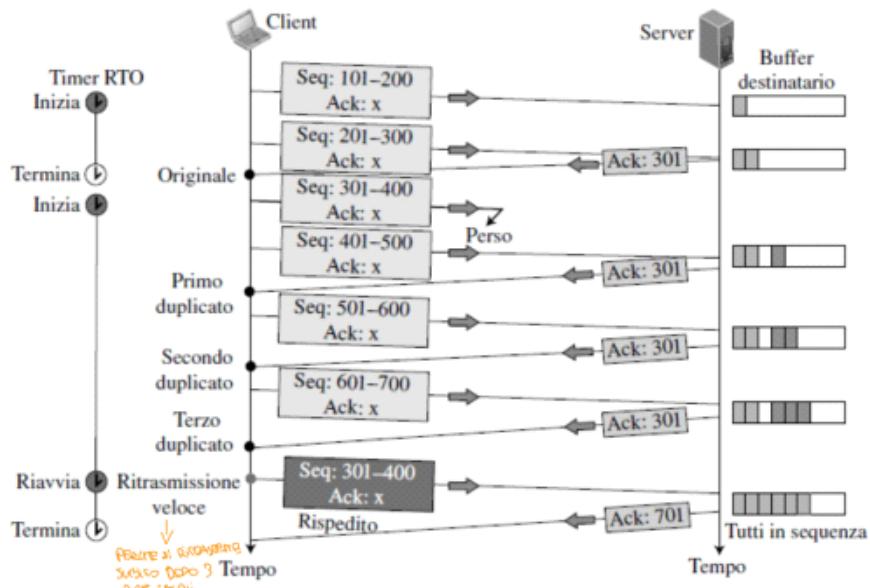
(1) Tutti i segmenti contenenti dati includono ACK
 (2) Se destinatario non ha dati da inviare e riceve segmento «in ordine» ritarda invio ACK di 500ms a meno che non riceva nuovo segmento
 (3) Se destinatario riceve segmento atteso e precedente non è stato riscontrato allora invia immediatamente ACK
 Se destinatario riceve (4) segmento fuori sequenza oppure (5) «mancante» oppure (6) duplicato allora invia immediatamente un segmento ACK (indicando prossimo numero atteso)



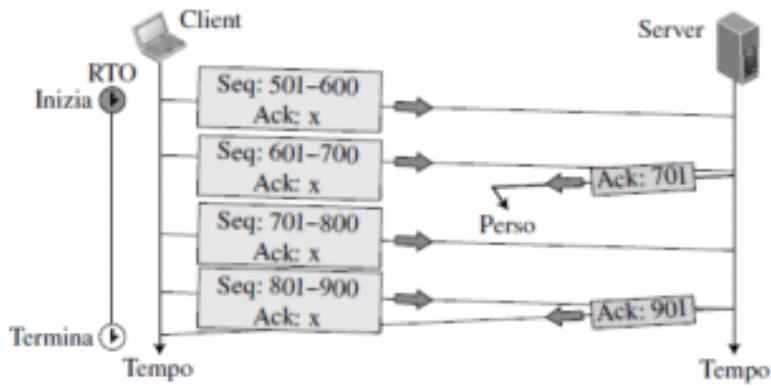
Segmento smarrito:



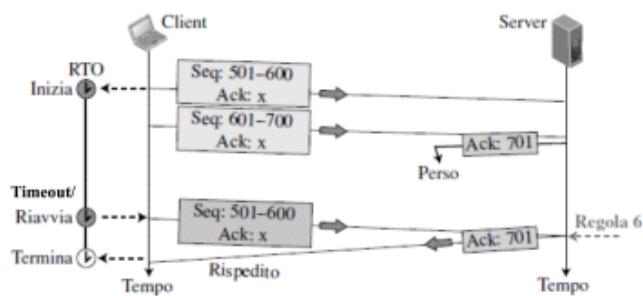
Ritrasmissione veloce



Riscontro smarrito

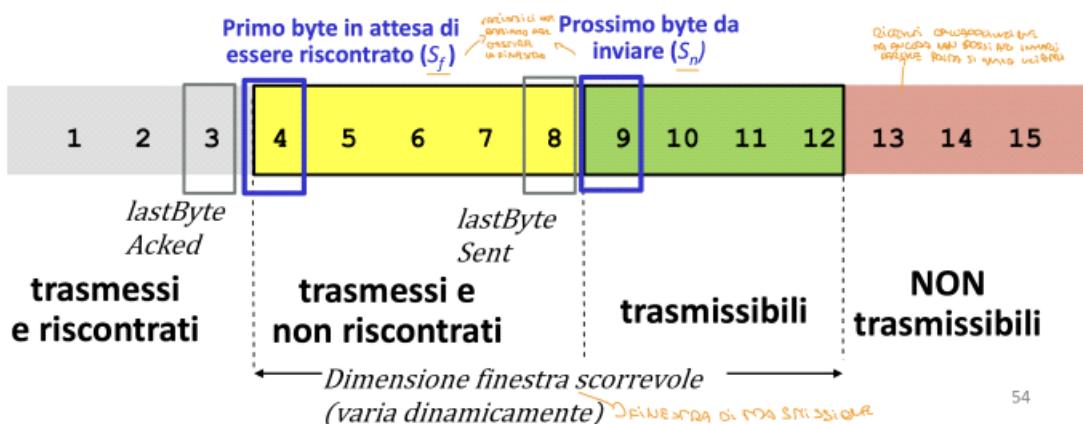


Riscontro perso corretto da ritrasmissione



FINESTRA DI TRASMISSIONE

Si mantengono i dati nel buffer, questo buffer viene scorso dalla finestra di trasmissione (sliding windows), che ci dice quanti dati possiamo trasmettere in un determinato istante, è dinamica, viene sovrapposta ai dati, a ogni ack si aumenta.

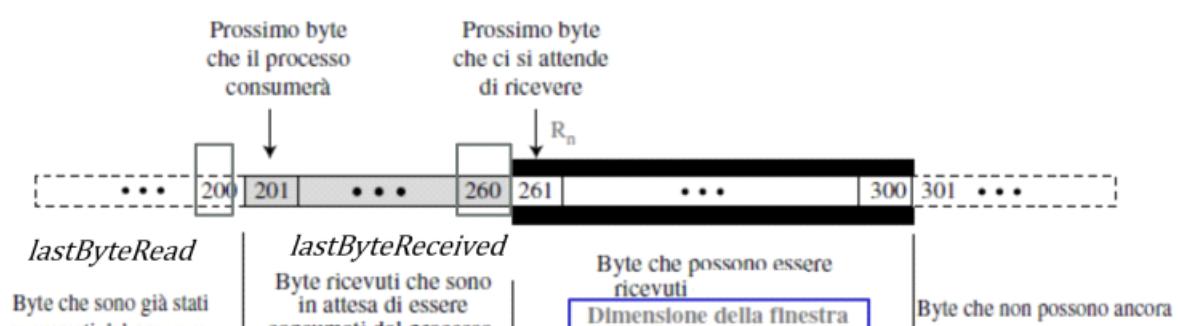


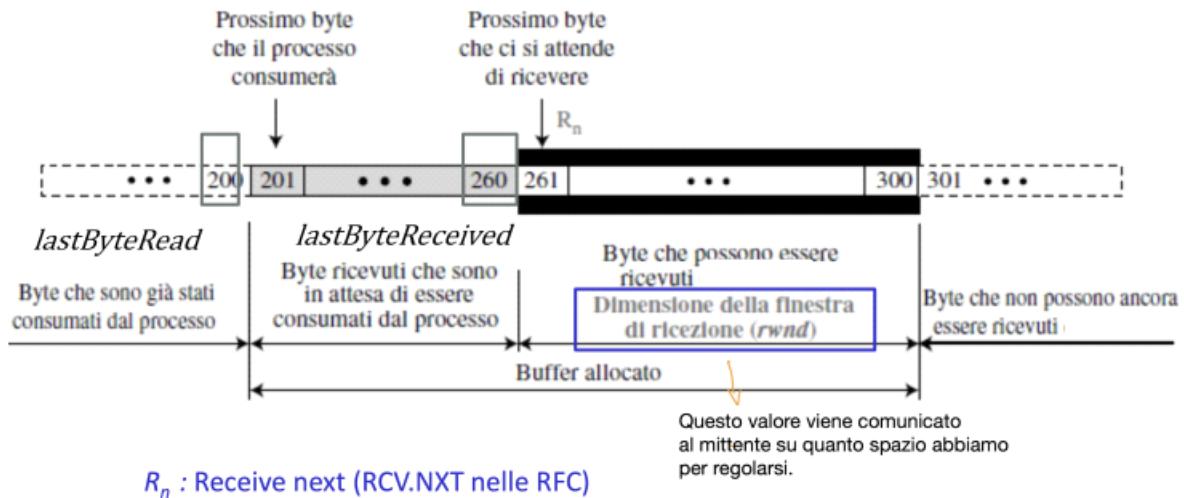
54

Sf (SendFirst): numero di sequenza del primo byte in attesa di essere riscontrato

Sn (Send Next): prossimo byte da inviare (prossimo numero di sequenza da inviare)

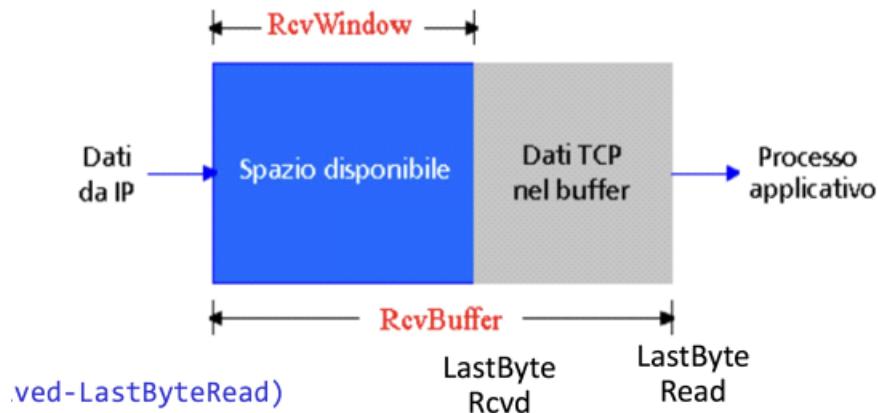
FINESTRA DI RICEZIONE





CONTROLLO DI FLUSSO

Vogliamo evitare di saturare il buffer di ricezione de destinatario dal mittente



Il destinatario calcola dinamicamente:

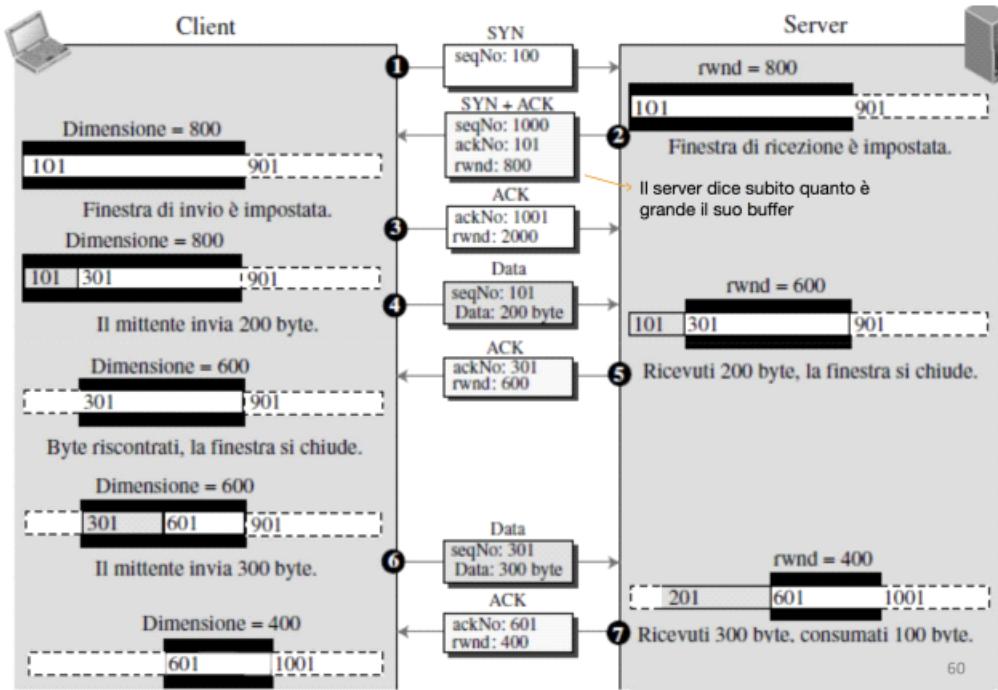
$$Rwnd = RcvBuffer - (LastByteReceived - LastByteRead)$$

Il destinatario manda la RWND al mittente, questo assicura che

$$LastByteSent - LastByteAcked \leq Rwnd$$

Se RWN = 0 vuol dire che il dest sta smaltendo i pacchetti ed è pieno, ogni tanto il mittente manda dei pacchetti sonda per ricevere aggiornamenti

Esempio



Controllo di congestione

Cerca di capire se nella rete c'è qualche problema e prova a rallentare la trasmissione, si verifica quando si richiedono e si vogliono mandare più pacchetti di quelli che la banda di trasmissione è capace.

In funzione della congestione percepita viene regolata la velocità di trasmissione.

TCP si adatta alla velocità della rete. Se percepisce scarso traffico, aumenta la frequenza di invio, altrimenti la diminuisce

- Receiver window: dipende dalla dimensione del buffer di ricezione
- Congestion window: basata su una stima della capacità della rete

I byte trasmessi corrispondono alla dimensione della finestra più piccola

Cwnd: finestra di congestione.

La rete di invio non può superare $\min(rwnd, cwnd)/RTT$. Dimensione finestra di invio: $\min(rwnd, cwnd)$

Modi per regolare la velocità:

- SLOW START
- AIMD: incremento additivo e decremento moltiplicativo (Additive increase multiplicative decrease)
- FAST RECOVERY: ripresa veloce

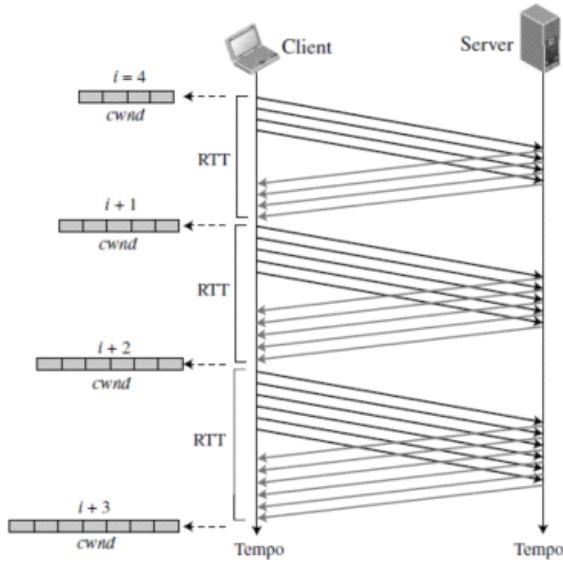
CONGESTIONE / CWND

- Si misura in MSS, ovvero **Maximum Segment Size**: quantità massima di dati trasportabili in un segmento (misura del payload)
- MSS dipende da MTU che dipende dalla tecnologia di collegamento
- Si sceglie in modo tale che il segmento TCP incapsulato in un datagramma IP stia dentro un Frame di collegamento
Esempi tipici di MSS 1460, 536, 512 byte (tolgo dall'MTU 20 byte header TCP + 20 byte header IP)
- RTT (Round Trip Time) è il tempo impiegato da un segmento di effettuare il percorso di andata e ritorno.

CONGESTIONE / AIMD

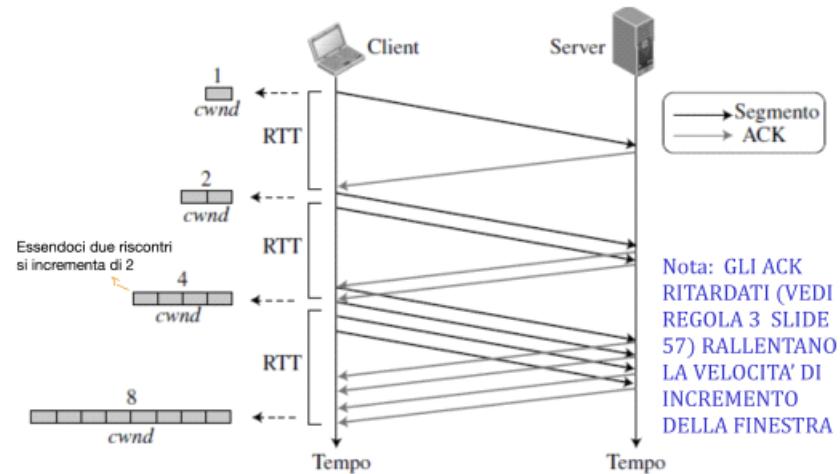
Ad ogni ACK ricevuto dal destinatario si aumenta di 1 MSS per ogni RTT (**congestion avoidance**)
Crescita proporzionale (lineare); per questo si divide, ad ogni evento di perdita si DIMEZZA la propria CWND

- **ACK:** $cwnd = cwnd + 1/cwnd$



CONGESTIONE / SLOW START

Si parte da $cwnd = 1$ MSS e ad ogni RTT si aggiunge una quantità pari al MSS. Incremento congwin di 1 MSS ad ogni ack non duplicato. Quindi la CWND raddoppia ad ogni RTT



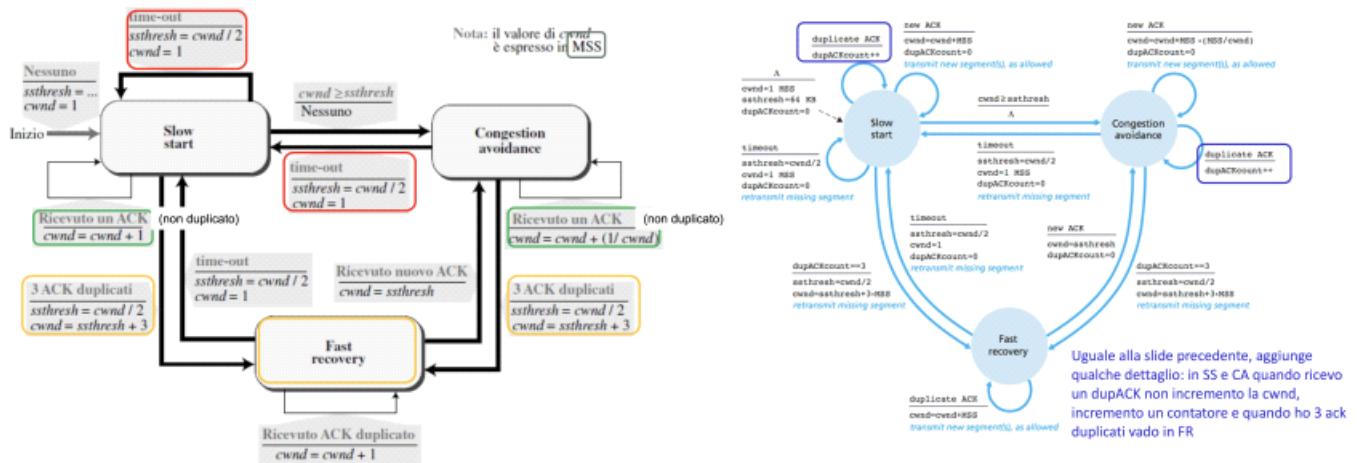
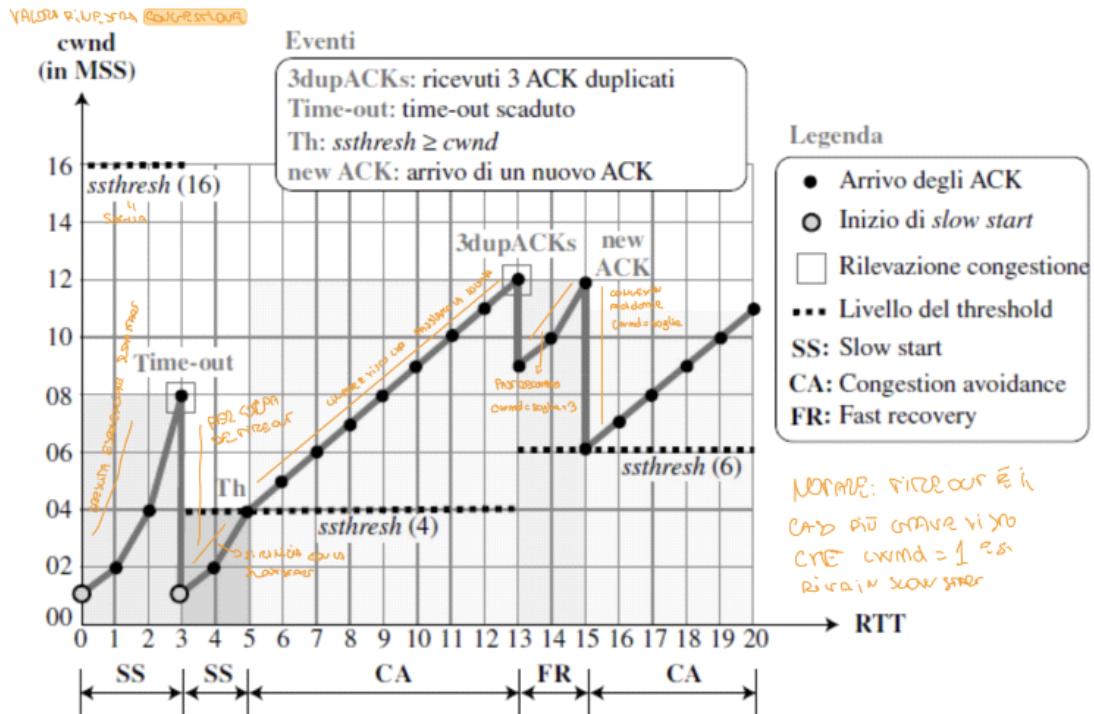
CONGESTIONE / TCP RENO

Mettiamo insieme slow start e AIMD, si ha una variabile soglia, con un valore alto. La soglia determina quando termina la slow start e inizia la fase di congestion avoidance (additive increase AI)

- Se $cwnd < \text{soglia}$, $cwnd$ aumenta esponenzialmente (slow start)
- Se $cwnd > \text{soglia}$, $cwnd$ aumenta linearmente (additive increase)

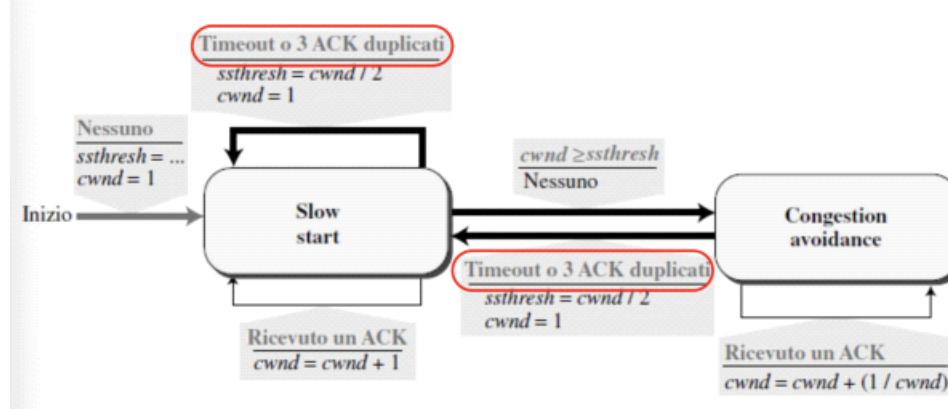
Dopo un evento di perdita:

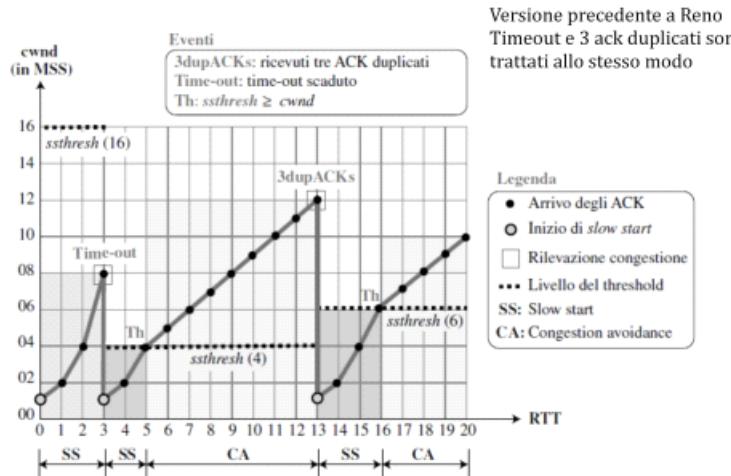
- 3 ACK duplicati: $\text{soglia} = cwnd/2$, $cwnd = \text{soglia} + 3\text{MSS}$ (*fast recovery*)
 - o Finché arrivano ack duplicati $cwnd = cwnd + 1$
 - o Se timeout si va in slow start
 - o Se arriva un nuovo ack non duplicato si va in AIMD e $cwnd = \text{soglia}$
- Timeout: $\text{soglia} = cwnd/2$, $cwnd = 1\text{ MSS}$ (*slow start*)



CONGESTIONE / TCB TAHOE

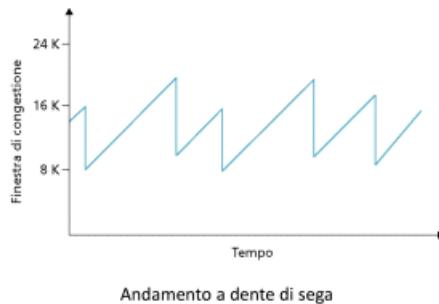
- Slow start e congestion avoidance
- Timeout e 3 ack duplicati gestiti allo stesso modo
 - Soglia = $cwnd/2$ e $cwnd = 1$ (si va in slow start)





Andamento della finestra di congestione:

- Nella maggior parte dei casi la **congestione** viene rilevata tramite 3 ack duplicati
- Trascurando slow start e Fast recovery -> **incremento additivo** (finestra aumenta di $1/cwnd$ ad ogni riscontro non duplicato) e **decremento moltiplicativo** ($cwnd$ dimezza)



TCP: throughput

Partendo dalle considerazioni dell'immagine sopra sull'andamento macroscopico della finestra Indicando con W il valore massimo in byte della finestra (ovvero quando si verifica l'errore), TCP in regime stazionario offre il seguente throughput medio (frequenza di invio media)

$$\text{MeanThroughput} = (0,75 * W) / \text{RTT}$$

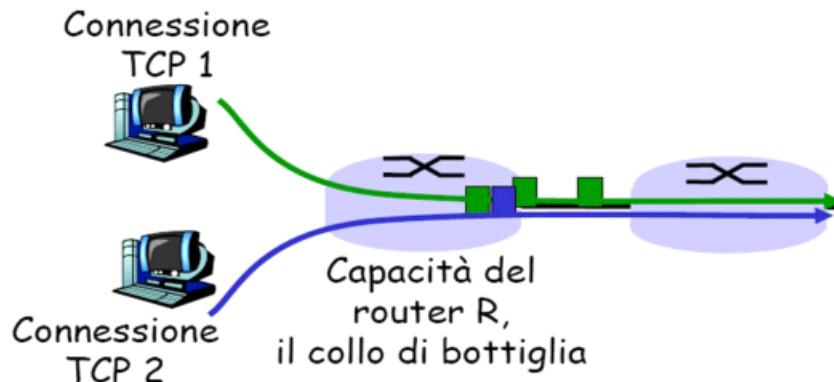
Equità (fairness)

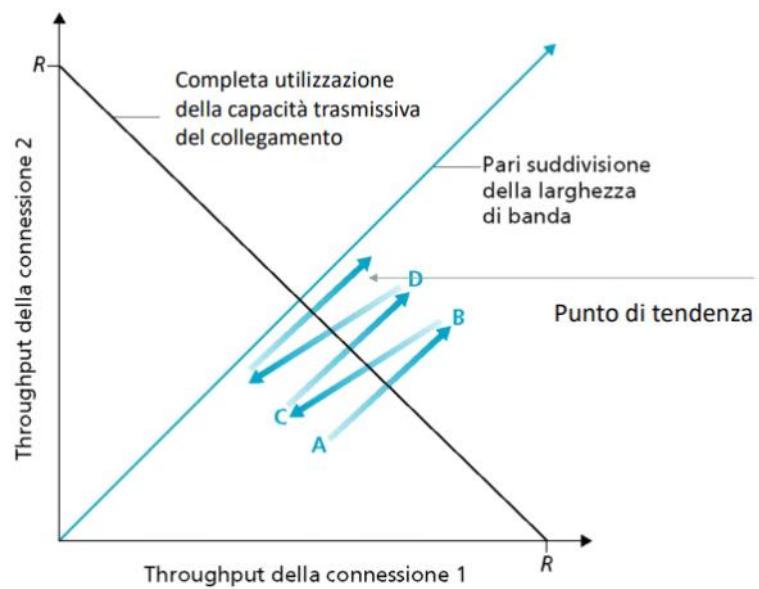
Ipotesi:

- K connessioni TCP insistono su un unico link di capacità R bit/s
- Le connessioni hanno gli stessi valori di MSS e RTT
- Non ci sono altri protocolli che insistono sullo stesso link

Risultato:

- Ognuna delle connessioni TCP tende a trasmettere R/K bit/s





UDP

lunedì 8 novembre 2021 11:23

UDP

Servizio di consegna best effort

- I datagramma UDP possono essere perduti o consegnati fuori sequenza
- L'affidabilità può essere aggiunta al livello applicazione

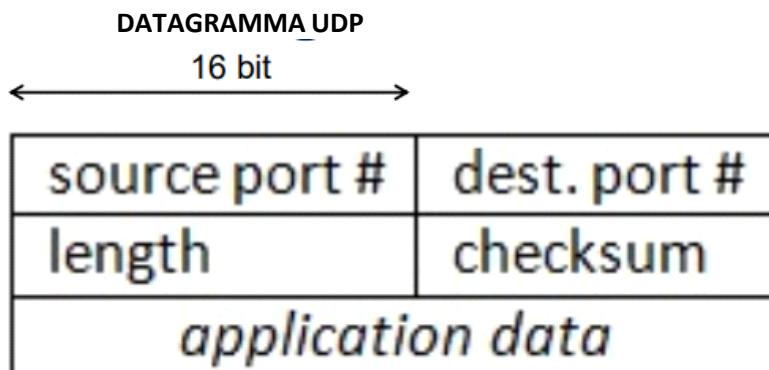
Orientamento al messaggio

- Ogni datagramma UDP è indipendente dall'altro
- I processi devono inviare messaggi di dimensioni limitate, che possono essere incapsulati in un datagramma UDP

Meno complesso di TCP, offre meno servizi, ma più indicato per contesti dove occorre completo controllo della temporizzazione (app time sensitive come lo streaming)

Proprietà:

- Nessuna connessione (connectionless)
 - o Intestazioni di 8 byte
 - o Senza controllo della congestione e di flusso
- Checksum facoltativo
- Facile e leggero da gestire
- Utilizzato spesso in applicazioni multimediali
 - o Tolleranza piccole perdite
 - o Sensibilità alla frequenza
- Altri impieghi: DNS, SNMP



- 8 byte di intestazione
- Porta: numeri di porta della comunicazione (per il demultiplexing è usato solo quello di destinazione)
- Lunghezza del messaggio: lunghezza totale (header + dati) del datagramma UDP (16 bit -> max 65535 byte)

Datagramma UDP: campi

- **Checksum (opzionale)**: checksum dell'intero datagramma, opzionale per il controllo errore end-to-end, il pacchetto corrotto viene scartato senza notifica la mittente. Calcolato sull'intero datagramma UDP più lo pseudo header (header IP)

Pseudoheader – usato per calcolo checksum

0	8	16	31
Indirizzo IP di Provenienza			
Indirizzo IP di Destinazione			
Zero	Proto (17)	Lunghezza UDP	

Calcolo del checksum UDP

Mittente:

- Campo checksum a 0
- Tratta il contenuto del datagramma UDP come una sequenza di parole da 16 bit
- Checksum: somma le parole di 16 bit in complemento a 1
- Complemento a uno del risultato della somma
- Il mittente pone il valore della checksum nel campo checksum del datagramma UDP

Ricevente

- Calcola il checksum del segmento ricevuto
- Il valore del checksum è 0?
 - Sì: nessun errore
 - No: errore si scarta il messaggio

SERVIZI FORNITI DAI PROTOCOLLI DI TRASPORTO

UDP:

- Trasferimento non affidabile tra mittente e destinatario
- Non fornisce:
 - Setup della connessione
 - Affidabilità
 - Controllo di flusso
 - Controllo della congestione
 - Timing
 - Garanzia di banda
- Richiede minor overhead

TCP:

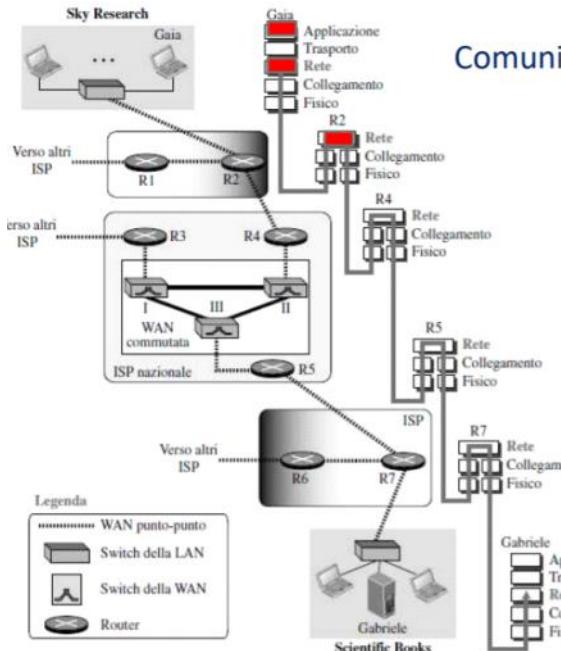
- *Connection-oriented*: handshake tra client e server
- *Trasporto affidabile* tra mitt e dest
- *Controllo di flusso*: il mittente non saturerà il destinatario
- *Controllo della congestione*: limita il mittente quando la rete è sovraccarica
- *Non fornisce*: timing, banda minima garantita

TCP offre un trasporto a stream, quindi si possono leggere da input quanti byte si desiderano, l'UDP è orientato ai messaggi, quindi devo leggere l'intero messaggio

LIVELLO DI RETE

mercoledì 10 novembre 2021 12:19

Livello di rete



Comunicazione a livello rete

1. L'entità a livello di rete riceve i segmenti dal livello di trasporto nell'host mittente, e incapsula i segmenti in datagrammi

2. I datagrammi sono inoltrati al prossimo nodo (host o router)

3. Il router esamina i campi intestazione in tutti i datagrammi IP che lo attraversano e li inoltra da un collegamento in ingresso ad un collegamento in uscita

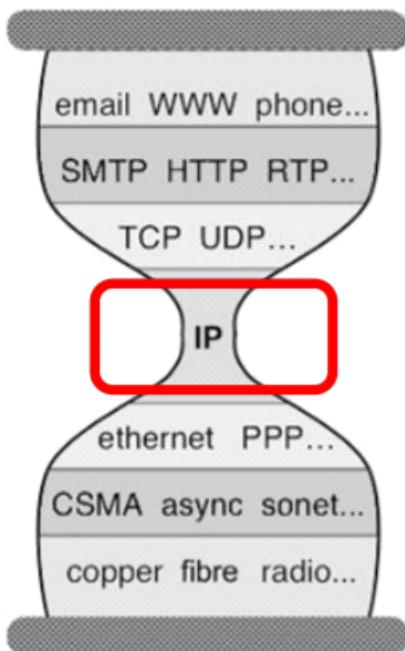
4. Sul lato destinatario, consegna i segmenti al livello di trasporto (demux TCP o UDP)

Il livello di rete è implementato nei livelli intermedi, mentre quello di trasporto è implementato nei punti terminali, il livello di rete implementa poche funzioni

Il livello di rete interconnette reti eterogenee

Al livello sottostante possiamo avere tecnologie di tipo diverso

Il modello a clessidra che rappresenta il piano dati



Abbiamo *tecnologie eterogenee sia ai livelli superiori che quelli inferiori*

Abbiamo poi un livello dirette relativamente semplice che sfrutta un solo protocollo(?)

In generale a livello di rete possiamo avere:

- Controllo errori
- Controllo flusso

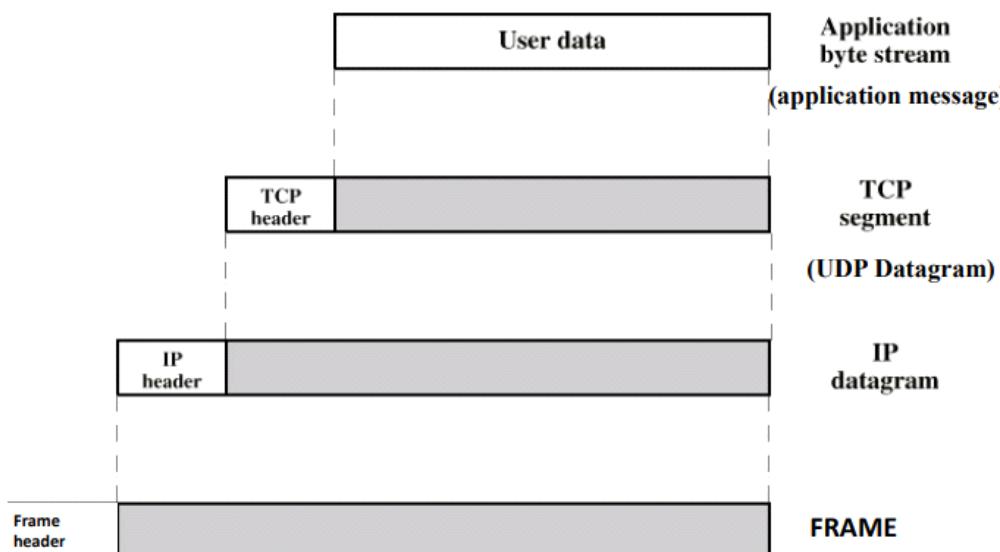
- Controllo congestione
- Qualità del servizio (rispetto al modello originale sono stati aggiunti servizi per implementare politiche di qualità del servizio)
- Sicurezza

IP RFC 791

Specifica successiva alle prime specifiche di protocolli connectionless

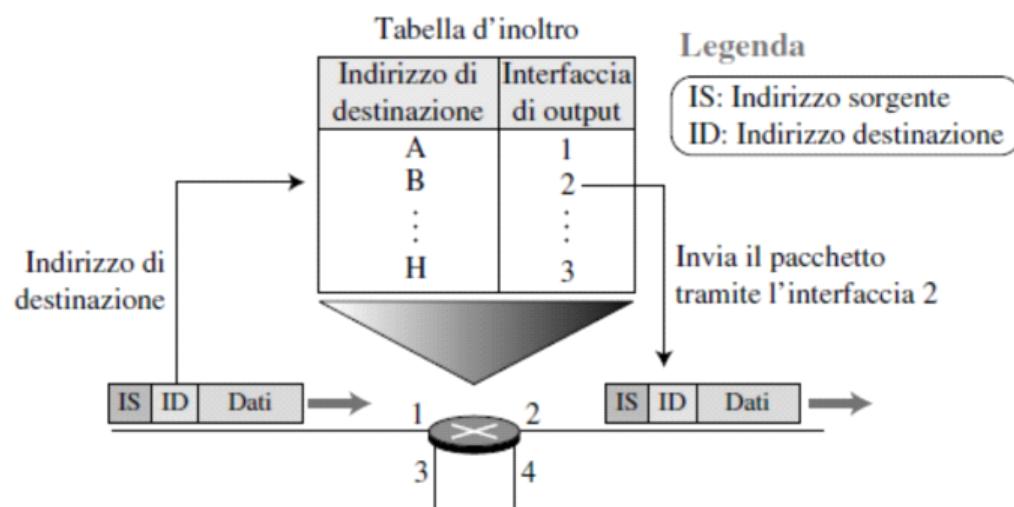
Il protocollo IP nasce come servizio best effort, non è affidabile, non prevede meccanismi di recupero degli errori, ci sono controlli di errore (send and pray)

Non prevede garanzie sulla QoS tempo di consegna e controllo di flusso, sono stati aggiunti dopo i meccanismi del genere



Funzionalità IP

Forwarding: trasferimento del pacchetto sull'appropriato collegamento di uscita



Instradamento: processo decisionale di scelta del percorso verso una destinazione

Siamo nel piano dati, abbiamo i dati utili (datagrammi) che arrivano al router, il router va a vedere l'intestazione, e in base all'info contenuta decide dove inoltrare il datagramma

Come decide?

Decide in base a una tabella di inoltro, in base all'indirizzo di destinazione si va a decidere l'interfaccia di inoltro (tutto fatto nel minor tempo possibile)

Come si costruisce la tabella di inoltro?

Funzione di instradamento -> piano di controllo

Instradamento: processo decisionale di scelta del percorso verso una destinazione, i router implementano degli algoritmi di instradamento (*routing*), con informazioni sulla rete circostante, costruzione di un suo modello della tipologia di rete e in base a quello costruire la tabella

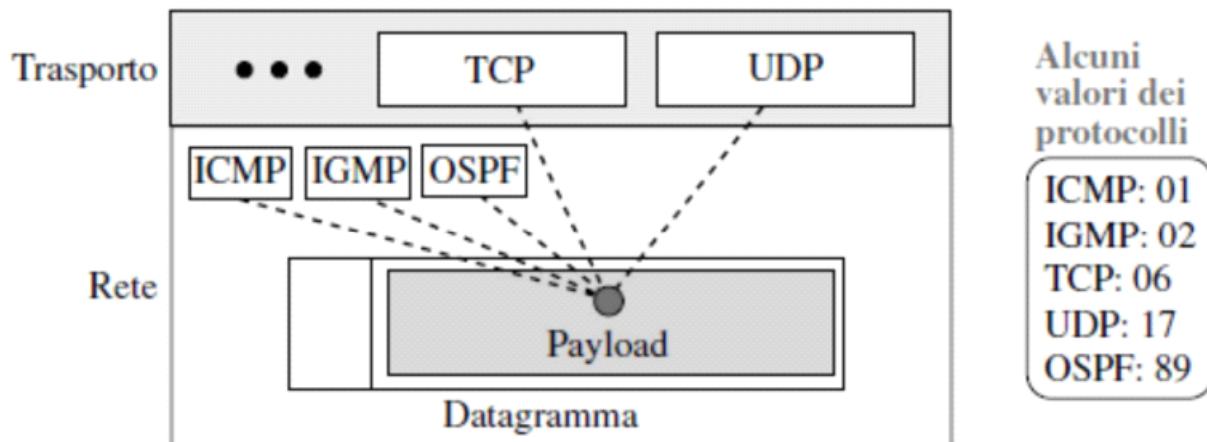
Meccanismi IP

Schemi di **indirizzamento**: strumento per identificare gli host nell'internet

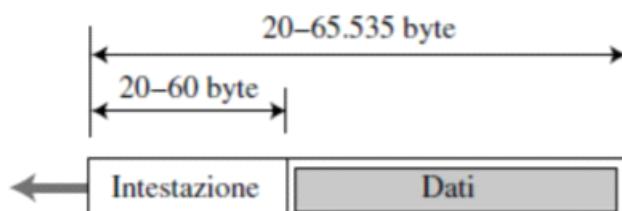
Modello datagram: (senza connessione) per la consegna dei dati

- Segmentazione e riunificazione dei pacchetti
- Controllo degli errori (solo header)
- Verifica TTL

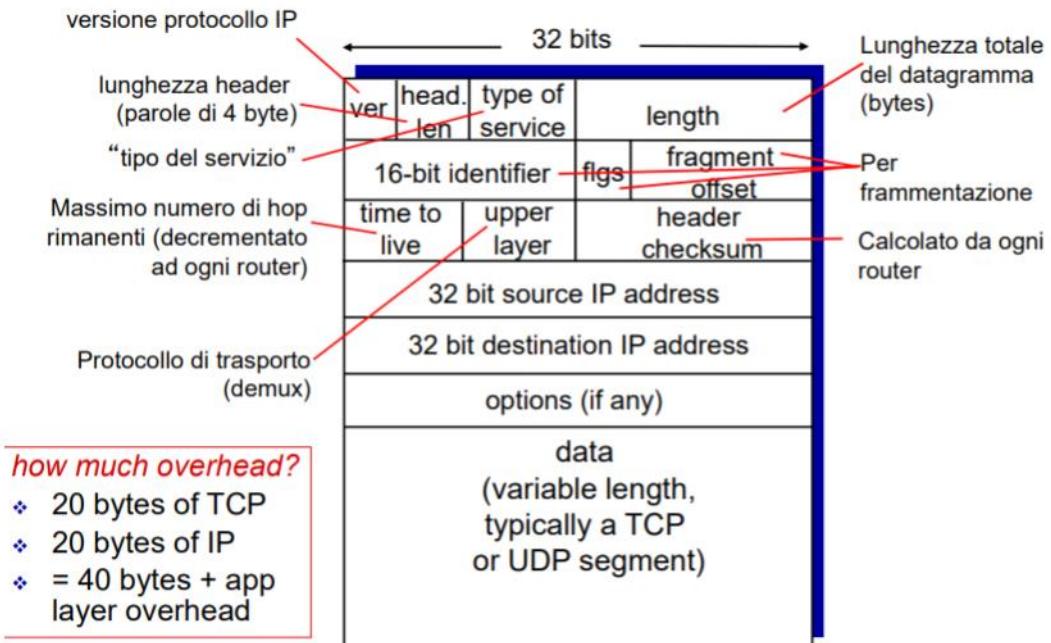
Anche a livello di rete si fa *multiplexing e demultiplexing*



Formato datagramma IP

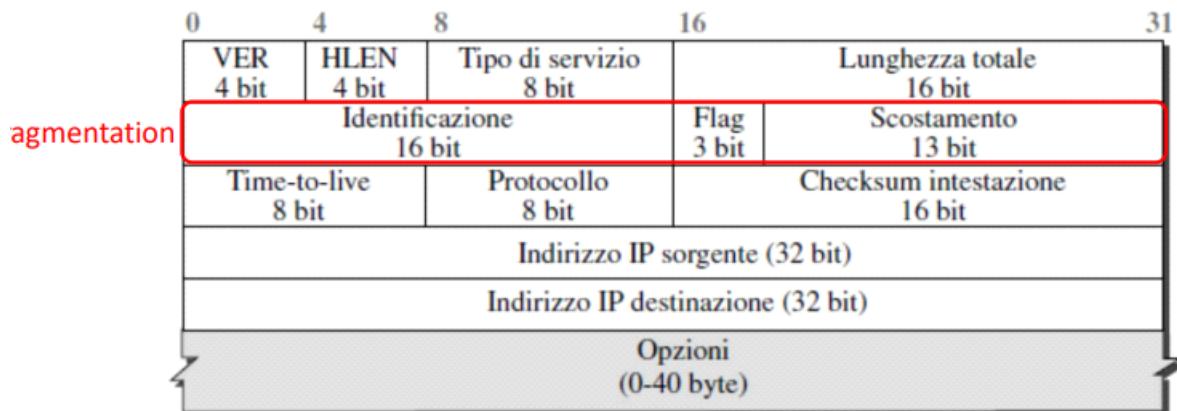


IP datagram format



- **Versione: versione di IP usata 4b**
- **Hlen: lunghezza dell'header espresso in parole da 32 bit, tipicamente 0101 20 byte (4b)**
- **Tipo di servizio: 8b**
 - 6 bit per *Differentiated services*
 - o I pacchetti vengono marcati in base alla classe di servizio (teleonata, controllo, multimedia streaming, dati a bassa priorità)
 - o Politiche di accodamento router
 - 2 bit *Explicit Congestion Notification (ECN)*: supporto a livello di rete e trasporto per la notifica di eventi di congestione, analizzare gli eventi e sapere se ci sono eventi di congestione, lo fa solo il livello di trasporto, con questi 2 bit i router collaborano nella notifica di congestione
- **Lunghezza del datagramma: 16 b lunghezza di tutto il datagramma in byte incluso header (max 65535 byte in pratica 1500)**
- **Identificatore, flag, offset: campi per la frammentazione**
- **TTL: 8bit viene decrementato a ogni passaggio da un router, quando arriva a 0 viene scartato, evita che i pacchetti vadano in giro all'infinito in percorsi ciclici, il valore iniziale dipende dal S.O.**
- **Protocollo: 8bit indica quale protocollo dello stato superiore deve ricevere i dati**
- **Checksum dell'intestazione: viene calcolato il checksum della sola intestazione ad ogni router se si ottiene un errore si scarta il datagram**
- **Indirizzi sorgente e destinazione: 32bit**
- **Opzioni: multiplo di 32 bit variabile (vengono usate tipicamente per test, monitoraggio)**
- **Dati**

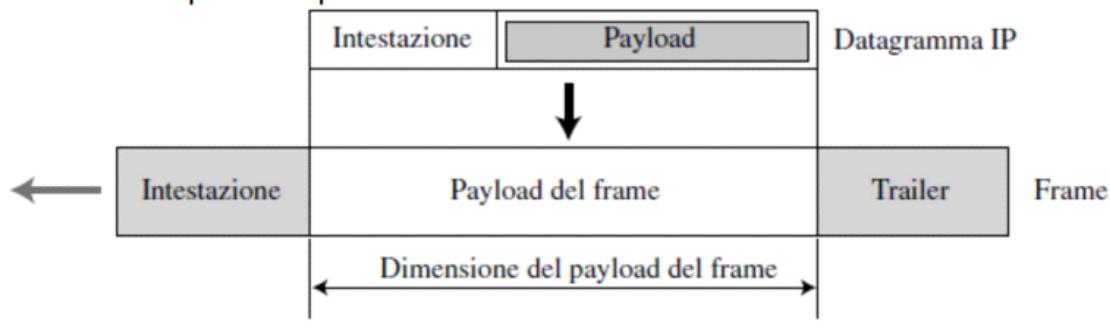
Frammentazione



MTU (Maximum Transfer Unit):

Quantità massima di dati trasportata dal protocollo di collegamento (in un frame). Può variare da una tecnologia di collegamento all'altra.

Pone un limite alla lunghezza dei datagrammi IP e tratte diverse possono porre limiti differenti



24

Meccanismo di frammentazione:

Se il router riceve un datagramma la cui dimensione supera l'MTU della rete verso cui deve inoltrare quel datagramma il router frammenta il datagramma IP in due o più datagrammi più piccoli, detti **frammenti**

Il riassemblaggio avviene alla destinazione

Se qualche frammento non arriva a destinazione si scarta l'intero datagramma

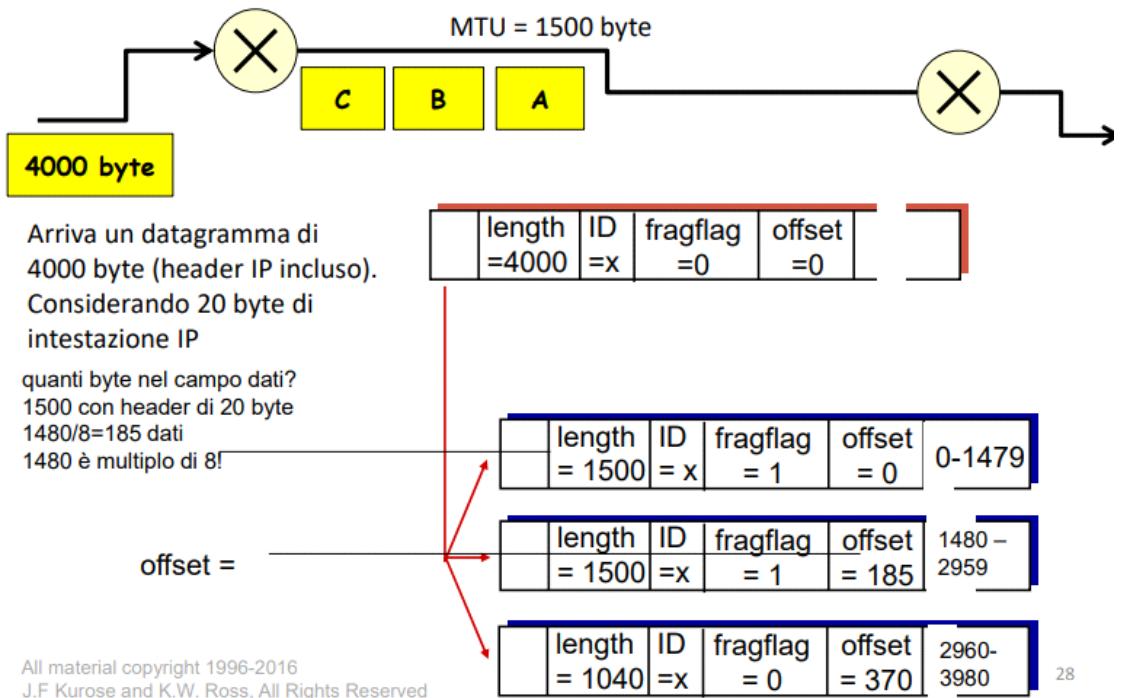
Ogni frammento è a sua volta un datagramma IP completo, il suo percorso può essere diverso da quello degli altri frammenti del datagramma

I frammenti possono arrivare fuori ordine, come si ricompone l'ordine?

Si usano identificatore, flag e offset

- **Identificatore** (16 bit) associato al datagramma dell'host sorgente. Associato a IP sorgente e IP destinazione identifica quel datagramma per un intervallo di tempo ragionevolmente lungo
 - I frammenti di quel datagramma mantengono il valore di questo campo
- Per metterli in ordine ho l'**offset** posizione relativa del frammento come multiplo di 8 byte
- **Flag:** identificare qual è l'ultimo frammento
 - 0: riservato
 - 1: *do not fragment*
 - 0: pacchetto può essere frammentato
 - 1: non deve essere frammentato
 - 2: *more fragments*
 - 0: il pacchetto è l'ultimo

- 1: non è l'ultimo



Il processo di frammentazione può essere ripetuto (se devo mandare un frammento su un collegamento con MTU più piccolo)

La ricostruzione del datagramma è un processo critico:

- Richiede risorse ai nodi intermedi e destinazione
- Ritardo di elaborazione
- Se un frammento manca deve essere scartato l'intero datagramma

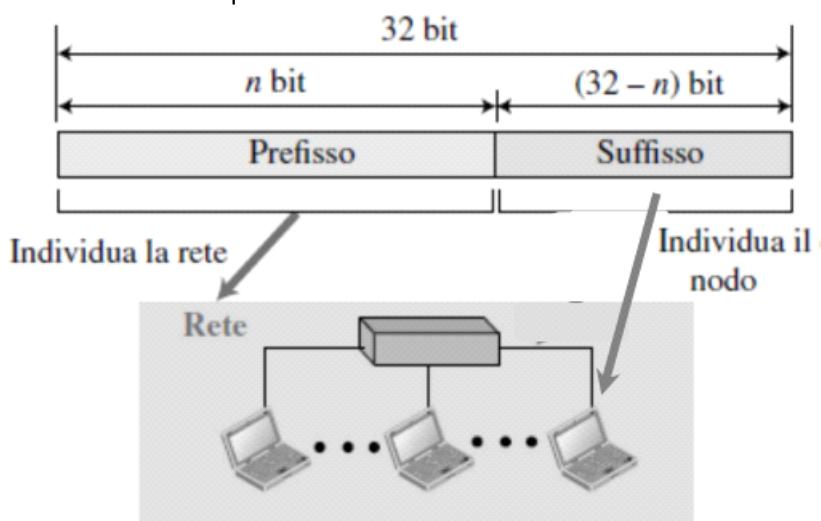
Indirizzamento IP

Ogni host è connesso a Internet tramite interfaccia di rete che è il confine fra host ed il collegamento su cui vengono inviati i datagrammi.

Ad ogni interfaccia è assegnato un **indirizzo ip**

Un indirizzo ip è costituito a 32 bit (4 byte) rappresentati in notazione decimale puntate

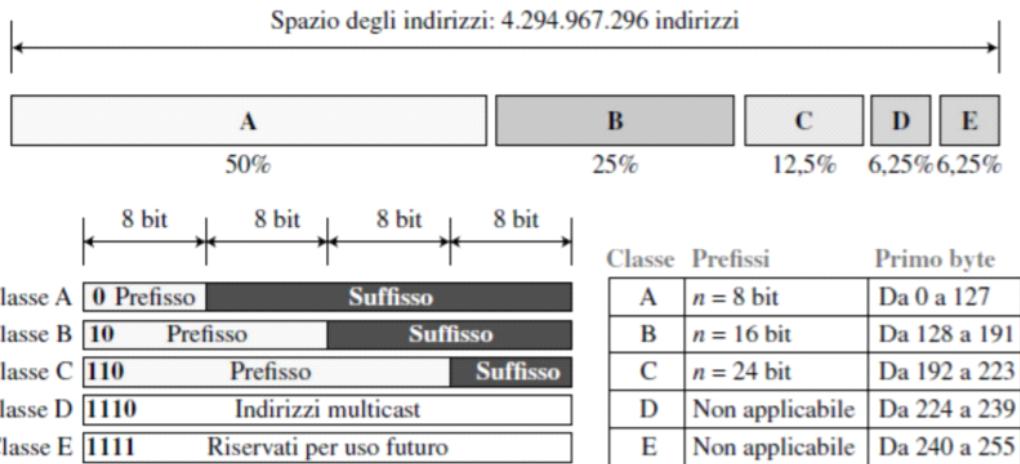
Ogni host ha un indirizzo univoco diviso in 2 parti: **network ID + Host ID** identificano una rete IP su Internet e l'host su quella rete IP



Classful addressing:

Prefissi di rete di lunghezza fissa

Reti divise in classi

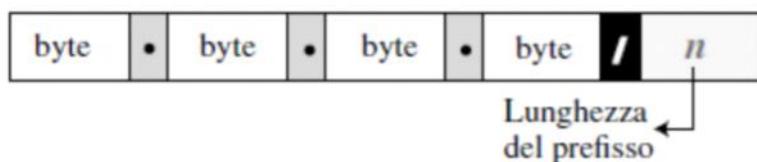


- 5 classi di indirizzi IP
 - **Classe A**: 7 bit per 128 reti IP, 24 bit
 - 0.0.0.0 - 127.255.255.255
 - Reti di 16 milioni di host
 - **Classe B**: 14 bit per ca. 16000 reti IP, 16 bit host id
 - 128.0.0.0 - 191.255.255.255
 - Reti di circa 64000 host
 - **Classe C**: 21 bit per reti IP e 8 bit per host id
 - 192.0.0.0 - 223.255.255.255
 - Reti di circa 256 host
 - **Classe D**, riservata a multicasting,
 - 224.0.0.0 - 239.255.255.255
 - **Classe E**, riservata per usi futuri
 - 240.0.0.0 – 255.255.255

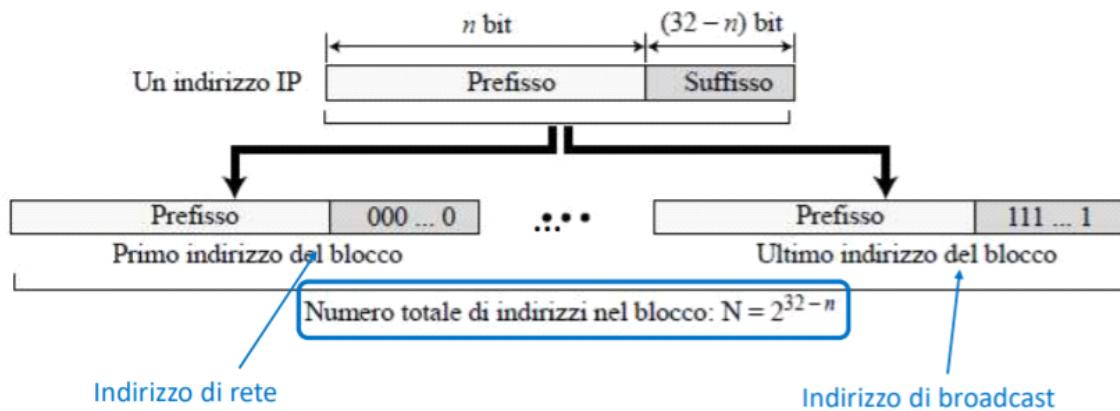
Dato un indirizzo IP è facile risalire all'indirizzo di rete, ma poco flessibile nell'utilizzo dello scarso range di indirizzi disponibili

Classless addressing

Notazione CIDR (Classless Intrdomain Routing):



Esempi:
 12.24.76.8/8
 23.14.67.92/12
 220.8.24.255/25

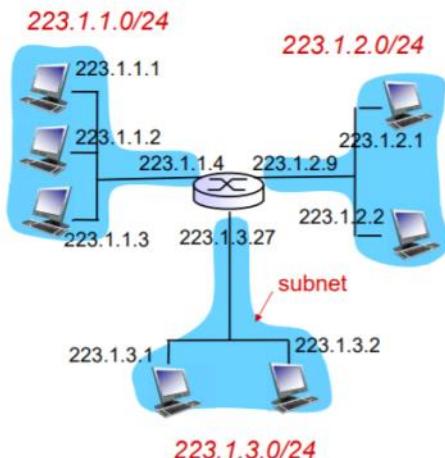


Indirizzo IP composto da un prefisso da n bit e un suffisso da $32 - n$ bit

Numero massimo di host:

Una rete con un prefisso di rete di n bit può avere al massimo $2^{(32 - n)} - 2$ host (tolgo indirizzo di rete e di broadcast)

Il prefisso indica l'indirizzo di rete



Maschera di sottorete (subnet mask)

Indirizzo: $a.b.c.d/n$

N bit più a sinistra costituiscono la parte di indirizzo di rete

Subnet mask: numero composto da 32 bit in cui i primi n bit a sx sono impostati a 1 e i rimanenti a 0, serve a distinguere quale parte di un indirizzo ip identifica la rete e quale l'host

: l'indirizzo IP **150.217.8.42** ha netmask **255.255.255.0**

Indirizzo IP	150.217.8.42	10010110 11011001 00001000 00101010
Subnet Mask	255.255.255.0	11111111 11111111 11111111 00000000
AND	150.217.8.0	10010110 11011001 00001000 00000000

La rete ha indirizzo
150.217.8.0/24

Assegnazione di blocchi di indirizzi:

Gli indirizzi Ip sono gestiti da ICANN e assegnati in blocchi agli ISP, come assegnano gli ISP gli indirizzi ai clienti? Gli ISP assegnano ai clienti sottoblocchi di indirizzi, divisi in blocchi di indirizzi contigui

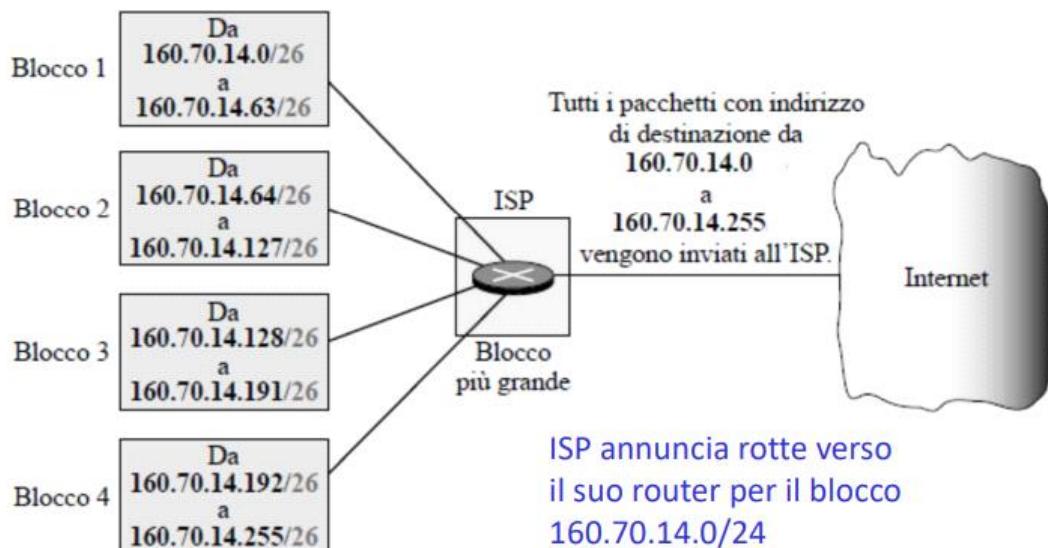
- ISP che deve suddividere un blocco in 8 blocchi di indirizzi (un blocco di 512 indirizzi per ciascuna organizzazione)

Blocco dell'ISP	11001000 00010111 00010000 00000000	200.23.16.0/20
Organizzazione 0	11001000 00010111 00010000 00000000	200.23.16.0/23
Organizzazione 1	11001000 00010111 00010010 00000000	200.23.18.0/23
Organizzazione 2	11001000 00010111 00010100 00000000	200.23.20.0/23
...
Organizzazione 7	11001000 00010111 00011110 00000000	200.23.30.0/23

- Il numero di indirizzi N in ogni subnetwork deve essere una potenza di 2
- La lunghezza del prefisso di ogni sottorete (n) va calcolata con la formula
 - $n = 32 - \log_2 N$ dove N è il numero di indirizzi della sottorete
 - Nel nostro caso: 23 -> sottoblocco identificato con 3 bit aggiuntivi
- Si assegnano blocchi di indirizzi **contigui** (nell'esempio 000, 001, 010..)
- NB se i blocchi sono di dimensioni diverse si parte dai blocchi più grandi

Aggregazione di indirizzi

Il CIDR permette anche di aggregare i blocchi di indirizzi per rendere più efficiente l'instradamento



Indirizzi speciali:

- This-host 0.0.0.0:** usato quando un host ha necessità di inviare un datagramma ma non conosce il proprio indirizzo IP (quando deve mandare il pacchetto per farselo assegnare)
- Limited-broadcast 255.255.255.255:** usato quando un router o un host devono inviare un datagramma a tutti i dispositivi che si trovano all'interno della rete. I router bloccano la propagazione alla sola rete locale
- Loopback 127.0.0.1:** il datagramma con questo indirizzo di destinazione non lascia l'host locale
- Indirizzi privati:** quattro blocchi riservati per indirizzi privati (riservati per reti locali)
10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, 196.245.0.0/16
- Indirizzi multicast:** blocco 224.0.0.0/4

Assegnazione di un indirizzo IP

- Configurazione manuale:** l'amministratore configura direttamente nell'host indirizzo IP, gateway/route, netmask, e indirizzo IP di almeno un server DNS
- DHCP:** L'host ottiene il proprio indirizzo e le altre informazioni in modo automatico

DHCP

Quando un host si aggiunge alla rete ottiene dinamicamente un indirizzo IP da un programma server in rete

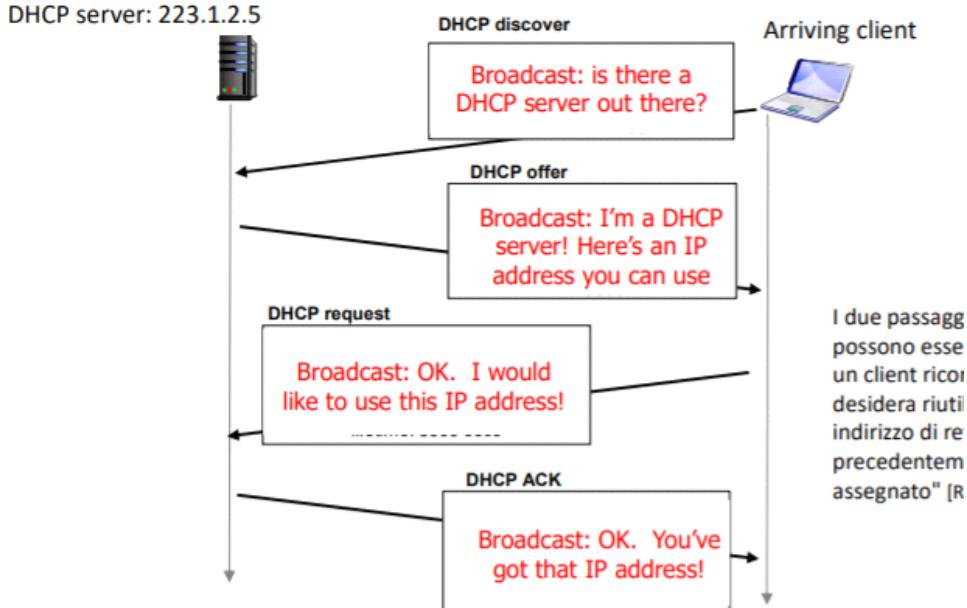
Cosa offre il servizio:

- Rinnovo indirizzo in uso
- Riuso di un indirizzo precedentemente assegnato
- Supporto per utenti in mobilità che si uniscono/lasciano la rete

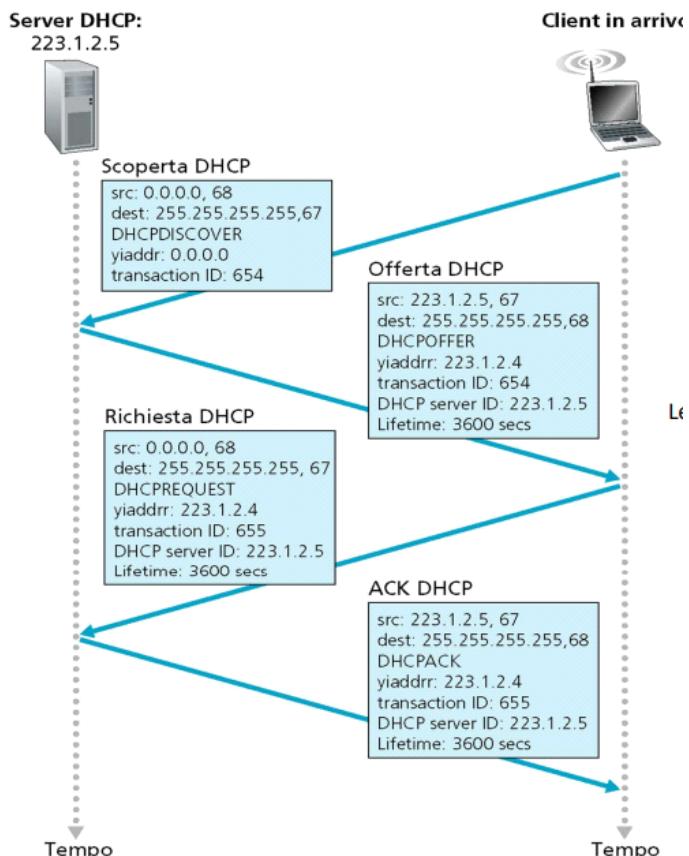
È un protocollo client-server

- L'host invia in broadcast un messaggio **DHCP discover** (opzionale)
- Il server DHCP risponde con un messaggio **DHCP offer** (opzionale)
- L'host richiede un indirizzo IP: messaggio **DHCP request**
- Il server DHCP risponde con un messaggio **DHCP ack** (se la richiesta va a buon fine)

Il server DHCP può essere ospitato sul router, servendo tutte le sottoreti a cui è collegato



Il client può anche utilizzare un indirizzo che gli è stato assegnato in passato se desidera



DHCP può restituire ulteriori informazioni:

- Indirizzo del gateway/router
- Netmask
- Nome e indirizzo IP di almeno un server DNS

Usa protocollo UDP

IP FORWARDING

Ogni datagramma IP è soggetto a "forwarding" da parte dell'host di origine e del router che sta attraversando

Inoltro

Inoltro di un pacchetto verso l'uscita verso un altro host o router che si prende in carico del datagramma o alla destinazione finale se siamo nella sottorete dell'host finale

Inoltro diretto

Il pacchetto IP ha come destinazione un host nella propria sottorete

Si manda direttamente il datagramma al destinatario e l'indirizzo a livello link è quello del destinatario(MAC address)

Inoltro indiretto

Il pacchetto ha come destinazione un host di un'altra rete, si delega l'invio a qualcun altro
L'indirizzo di dest IP è quello originale, l'indirizzo a livello di collegamento è quello del router (a cui viene delegato l'invio)

Si osserva come, in entrambi i casi, condizioni necessarie perché tutto funzioni sono che:

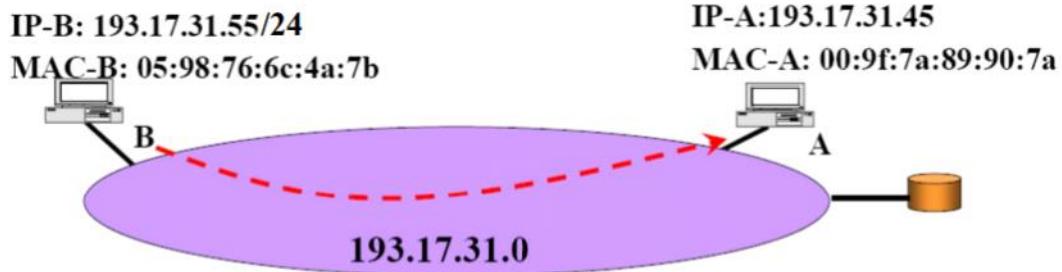
» esista un cammino (funzionante e) diretto, a livello data-link, tra tutti gli host che appartengono ad una stessa sottorete;

» ogni host coinvolto abbia un indirizzo IP "giusto", cioè con uguale net ID (cioè appartenga alla stessa sottorete) e con host ID univoco nella sottorete.

Le due condizioni insieme diventano condizione necessaria e sufficiente perché la comunicazione "funzioni".

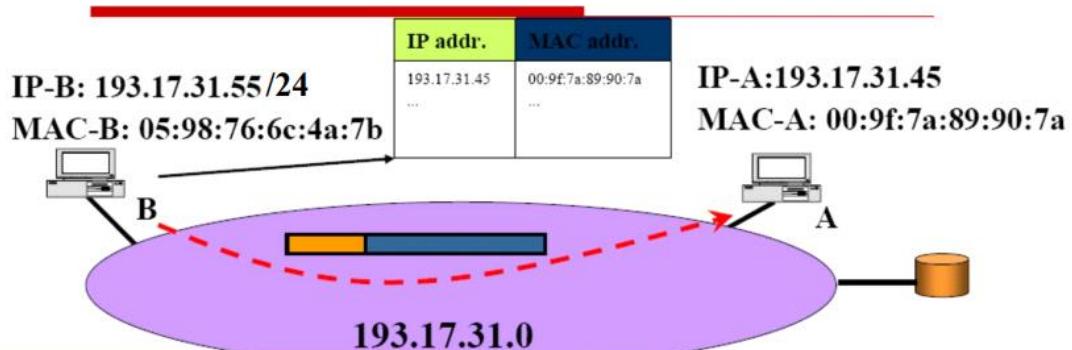
Ad ogni interfaccia verso la rete IP viene assegnato un indirizzo IP distinto

Il router svolge funzioni di instradamento a livello IP. Legge gli indirizzi IP, consulta la propria tabella di forwarding e decide dove mandare il pacchetto IP.



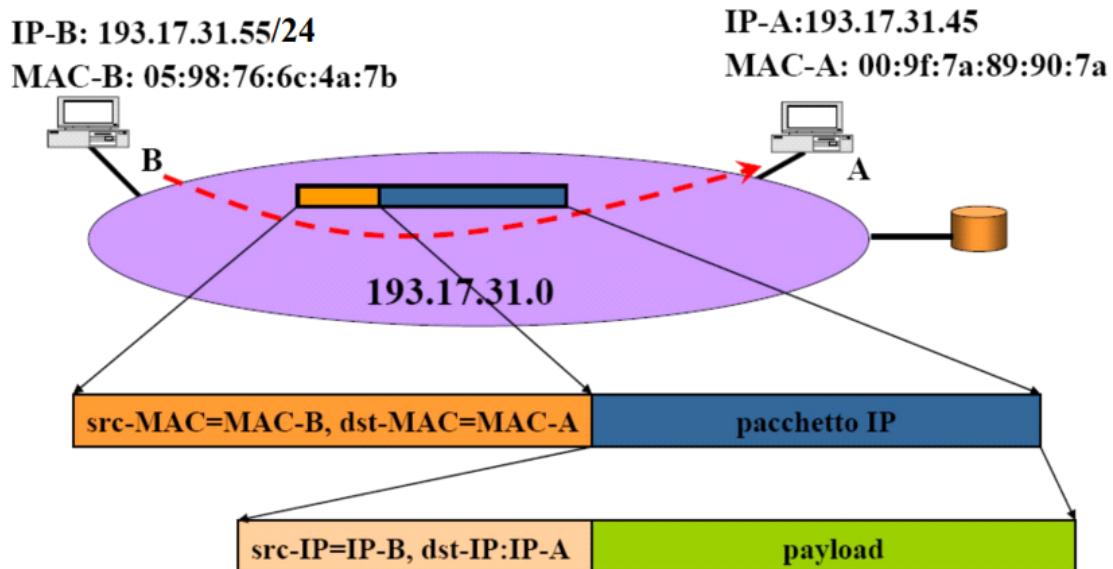
1. L'entità IP di B deve spedire un pacchetto all'indirizzo IP-A

2. B conosce l'indirizzo IP-B della propria interfaccia e dal confronto con IP-A capisce che A si trova nella stessa rete



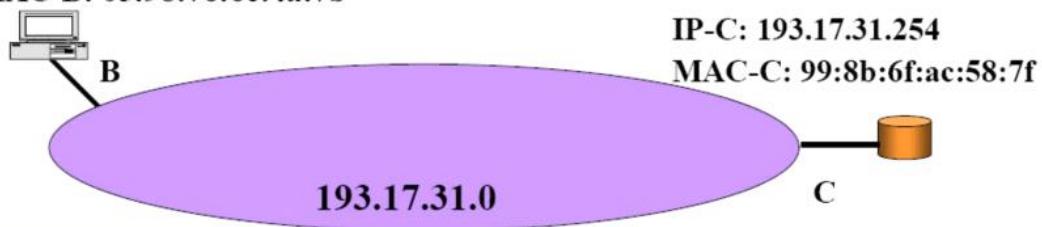
3. B consulta una tabella di corrispondenza tra indirizzi IP e indirizzi della rete (indirizzi MAC nel caso di rete locale) per reperire l'indirizzo MAC-A

4. L'entità IP di B passa il pacchetto al livello inferiore che crea un pacchetto con destinazione MAC-A



Inoltro indiretto

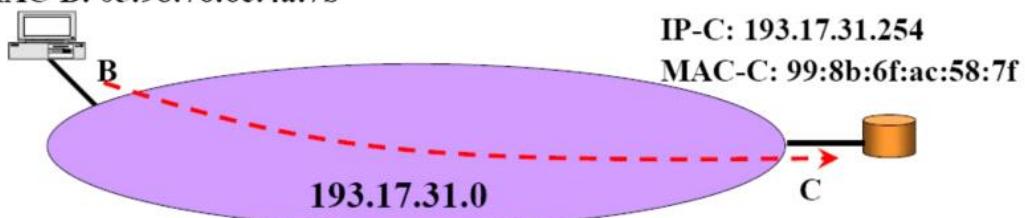
IP-B: 193.17.31.55/24
MAC-B: 05:98:76:6c:4a:7b



1. L'entità IP di B deve spedire un pacchetto all'indirizzo IP-D=131.17.23.4

2. B conosce l'indirizzo IP-B della propria interfaccia e dal confronto con IP-D capisce che D NON si trova nella stessa rete

IP-B: 193.17.31.55/24
MAC-B: 05:98:76:6c:4a:7b

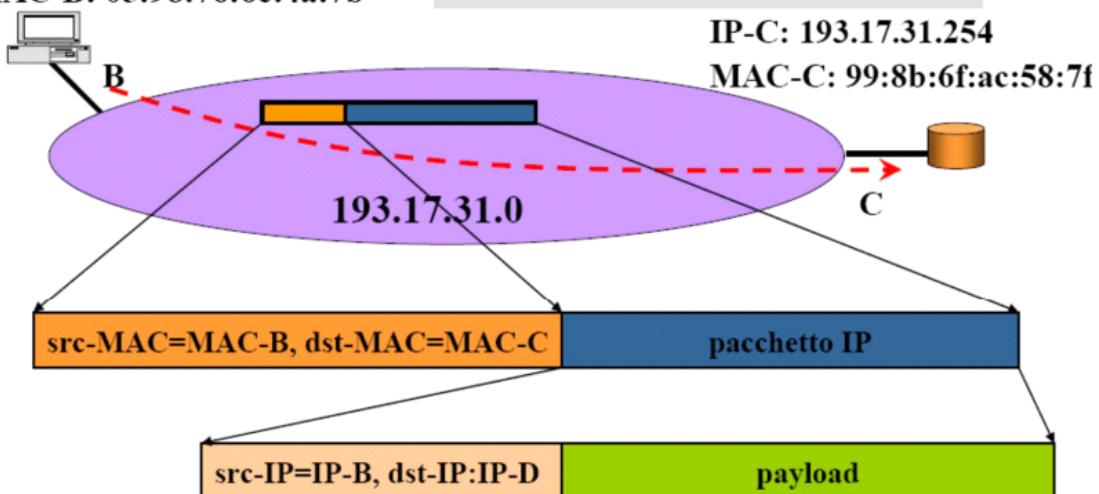


3. B deve dunque inoltrare il pacchetto ad un router (di solito è configurato un solo default router)

4. B recupera l'indirizzo MAC del router nella tabella di corrispondenza e passa il pacchetto al livello inferiore

IP-B: 193.17.31.55/24
MAC-B: 05:98:76:6c:4a:7b

**5. il pacchetto viene
costruito e spedito
sull'interfaccia**



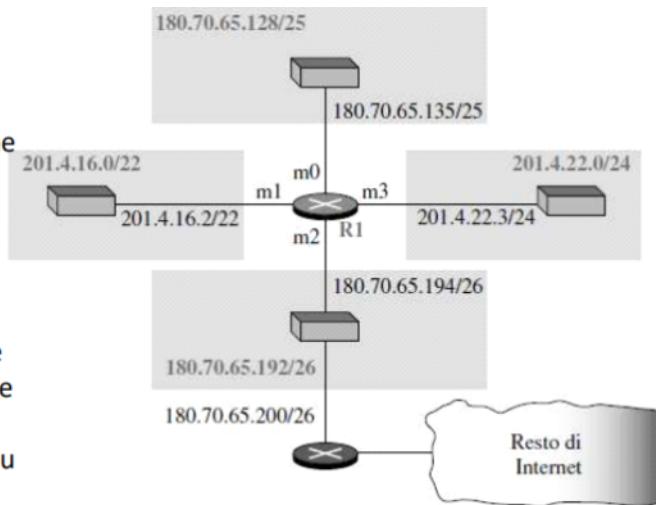
Inoltro - router

Esempio

R1 riceve datagram con IP destinazione 180.70.65.140:

10110100.01000110.01000001.10 001100

1. Applica la prima maschera a 180.70.65.140 ottenendo 180.70.65.128, che non combacia con l'indirizzo di rete corrispondente
2. Applica la seconda 180.70.65.128 che combacia con l'indirizzo di rete corrispondente. Inoltra il datagram su interfaccia m0.



Indirizzo di rete/maschera	Salto successivo	Interfaccia
180.70.65.192/26	–	m2
180.70.65.128/25	–	m0
201.4.22.0/24	–	m3
201.4.16.0/22	–	m1
Default	180.70.65.200	m2

85

Aggregazione degli indirizzi

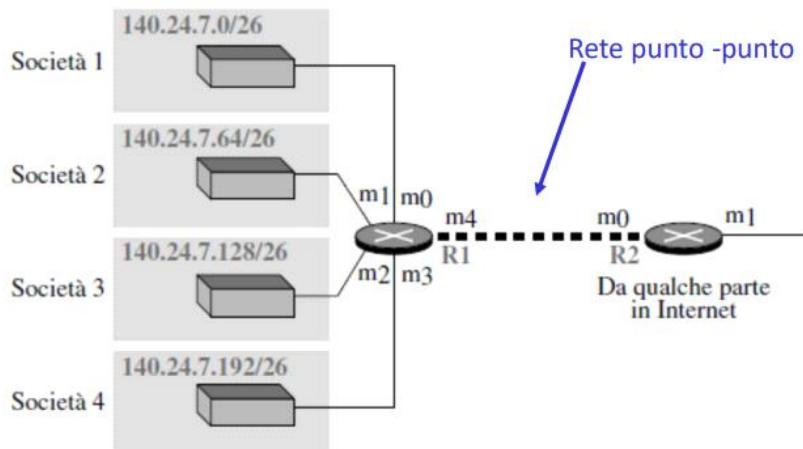


Tabella d'inoltro per R1

Indirizzo di rete/maschera	Indirizzo del salto successivo	Interfaccia
140.24.7.0/26	-----	m0
140.24.7.64/26	-----	m1
140.24.7.128/26	-----	m2
140.24.7.192/26	-----	m3
0.0.0.0/0	Indirizzo di R2	m4

Tabella d'inoltro per R2

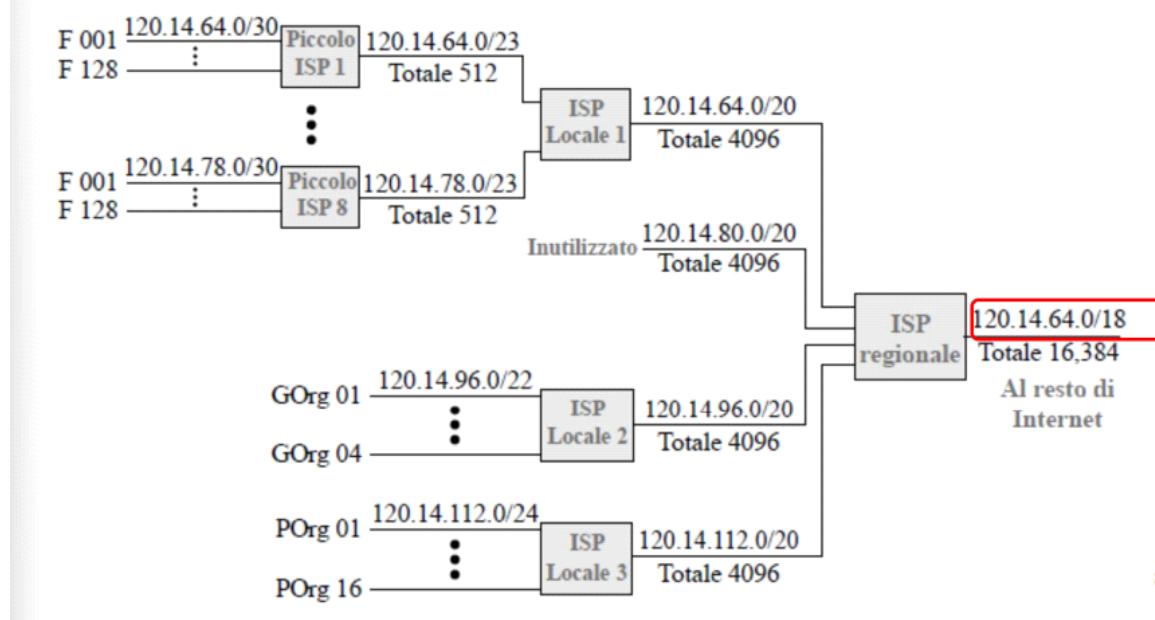
Indirizzo di rete/maschera	Indirizzo del salto successivo	Interfaccia
140.24.7.0/24	Indirizzo di R1	m0
0.0.0.0/0	Router di default	m1

86

Routing gerarchico

ISP regionale a cui sono stati assegnati 16.384 indirizzi a partire da 120.14.64.0

ISP regionale ha deciso di dividere questo blocco in 4 sottoblocchi, ciascuno con 4096 indirizzi. Il resto di internet non è consapevole di tale divisione



88

NAT (Network Address Translation)

NAT permette di trasmettere su Internet il traffico proveniente da sistemi attestati su sottoreti private, in cui sono assegnati indirizzi IP privati

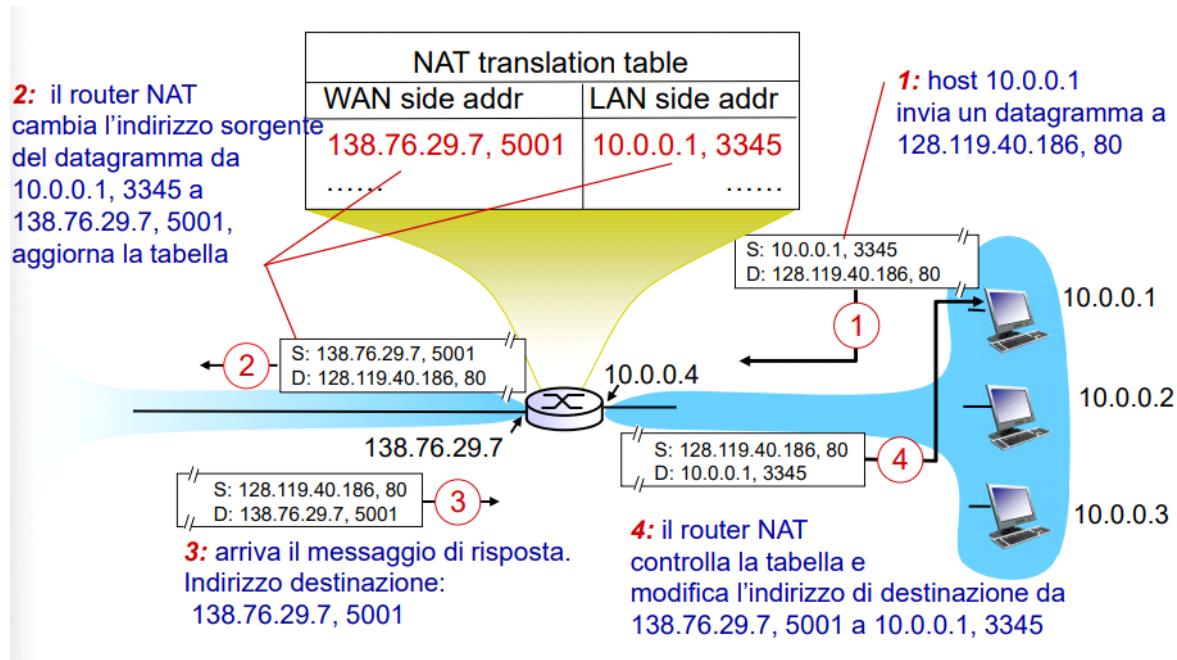
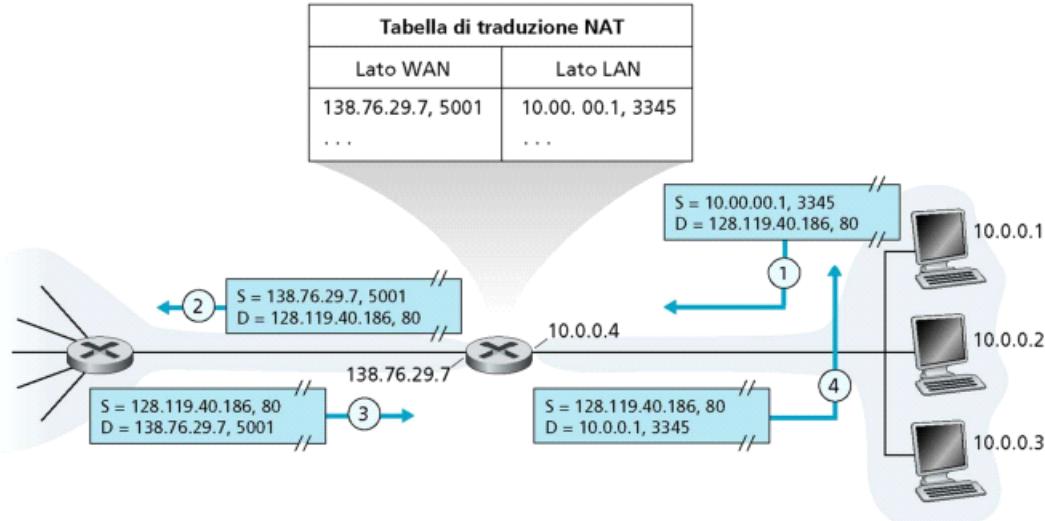
L'accesso di una rete privata su Internet è realizzato attraverso un router abilitato alla NAT, questo router ha almeno un indirizzo IP pubblico

- Il traffico in ingresso e uscita dal router ha come indirizzo IP sorgente l'indirizzo pubblico del router

Il router di accesso ha in memoria una tabella di traduzione NAT, le cui righe contengono le associazioni:

(IP privato, porta) (IP pubblico e porta assegnata dal router)

Tabella di traduzione NAT	
Lato WAN	Lato LAN



Protocollo ICMP

Entra in azione quando si è verificato un errore

Protocollo implementato per funzioni di controllo

Può essere usato sia da host che da router per errori o per sapere se un host è raggiungibile PING

I messaggi ICMP sono incapsulati all'interno di datagrammi IP, gli host che implementano un protocollo IP devono supportare anche ICMP

Casi tipici di utilizzo:

Router o host che devono informare la sorgente che qualcosa è andato storto:

- Host non raggiungibile, porta non raggiungibile, niente in ascolto...

In generale servono per informare il mittente di errori o di eventi avvenuti nell'inoltro di un pacchetto IP

I pacchetti ICMP vengono instradati dai router prima dei pacchetti IP ordinari

Ci sono regole per evitare il moltiplicarsi di questi pacchetti di controllo:

- Per pacchetti IP frammentati, i messaggi ICMP sono relativi solo al frammento con offset 0
- Non sono mai inviati in risposta a pacchetti IP con indirizzo mittente che non rappresenta in modo univoco un host (es 0.0.0.0, 127.0.0.1)
- Mai inviati in risposta a broadcast o multicast

Non mandati in risposta a messaggi di errore ICMP, ma possono essere inviati in risposta a messaggi ICMP di interrogazione

Tipi di messaggio ICMP

- Messaggi di segnalazione/errore
- Messaggi richiesta/risposta

Nell'header ICMP i campi Tipo (8 bit) e Codice (8 bit) indicano tipo e significato dei messaggi

Valori del tipo e dei codici:

Messaggi di segnalazione errori	Messaggi di richiesta
03 : destinazione non raggiungibile (codici da 0 a 15)	08 and 00 : richiesta e risposta echo (solo codice 0)
04 : source quench (solo codice 0)	13 and 14 : richiesta e risposta timestamp (solo codice 0)
05 : reindirizzamento (codici da 0 a 3)	
11 : tempo scaduto (codici 0 e 1)	
12 : problema ad un parametro (codici 0 e 1)	

Nota: si veda il sito web del volume per ulteriori spiegazioni relative ai valori dei codici.

Nell'header ICMP i campi Tipo e codice indicano il tipo e significato dei messaggi (entrambi 8 bit)

ICMP Tipo	Codice	Descrizione
0	0	risposta al messaggio di eco (a ping) - <i>echo replay</i>
3	0	rete di destinazione irraggiungibile - <i>destination network unreachable</i>
3	1	host di destinazione irraggiungibile - <i>destination host unreachable</i>
3	2	protocollo di destinazione irraggiungibile - <i>destination protocol unreachable</i>
3	3	porta di destinazione irraggiungibile - <i>destination port unreachable</i>
3	6	rete di destinazione sconosciuta - <i>destination network unknown</i>
3	7	host di destinazione sconosciuto - <i>destination host unreachable</i>
4	0	strozzamento della sorgente (controllo della congestione) - <i>source quench</i>
8	0	richiesta di eco - <i>echo request</i>
9	0	annuncio dal router - <i>router advertisement</i>
10	0	scoperta del router - <i>router discovery</i>
11	0	TTL scaduto - <i>TTL expired</i>
12	0	cattiva intestazione IP - <i>IP header bad</i>

Ping

Il ping è un programma che ci permette di mandare messaggi di richiesta e attendere risposta ICMP invia una richiesta, fa un payload lo incapsula direttamente in un datagramma IP tipo 8 e codice 0, un altro host se attivo può rispondere con una risposta eco tipo 0 e codice 0

Fornisce anche misure dell'RTT, può anche misurare l'affidabilità e la congestione del router tra due host in maniera grossolana mandando una sequenza di messaggi richiesta-risposta

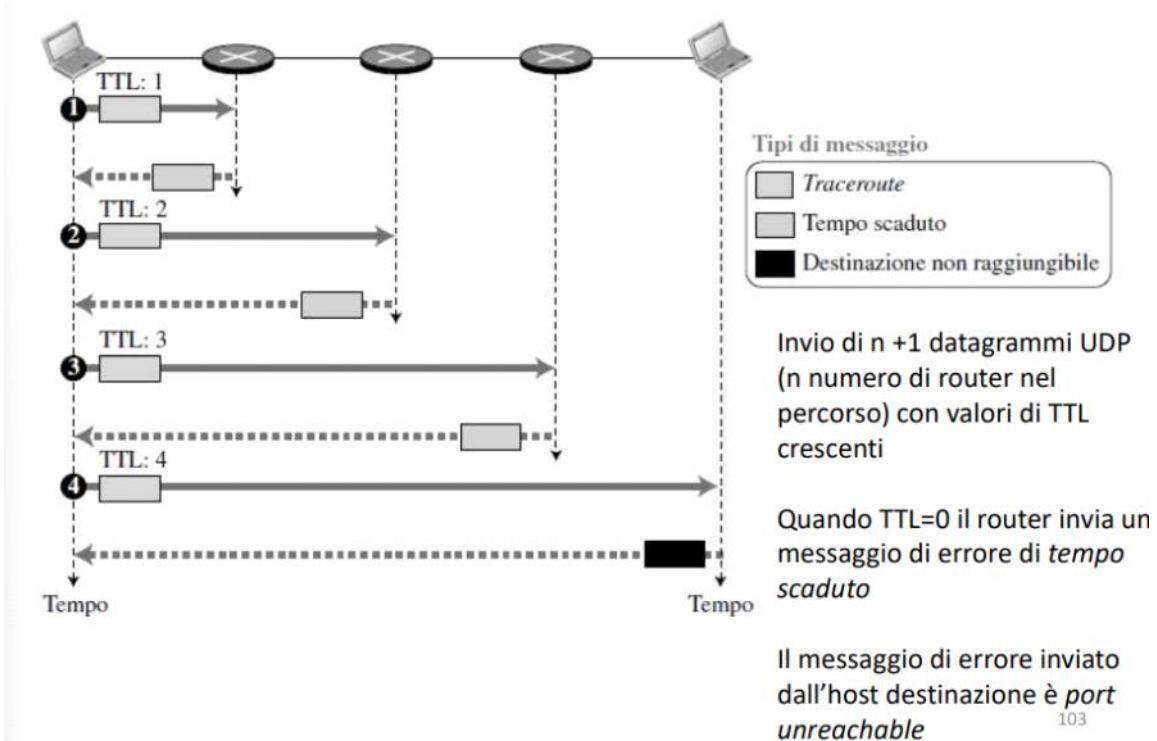
Traceroute

Traceroute individua il percorso di un datagramma dalla sorgente alla destinazione, mostra l'indirizzo IP dei router che vengono visitati lungo il percorso

Di base si imposta il TTL a un certo numero partendo da uno basso e via via incrementare in modo da costringere i router nel percorso a mandare il messaggio di errore di pacchetto esaurito

- Uso del protocollo UDP, si invia un datagramma ad una porta su cui è improbabile che un processo sia in ascolto
- I datagrammi sono configurati in modo da scatenare l'invio di messaggi di errore di tipo (TTL esaurito, porta non raggiungibile)

Tipicamente vengono mandati 3 messaggi per ogni TTL



Iposta anche un timer per trovare il tempo di round trip di ciascun router e della destinazione

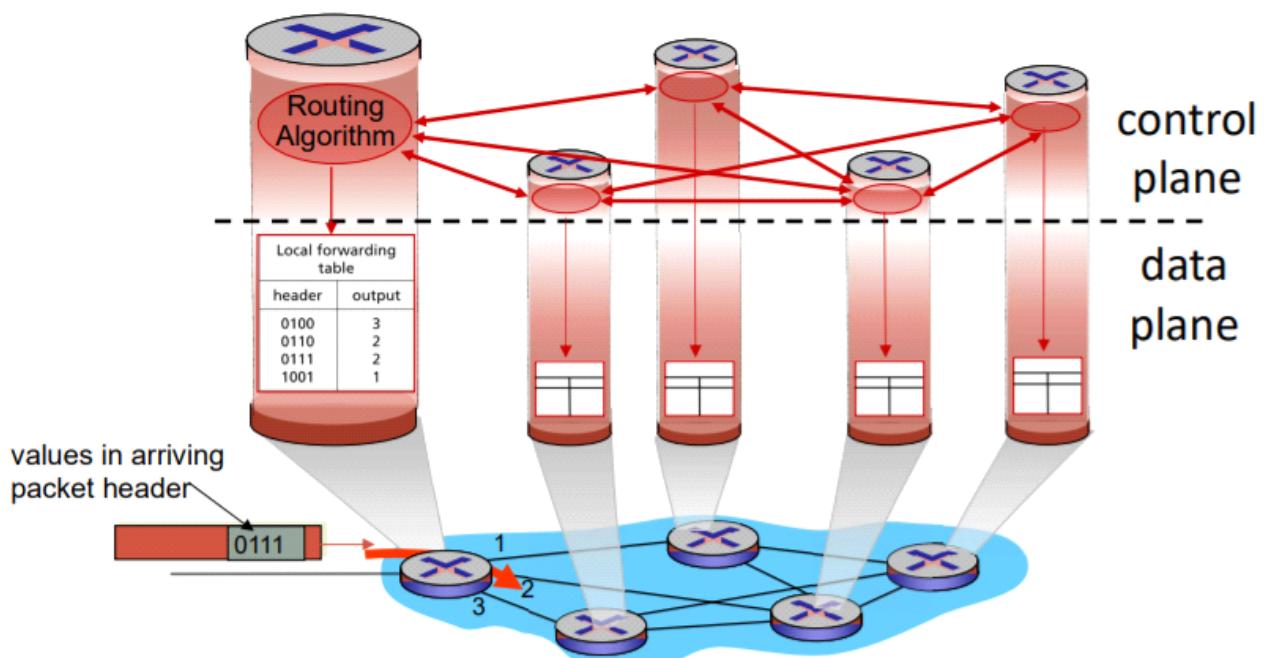
Architettura di un router

Forwarding: *data plane*

Azione del nodo a livello di rete (terminale host o router) per trasferire i pacchetti sull'appropriato collegamento in uscita, tramite la tabella di inoltro *data plane*

Routing: *control plane*

Algoritmi di routing e protocolli di routing che li implementano e le comunicazioni tra i router a determinare i valori che verranno inseriti nella tabella di inoltro



routing basato su destinazione

Il router va a prendere il valore della destinazione nel pacchetto, consulta la tabella di inoltro e decide su quale uscita inoltrare il pacchetto

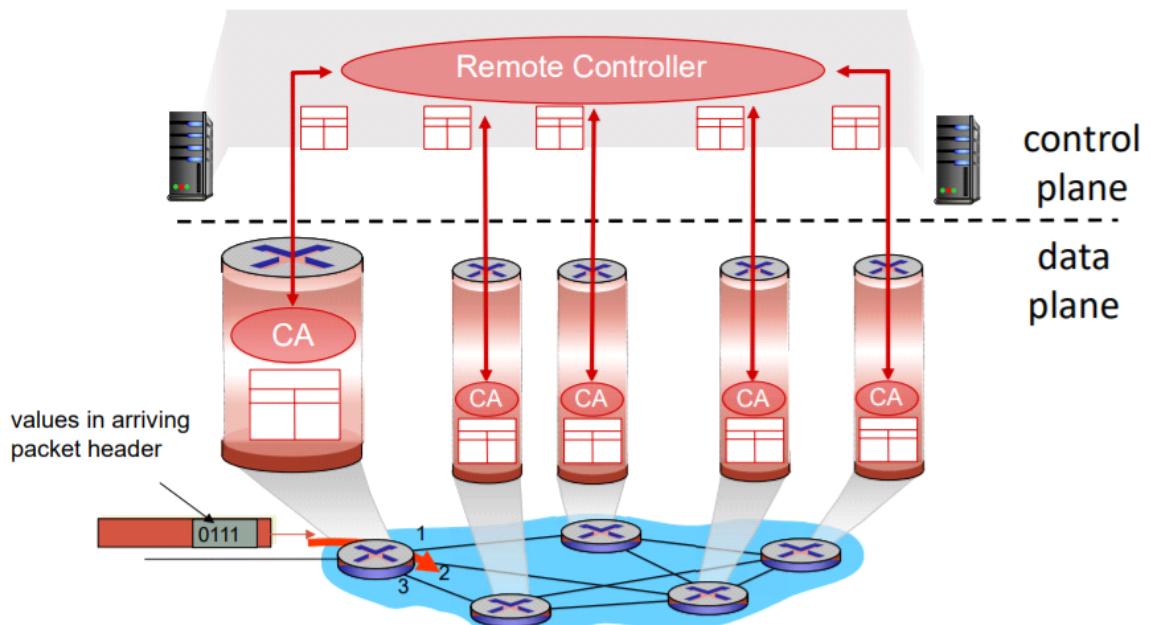
Routing decentralizzato:

Ogni router implementa algoritmi e protocolli di instradamento, costruisce la sua tabella scambiandosi messaggi con gli altri router, in base a queste informazioni costruisce la tabella di instradamento

Software defined networking

In questo approccio, mentre nell'architettura decentralizzata ogni router implementa funz del piano dati e funz del piano di controllo, in questo approccio abbiamo che a livello di router si implementa principalmente la funzione di inoltro, mentre la funzione di controllo non è implementata sui router, ma da un software (Control agent o controller remoto?) che riceve dai router delle info sulla tipologia, stato collegamenti e traffico in base a queste info si costruisce la topologia della rete info sui punti raggiungibili e i percorsi. La funzione del control plane e dei router è lasciata al software in esecuzione su un server.

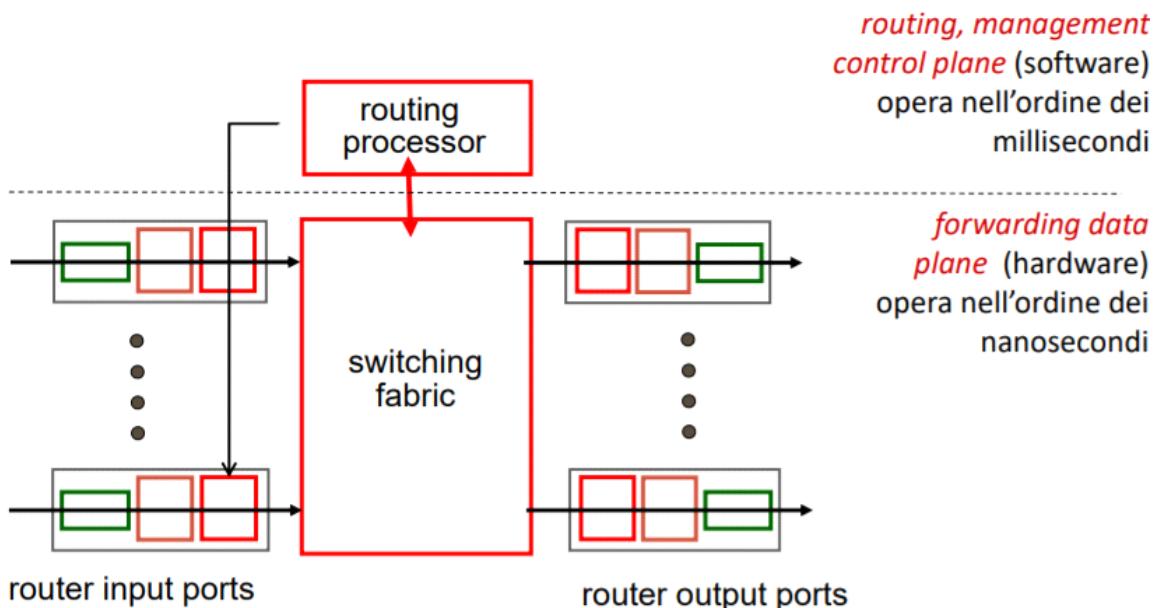
Routing logicamente centralizzato



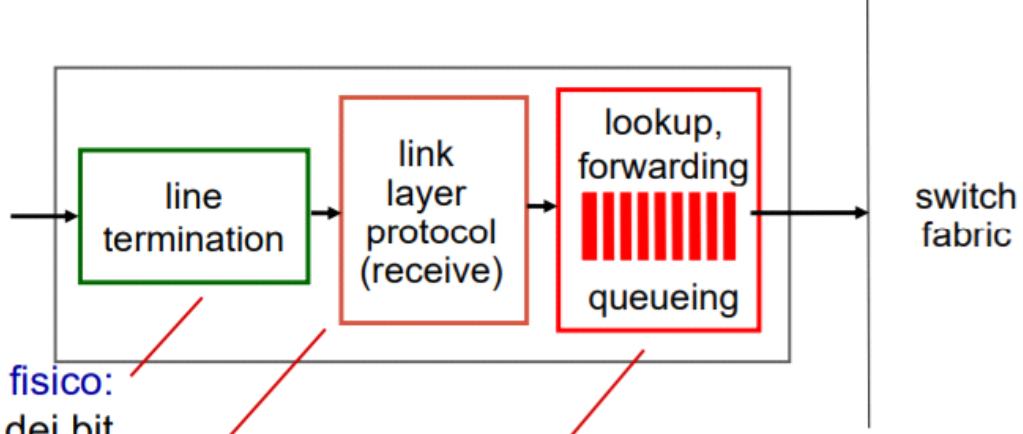
Vantaggio principale di un routing decentralizzato:
Se c'è un cambiamento appena il router lo sa aggiorna i percorsi

Architettura di un router

- Porte di input
- Porte di output
- Processore di routing
- Switching fabric (struttura di commutazione)



Porte di input:



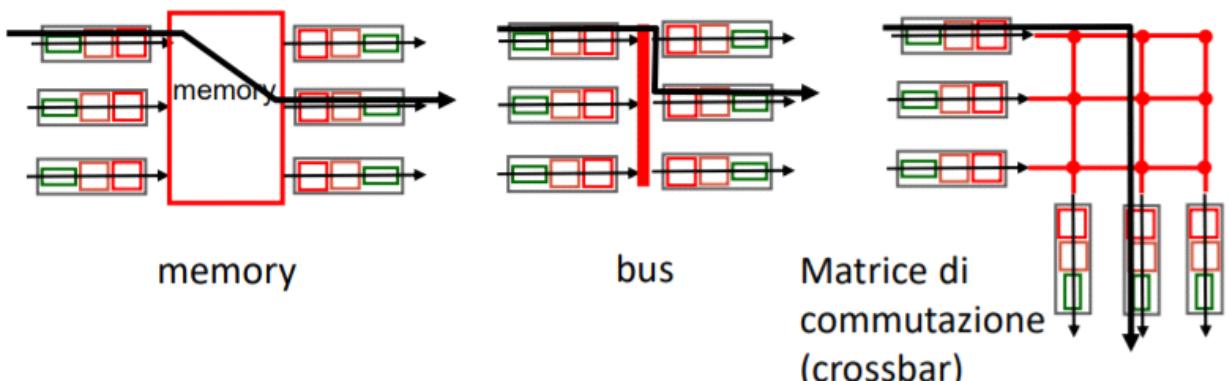
Livello fisico:
Ricezione dei bit

Porta di input: terminazione di linea riceve il segnale elettrico
 Link layer protocol: implementazione dei protocolli di collegamento
 Lookup forwarding queueing: decidere su quale porta inoltrare il messaggio
 Ogni porta tiene una copia della tabella di inoltro per determinare la porta nel modo più veloce possibile
 Elaborazione alla velocità "line rate"
 Se arrivano i pacchetti a una velocità maggiore di quella di inoltro si crea accodamento

Struttura di commutazione - switching fabric

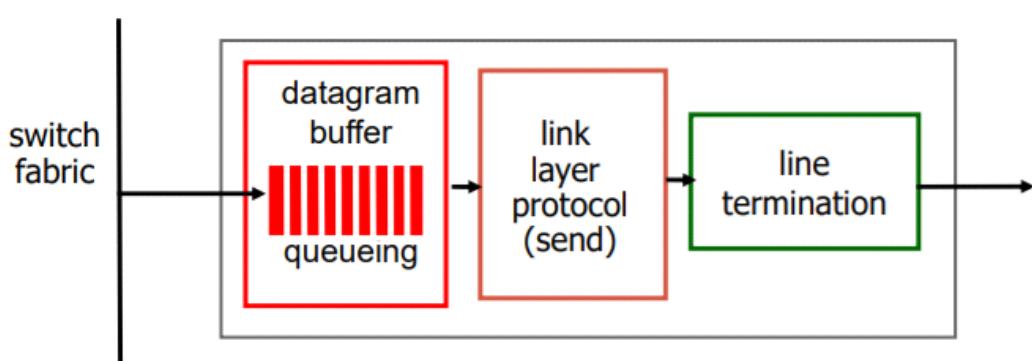
Mette in relazione la porta di input e quella di output
 Trasferisce il pacchetto dal buffer di input al buffer di output.
 Velocità di commutazione: velocità con cui i pacchetti possono essere trasferiti dagli ingressi alle uscite

- N input: velocità di commutazione auspicabile N volte la velocità della linea



Memoria: i pacchetti vengono copiati in memoria e spostati da una porta d i input a una di output
 Bus: bus condiviso per trasferire i pacchetti
 Matrice di commutazione (interconnessione): implementate con dispositivi a stato solido in cui è possibile chiudere i collegamenti per creare un percorso tra p input e p output

Porte di output:



Buffering: Coda per eventuali pacchetti in attesa se arrivano a una velocità maggiore di quella di uscita (vincolo della tecnologia usata per trasmettere)

I router possono avere più code per la gestione dei pacchetti in uscita, è possibile "accelerare" il flusso di traffico andando a rallentare gli altri (traffic shaping) dai una priorità al flusso di traffico che deve essere inviato con meno attesa e rallentati gli altri, dai una priorità a una determinata coda

Scheduling: politiche per definire l'ordine di trasmissione dei datagrammi

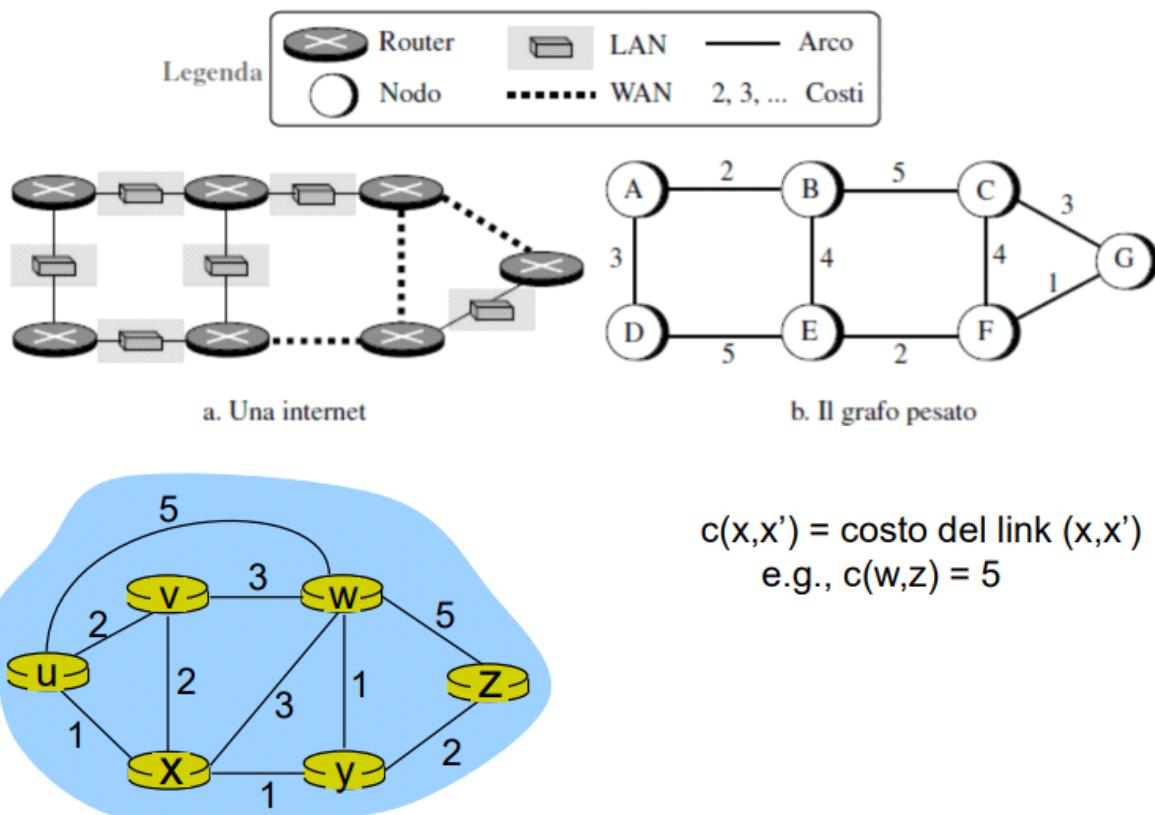
Ritardi e perdite:

Accodamento e buffer overflow (perdita) nei buffer di I/O

Routing:

Unicast: un datagramma è destinato a una sola destinazione

Multicast: il datagramma è destinato a un gruppo di host



key question: qual è il cammino a costo minimo tra u e z ?

routing algorithm: algoritmo che calcola il cammino a costo minimo

Routing statico:

L'amministratore configura manualmente la tabella di inoltro. Usato per reti piccole la cui topologia non varia molto

Routing dinamico:

Protocolli specifici che provvedono automaticamente a inserire nella tabella le entry relative ai possibili percorsi

Algoritmi di instradamento

- **Globali:** ciascun router riceve informazioni su tutta la topologia della rete, lancia l'algoritmo e ottiene la tabella di inoltro (link state) -> architettura centralizzata
 - o l'algoritmo riceve in ingresso informazioni su tutti i collegamenti tra i nodi e i loro costi

- Decentralizzati: nessun nodo conosce la topologia di tutta la rete, comincia a calcolare i percorsi andando a ricevere le info dai nodi collegati direttamente, dopo un tot di aggiornamenti riesce a costruirsi percorsi più precisi (*algoritmo distance vector*)
 - o Ogni nodo elabora un vettore di stima dei costi verso tutti gli altri nodi della rete
 - o Il cammino a costo minimo viene calcolato in modo distribuito e iterativo scambiandosi informazioni con i nodi vicini

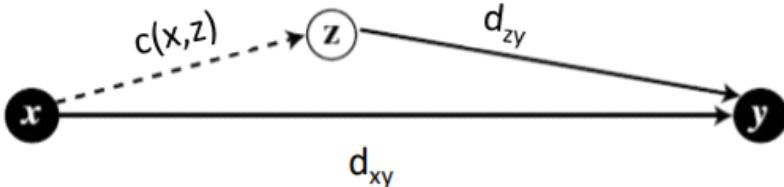
Algoritmo Distance vector

Su un nodo riceve parte delle informazioni da uno o più nodi direttamente connessi (vicini) effettua calcoli e comunica i risultati ai vicini

- Iterativo
 - Asincrono: i nodi non sono sincronizzati sulla topologia della rete

Per il calcolo del percorso minimo si basa sull'equazione di bellman ford, ci detta quali sono le info che un nodo deve mandare agli altri per aggiornare la sua tabella

$$d_{xy} = \min(d_{xy}, (c(x,z) + d_{zy}))$$



Equazione di Bellman-Ford:

Formula per trovare il percorso a costo minimo tra un nodo sorgente x e un nodo di destinazione y, tramite dei nodi intermedi (vicini v)

$C(x, v)$ costo tra nodo sorgente x e il vicino v sia noto

D: distanza tra x e y, è una metrica, tra x e y ci possono essere altri nodi fisici

Il costo è il costo del collegamento fisico, la distanza è il costo di un percorso, vado ad astrarre il collegamento fisico da x a y

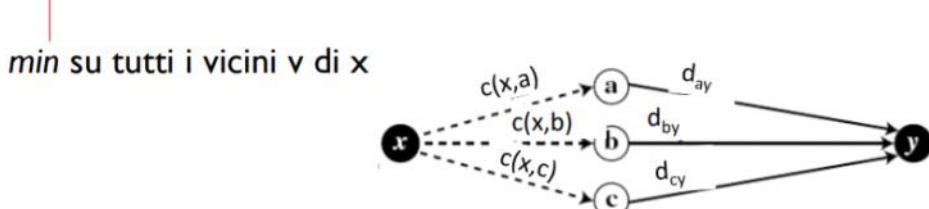
Bellman-Ford equation

d_{xy} := costo del cammino a costo minimo da x a y

allora

$$d_{xy} = \min_v \{ c(x, v) + d_{vy} \}$$

| Costo dal vicino v alla destinazione y
 | costo da x al vicino v

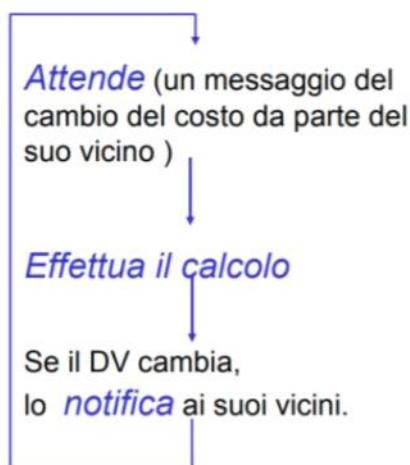


Come funziona l'algoritmo:

- Ciascun nodo x inizia con una stima delle distanze verso tutti i nodi in N
- D_{xy} = stima del costo minimo da x a y per ogni $y \in N$
- Il nodo x mantiene quindi le seguenti informazioni:
 - Conosce il costo verso ciascun vicino v : $c(x,v)$
 - x mantiene il suo vettore distanza $D_x = [D_{xy} : \text{per ogni } y \in N]$
 - Riceve i vettori distanza dei suoi vicini. Per ogni vicino v , x mantiene $D_v = [D_{vy} : \text{per ogni } y \in N]$
- Idea di base:
- Ogni nodo invia una copia del proprio vettore distanza a ciascuno dei suoi vicini.
- Quando un nodo x riceve un nuovo vettore distanza, DV, da qualcuno dei suoi vicini v , lo salva e usa l'equazione di Bellman-Ford per aggiornare il proprio vettore distanza

$$D_{xy} = \min \{c(x,v) + D_{vy}\} \text{ per ogni } y \in N$$
- Finché tutti i nodi continuano a cambiare i propri DV in maniera asincrona, ciascuna stima dei costi D_{xy} converge all'effettivo costo del percorso a costo minimo d_{xy}

Ciascun nodo:



- Iterativo, asincrono: ogni iterazione locale è causata da:
 - Cambio del costo di uno dei collegamenti locali
 - Ricezione da qualche vicino di un vettore distanza aggiornato
- Distribuito:

- Ogni nodo aggiorna i suoi vicini solo quando il proprio DV cambia
- I vicini avvisano i rispettivi vicini solo se necessario

Esempio slide 13

```

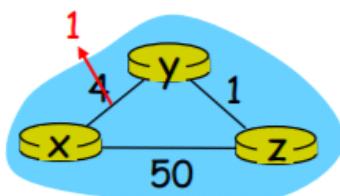
1 Distance_Vector_Routing ( )
2 {
3     // Inizializzazione (creazione dei vettori distanza iniziali del nodo)
4     D[me_stesso] = 0
5     for (y = 1 to N)
6     {
7         if (y è un vicino)
8             D[y] = c[me_stesso][y]
9         else
10            D[y] = ∞
11    }
12    spedisci il vettore {D[1], D[2], ..., D[N]} a tutti i vicini
13    // Aggiornamento (usare il vettore ricevuto dal vicino per aggiornare quello locale)
14    repeat (sempre)
15    {
16        wait (un vettore D_w da un vicino w o un qualsiasi cambiamento negli archi)
17        for (y = 1 to N)
18        {
19            -D[y] = min {D[y], (c[me_stesso][w] + D_w[y])} // Equazione di Bellman-Ford
20        } D[y] = min_v [ c[me_stesso][v]+D_v[y] ]
21        if (c'è un cambiamento nel vettore)
22            spedisci il vettore {D[1], D[2], ..., D[N]} a tutti i vicini
23    }
24 } // Fine dell'algoritmo distance-vector routing

```

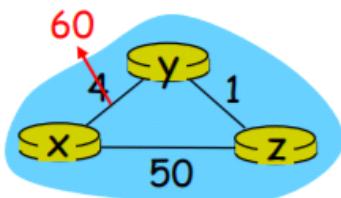
Link cost changes:

Se cambia il costo del collegamento:

- Il nodo rileva il cambiamento del costo
- Ricalcola il vettore distanza
- Se il DV cambia, avvisa i vicini



Il costo da 4 diventa 1, il nodo rileva il cambiamento, ricalcola il vettore distanza e lo manda ai vicini, e i vicini a loro volta aggiornano le loro se gli cambia qualcosa



Il costo del collegamento aumenta -> può portare a dei problemi

Y vede che il collegamento aumenta la distanza, ma Z gli aveva detto che raggiungeva X con costo 5, Y quindi Y pensa che con un costo di 6 può raggiungere X, Y lo manda a Z che aggiorna di nuovo e così via per un po', può crearsi un loop infinito (problema conteggio verso l'infinito)

Count-to-infinity problem:

- Split-horizon with poisoned reverse

Se un nodo X inoltra V i pacchetti destinati a Z allora X invia a V che la distanza verso Z è infinito, le rotte ricevute tramite un'interfaccia devono essere pubblicate indietro a que

Il'interfaccia con una metrica non raggiungibile

Algoritmo link state

Algoritmo globale

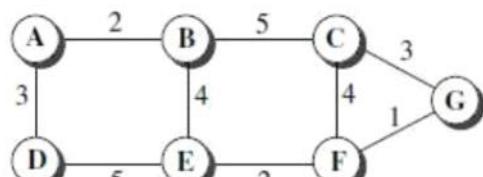
La topologia di rete e tutti i costi dei collegamenti sono noti a tutti i nodi attraverso il "link-state-broadcast"

Tutti i nodi dispongono delle stesse info

Calcola il cammino a costo minimo da un nodo a tutti gli altri nodi della rete, si usa l'algoritmo di dijkstra

Crea una tabella di inoltro per quel nodo

Link state database

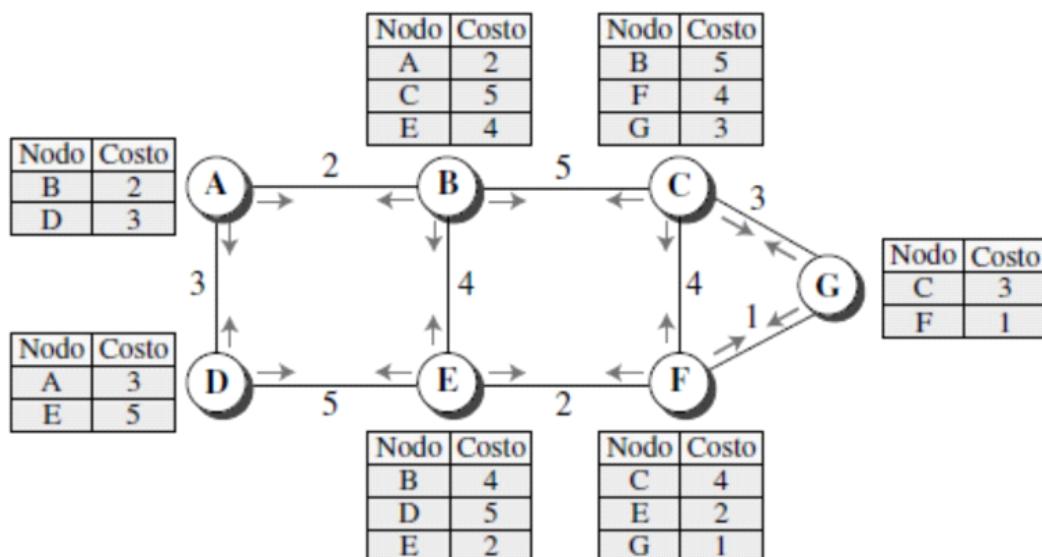


a. Il grafo pesato

	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

b. Il link-state database

LSP (LS Packet) creati ed inviati da ciascun nodo



Notazione

- $c(x,y)$: costo dei collegamenti dal nodo x al nodo y
 - ∞ se non sono adiacenti.
- $D(v)$: costo del cammino dal nodo origine alla destinazione v per l'iterazione corrente
- $p(v)$: immediato predecessore di v lungo il cammino
- N' : sottoinsieme di nodi per cui il cammino a costo minimo dall'origine è definitivamente noto

Algoritmo di dijkstra

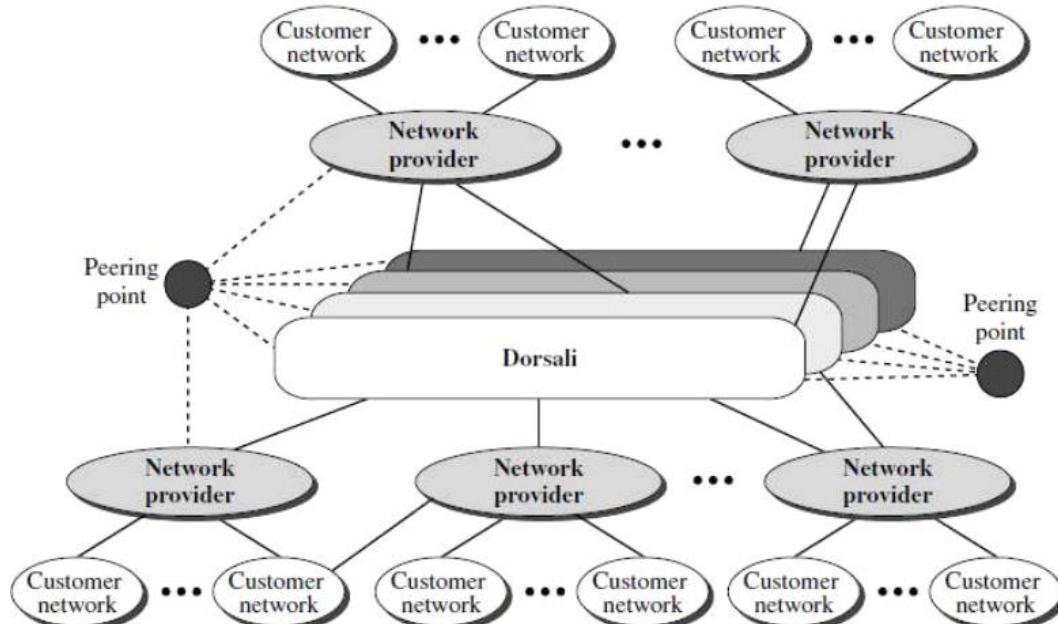
```
1 Initialization of node u:
2    $N' = \{u\}$ 
3   Per tutti i nodi  $v$ 
4     se  $v$  è adiacente a  $u$ 
5       allora  $D(v) = c(u,v)$ 
6       altrimenti  $D(v) = \infty$ 
7
8 Loop
9   trova  $w$  non in  $N'$  tale che  $D(w)$  sia minimo
10  aggiungi  $w$  a  $N'$ 
11  aggiorna  $D(v)$  per tutti  $v$  adiacenti a  $w$  e non in  $N'$  :
12    D(v) = min( D(v), D(w) + c(w,v) )
13  /* il nuovo costo verso  $v$  è il vecchio costo verso  $v$  oppure il
     costo del cammino minimo noto verso  $w$  più il costo da  $w$  a  $v$  */
14 Finché tutti i nodi sono in N'
```

LS vs DV

- Complessità dei messaggi:
 - LS: con n nodi, E collegamenti, implica l'invio di $O(nE)$ messaggi.
 - DV: richiede scambi tra nodi adiacenti. Il tempo di convergenza può variare.
- Velocità di convergenza:
 - LS: l'algoritmo $O(n^2)$ richiede $O(nE)$ messaggi.
 - DV: può convergere lentamente. Può presentare cicli d'instradamento. Può presentare il problema del conteggio all'infinito.
- Robustezza: cosa avviene se un router funziona male?
- LS:
 - un router può comunicare via broadcast un costo sbagliato per uno dei suoi collegamenti connessi (ma non per altri).
 - I nodi si occupano di calcolare soltanto le proprie tabelle.
- DV:
 - un nodo può comunicare cammini a costo minimo errati a tutte le destinazioni.
 - la tabella di ciascun nodo può essere usata degli altri.
 - Un calcolo errato si può diffondere per l'intera rete.

39

Struttura di internet



Instradamento gerarchico

I router sono organizzati in sistemi autonomi

Sistema autonomo

Gruppo di router sotto lo stesso controllo amministrativo (es. router e collegamenti di un ISP, un ISP può comunque dividere la sua rete in più AS)

Dentro un sistema autonomo, l'amministratore sceglie quale protocollo di routing usare (LS o DV) ci sono delle ISP che possono decidere di partizionare in più SA la stessa rete

Instradamento gerarchico

I router sono organizzati in sistemi autonomi (AS: gruppi di router sotto lo stesso controllo amministrativo)

La rete è un'interconnessione di AS

- Instradamento dentro un sistema autonomo: **Interior Gateway Protocol (IGP)**
- Protocollo di routing tra sistemi autonomi: **Exterior Gateway Protocol (EGP)**

All'interno di un AS l'amministrazione sceglie che protocollo usare, tra AS si usa sempre EGP

Tipi di AS:

- **Stub**: collegato solo a un altro AS
- **Multihomed**: collegato a più di un altro AS (ma trasporta - come stub - solo traffico di cui è origine o destinazione) non trasportano traffico di altri AS, ma solo traffico originato da loro e uscente (es. azienda che usa più ISP)
- **Transito** inoltro di traffico da un AS a un altro

Ogni AS ha un identificativo associato (assegnato da IANA) (ASN)

Routing INTRA-AS e INTER-AS

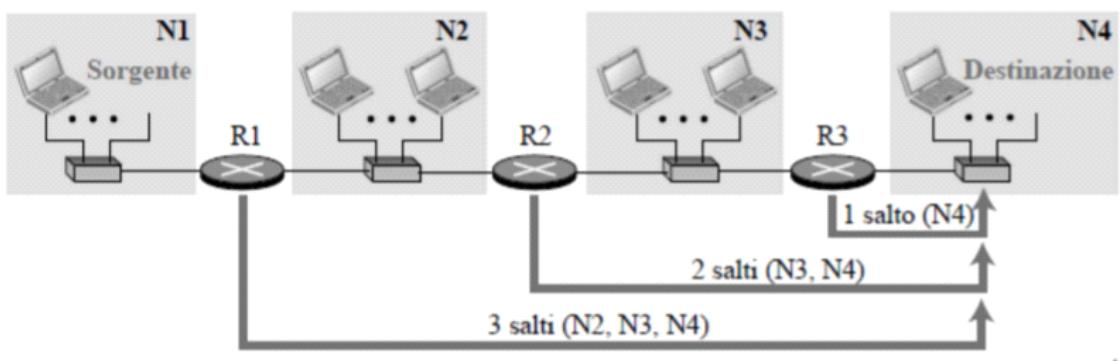
- INTRA-AS routing protocol determina (da solo) rotte per le destinazioni interne ad AS
 - o RIP
 - o OSPF
- INTRA-AS e INTER-AS routing protocol determinano (insieme) le rotte per le destinazioni *esterne* all'AS
 - o Border gateway protocol
 - o Standard de facto
 - o Consente di conoscere le destinazioni raggiungibili attraverso sistemi autonomi vicini
 - o Propaga le informazioni di raggiungibilità ai router interni del proprio AS
 - o Determina percorsi buoni verso le sottoreti di destinazione

Protocolli INTRA-AS

RIP (Routing Information Protocol)

Routing information protocol, implementa l'algoritmo DV con poisoned Reverse.

Metrica per la distanza: nel RIP si usa il numero di sottoreti attraversate (max 15) (inf = 16)



Esempio

Tabella d'inoltro per R1

Rete di destinazione	Prossimo router	Costo (in hop)
N1	—	1
N2	—	1
N3	R2	2
N4	R2	3

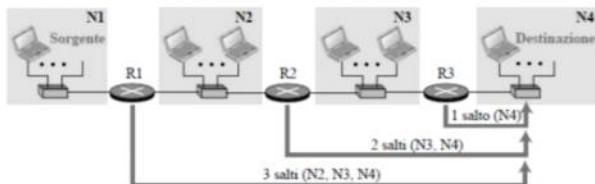


Tabella d'inoltro per R3

Rete di destinazione	Prossimo router	Costo (in hop)
N1	R2	3
N2	R2	2
N3	—	1
N4	—	1

Tabella d'inoltro per R2

Rete di destinazione	Prossimo router	Costo (in hop)
N1	R1	2
N2	—	1
N3	—	1
N4	R3	2

47

Domanda possibile ^

RIP utilizza un processo *demone* che usa UDP sta in ascolto sulla porta 520.

Ogni 30 sec si scambiano i distance vector (oppure se cambia la tabella di inoltro)

Algoritmo

Un nodo invia la tabella di inoltro ai propri vicini (periodi di circa 30 secondi)

Nodo R riceve advertisement da vicino V con $D_V[y]$:

$$D_R[y] = 1 + D_V[y]$$

Aggiunge un hop ($c(x,v)$) e considera come next-hop V

Il nuovo percorso viene inserito in tabella se:

- se è un percorso non presente nella tabella e quindi viene aggiunto
- se $D_V[y] + 1 < D_{\text{Old}}[y]$ costo ricevuto inferiore a quello del vecchio percorso
- Se V è nextHop del vecchio e nuovo percorso, ma il costo è cambiato (diminuito o incrementato)

$D_V[y]$: distance vector di y -> distanza di v da y

OSPF (Open Shortest Path First) (Intra AS)

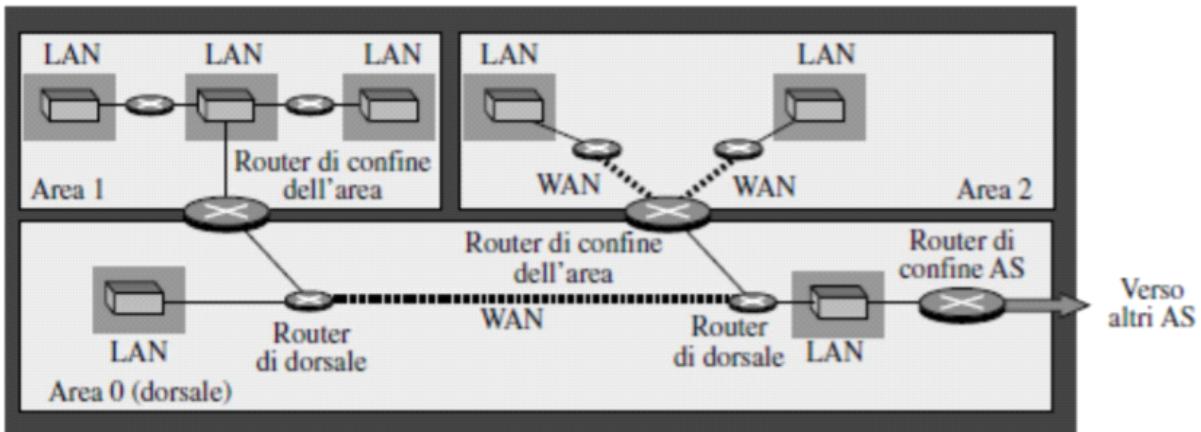
Algoritmo: Link State

Metrica decisa dall'amministratore (latenza, affidabilità, banda, numero di hop..)

RIP viene usato per reti medio-piccole, OSPF può essere usato per reti più grandi.

L'AS può essere suddiviso in aree (per ridurre flooding di Link State Packet), una delle aree fa da dorsale. (i pacchetti link state vengono mandati in broadcast)

Sistema autonomo (AS)



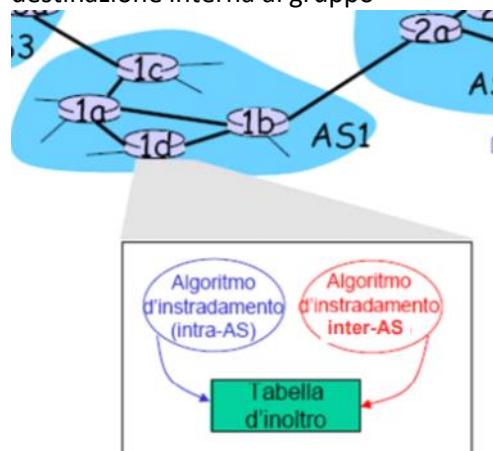
Si spezza la comunicazione broadcast, i messaggi vengono mandati attraverso le aree, per arrivare alle altre aree devono passare dalla backbone

Usa IP (porta 89)

Routing INTER-AS

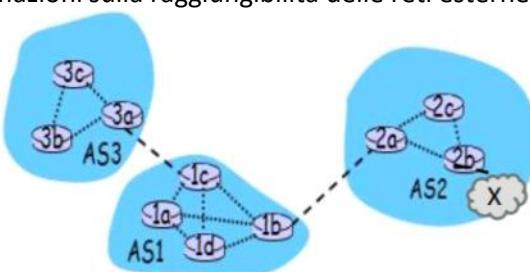
Routing TRA sistemi autonomi

Ciascun sistema autonomo sa come inoltrare i pacchetti lungo il percorso ottimo verso qualsiasi destinazione interna al gruppo



Gateway router si scambiano informazioni sulla raggiungibilità delle reti esterne

Esempio



- AS1 scopre (grazie a INTER-AS routing protocol) che una sottorete X è raggiungibile via AS2 (a cui AS1 è collegato mediante il gateway 1b)
- AS1 propaga (con INTER-AS routing protocol) tale informazione al suo interno
- Un router R (es. 1d in figura) di AS1 riceve l'informazione “rete X raggiungibile via AS2”:

Aggiorna (se necessario) la tabella di inoltro

BGP (Border Gateway Protocol)

BGP4: unico protocollo INTER-AS usato in Internet

Protocollo per gestire comunicazioni inter-as, è un protocollo unico.

Andiamo a distinguere le sessioni esterne e interne

Quando vengono "annunciate" delle rotte, vengono annunciate il più possibile con indirizzi aggregati

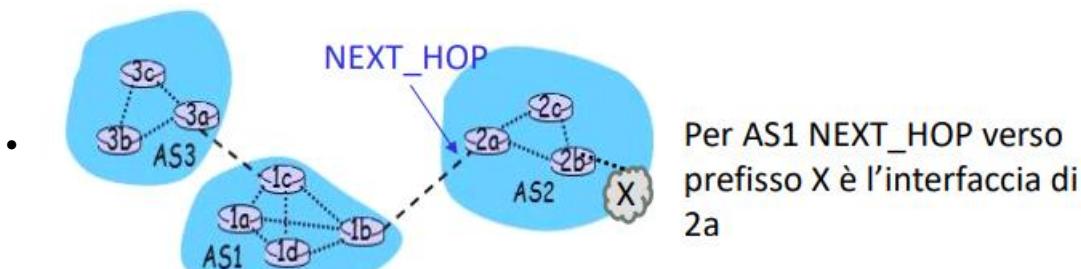
Pubblicizzare un prefisso significa impegnarsi a instradare pacchetti destinati a reti in quel prefisso

Un gateway riceve info sulla sessione eBGP da altri AS e distribuisce informazioni ai router interni con sessioni iBGP

In BGP il percorso migliore non è necessariamente quello a costo minimo perché si parla di diverse amministrazioni (magari vodafone da tim non ci vuole passare)

Advertisement (ADV) BGP

- "route" = prefisso + attributi
- Attributi più importanti:
 - o AS_PATH:
 - Sequenza degli AS attraversati nel path pubblicato dall'advertisement
 - Usato per scartare adv già ricevuti, scegliere tra più percorsi per lo stesso prefisso
 - o NEXT_HOP
 - Indica l'indirizzo IP del primo router lungo un percorso annunciato (al di fuori dell'AS che riceve l'annuncio) a un dato prefisso di rete



Politiche di importazione: quando un gateway riceve un ADV usa tali politiche per accettare/rifiutare ADV

Scelta delle rotte: un router può ricevere più di 1 rottta per lo stesso prefisso

Sequenza di regole:

1. Attributo di "preferenza locale" (LOCAL-PREF scelta da amministratore o impostato dai router dell'AS) -> vengono selezionati quelli coi valori più alti
2. Shortest AS-PATH
3. Closest NEXT-HOP interface

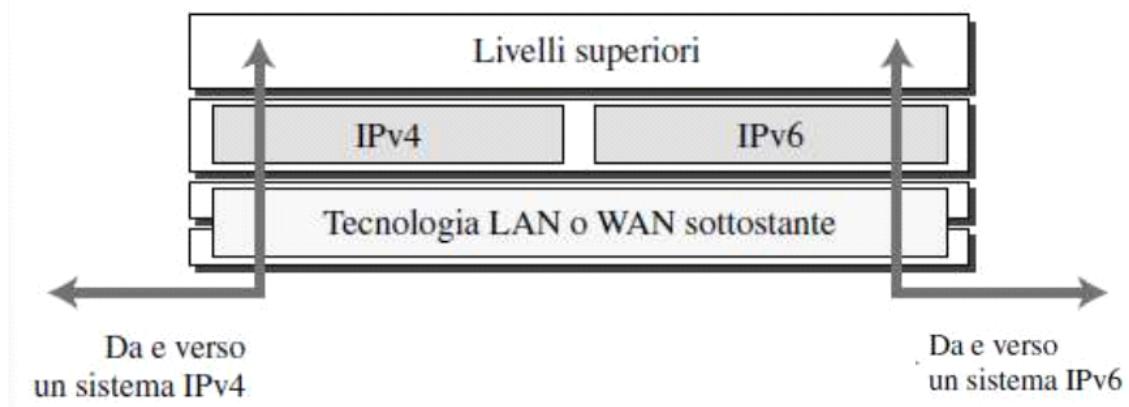
IPv6

0	4	12	16	24	31
Versione	Classe di traffico	Etichetta di flusso			
Lunghezza del payload	Prossima intestazione				Hop limit
Indirizzo sorgente (128 bit = 16 byte)					
Indirizzo destinazione (128 bit = 16 byte)					

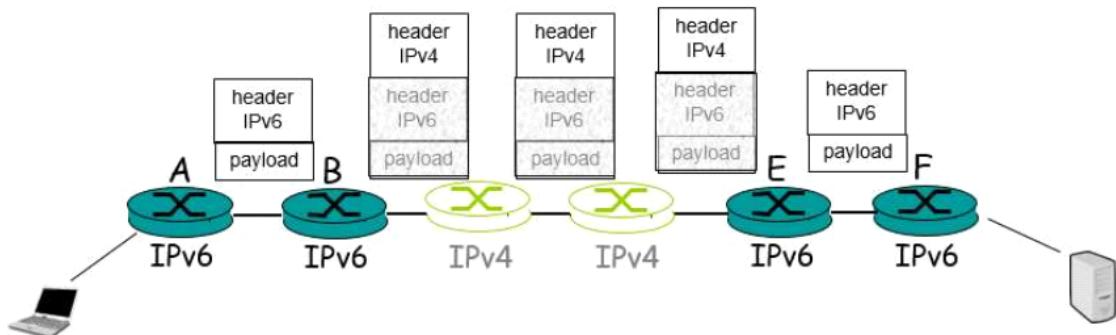
Motivi:

- Esaurimento indirizzi a 32 bit -> indirizzi a 128 bit
- Velocizzare elaborazione e forwarding pacchetti -> lunghezza fissa (40 byte) dell'header, eliminazione del checksum
- Risparmiare ai router costo frammentazione
- Facilitare QoS -> flow label per identificare datagram appartenenti allo stesso flow

Dual stack



Tunneling:



Datagrammi IPv6 come payload di datagrammi tra (interfacce di) router IPv4

LIVELLO LINK

venerdì 21 aprile 2023 18:00

Il livello collegamento muove i datagrammi da un nodo al nodo adiacente su un singolo link di comunicazione

Terminologia:

- Host e router: nodi
- Canali di comunicazione:
 - o Collegamenti cablati
 - o Collegamenti wireless
- Unità dati di livello 2: frame
 - o Incapsula un datagram

Un datagram può essere gestito da diversi protocolli su collegamenti differenti.

Anche i servizi erogati dai protocolli del livello di link possono essere differenti. Ad es non tutti forniscono un servizio di consegna affidabile

Collegamenti:

- Punto-punto: collegamento dedicato a due soli dispositivi
- Broadcast: collegamento condiviso tra più dispositivi. Quando un nodo trasmette un frame, il canale lo diffonde e tutti gli altri nodi ricevono una copia

Livello suddivisibile in due sottolivelli:

- Data-Link Control
- Medium Access Control
 - o Accesso al medium
 - o Regole per accedere al medium

Servizi offerti:

- Framing:

Frame: campo dati, intestazione e eventuale trailer

I protocolli incapsulano i datagram del livello di rete all'interno di un frame a livello di link

Il framing separa i vari messaggi durante la trasmissione da una sorgente a una destinazione.

Per identificare src e dest si usano indirizzi MAC

- Consegna affidabile:

Considerata non necessaria nei canali più affidabili -> bassi errori sui bit (fibra ottica, coassiale, doppino)

È spesso utilizzata nei collegamenti ad alto tasso di errori (es. wireless)

- Controllo di flusso

Evita che il nodo trasmittente saturi quello ricevente

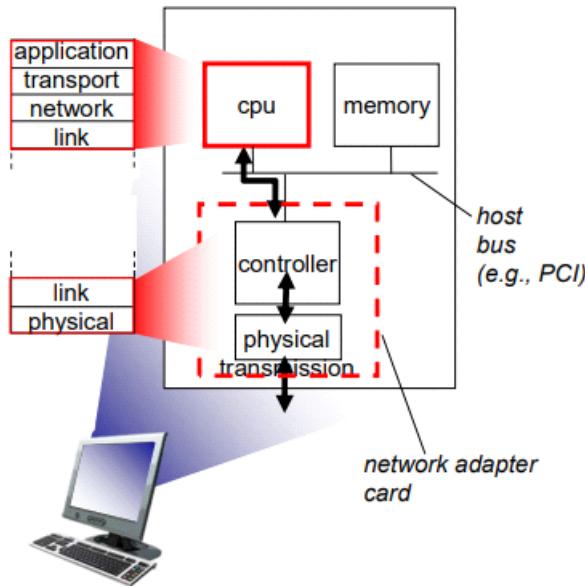
- Rilevazione degli errori

Gli errori sono causati da attenuazione del segnale e rumore. Il nodo ricevente individua la presenza di errori grazie a un bit di controllo inserito dal trasmittente all'interno del frame

- Correzione degli errori

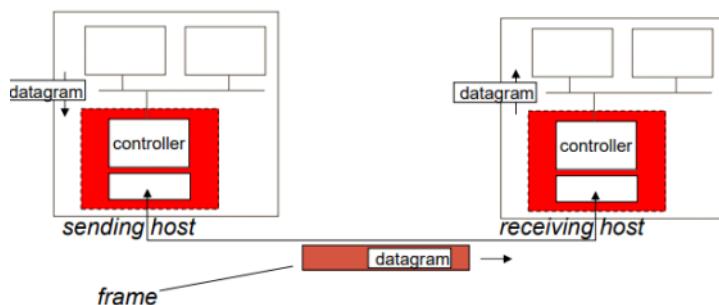
Il nodo ricevente determina il punto in cui si è verificato l'errore e lo corregge

Dove è implementato?



Implementato nella scheda di rete ("adaptor"), in ogni host. Collegato al system bus dell'host.

Comunicazione tra adaptor:



Mittente:

- Incapsula il datagram in un frame
- Aggiunge error checking bits, rdt, flow control ecc.

Destinatario

- Controllo errori, rdt, flow control ecc.
- Estrae il datagramma e lo passa a livello superiore

Collegamenti ad accesso multiplo

Canale di broadcast singolo e condiviso-> due o più trasmissioni simultanee dai nodi -> interferenza -> *collisione* se un nodo riceve due o più segnali nello stesso istante

Protocollo di accesso multiplo

Algoritmo distribuito che determina quale nodo può trasmettere. La comunicazione per la condivisione del canale usa il canale stesso

Protocollo di accesso multiplo ideale

Dato: canale broadcast con rate R bps

Desiderata:

1. Quando un solo nodo trasmette, può inviare dati con un rate di R bps
2. Quando M nodi trasmettono, ciascuno trasmette con rate di R/M bps
3. Decentralizzato (non ci sono nodi speciali che controllano la trasmissione)

Protocolli MAC

Tre classi principali:

- A suddivisione del canale:

Dividono il canale in "pezzi" più piccoli (risorse), la risorsa è allocata al nodo in modo esclusivo

- *Ad accesso random:*

Il canale è condiviso, ci possono essere collisioni, si hanno meccanismi per recuperare da eventi di collisione

- *A rotazione*

Channel partitioning MAC protocols:

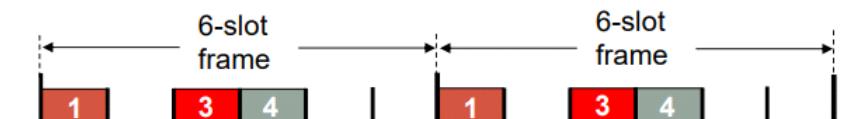
TDMA (time division multiple access)

Accesso al canale in intervalli di tempo. Ciascun intervallo di tempo è suddiviso in time slot.

Ogni stazione ha a disposizione un time slot di lunghezza fissa (len = packet transmission time) in ogni intervallo

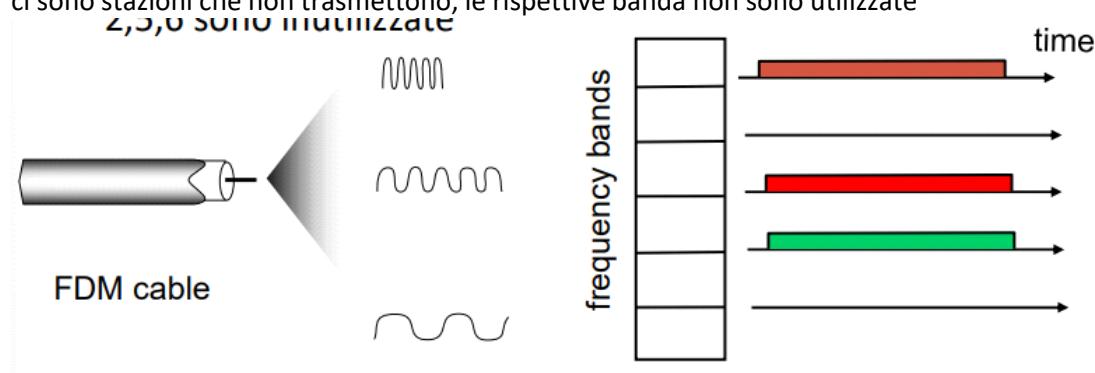
- **Gli slot non utilizzati sono sprecati**

- esempio: 6-station LAN, 1,3,4 hanno dati da inviare, slots 2,5,6 non utilizzati



FDMA (frequency division multiple access)

Spettro del canale diviso in bande di frequenza, ogni stazione usa una banda di frequenza fissata, se ci sono stazioni che non trasmettono, le rispettive banda non sono utilizzate



Protocolli ad accesso random

Quando un nodo deve inviare dei dati li manda al massimo rate di R, senza coordinamento a priori tra i nodi. Si utilizza un protocollo ad accesso random che specifica come rilevare le collisioni e come recuperare dopo le collisioni

Slotted ALOHA

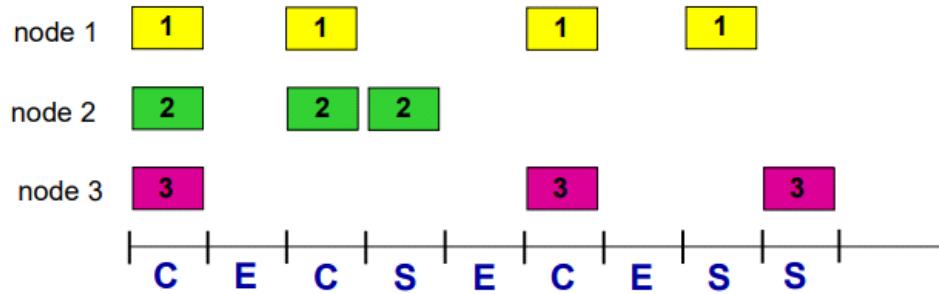
Assunzioni:

- Time slot diviso in slot uguali e fissi
- I nodi iniziano a trasmettere all'inizio dello slot
- Il tempo è discreto e sincronizzato
- Se 2 o più nodi trasmettono, ogni nodo rileva la collisione

Funzionamento

Quando un nodo deve trasmettere un frame, comincia la trasmissione nello slot di tempo successivo

- Se non ci sono collisioni: può inviare un nuovo frame nello slot successivo
- Se rileva collisioni: ritrasmette con probabilità p nello slot successivo finché non ha successo



Pro:

- Il singolo nodo attivo può trasmettere alla massima velocità offerta dal canale
- Decentralizzato: devono essere sincronizzati gli slot di tempo

Contro:

- Collisioni, spreco di slot
- Se c'è un gran numero di host si usa il canale solo il 37% del tempo con successo

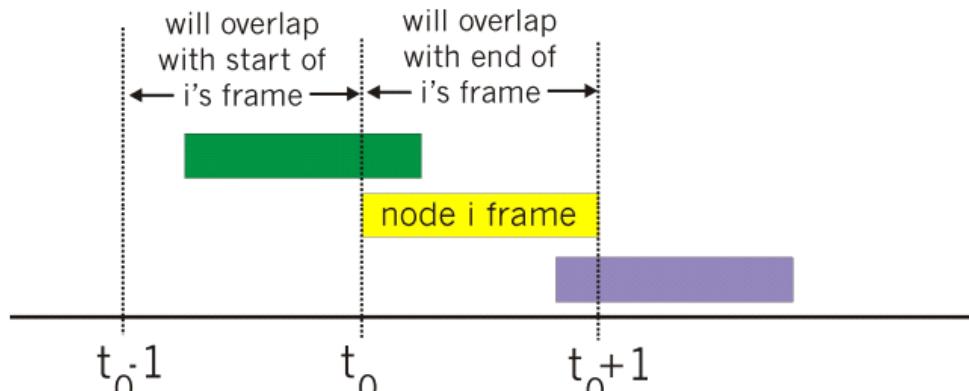
ALOHA puro

Più semplice, non c'è sincronizzazione dei time slot, non ci sono time slot.

Quando un nodo ha dati da inviare li trasmette.

Ha probabilità di collisione maggiore rispetto allo slotted ALOHA

- Un frame inviato a t_0 collide con altri frame inviati nell'intervallo $[t_0-1, t_0+1]$



CSMA (carrier sense multiple access)

Ascoltare prima di parlare

Se il canale è libero il nodo trasmette l'intero frame altrimenti ritarda la trasmissione

Le collisioni possono ancora avvenire: a causa del ritardo di propagazione, due nodi potrebbero non rilevare le comunicazioni reciproche

Collisione: tempo di trasmissione del pacchetto sprecato.

La *distanza* e il *ritardo di propagazione* influiscono sulla probabilità di rilevare collisioni

CSMA/CD (collision detection)

Le trasmissioni che collidono vengono abortite, minori perdite. Facile in LAN cablate difficile in LAN wireless

Ethernet CSMA/CD algorithm

1. NIC (network interface card) riceve il datagramma dal livello rete e crea il frame
2. Se il canale è in idle si trasmette, altrimenti si aspetta
3. Se si trasmette l'intero frame senza collisione bene
4. Se NIC rileva un'altra trasmissione mentre trasmette si ferma e manda un jam signal (segnale di disturbo) che avverte le altre stazioni della collisione
5. Dopo aver interrotto la connessione NIC entra in binary (exponential) backoff:
 - Dopo la collisione m NIC sceglie K random $\{0, 1, \dots, 2^m - 1\}$ e aspetta $K * 512$ bit times,

torna allo step 2. (con più collisioni si aspetta sempre di più)

Protocolli MAC a "rotazione"

Polling:

Il nodo master "invita" i nodi slave a trasmettere a rotazione.

Svantaggi:

- Polling overhead (il master deve coordinare tutto con cose inutili)
- Latenza
- Single point of failure (master)

Token passing:

Token passato da nodo a nodo per essere autorizzato a comunicare

Svantaggi:

- Token overhead
- Latenza
- Single point of failure (token)

• *Suddivisione del canale*

- Time Division, Frequency Division

• *random access* (dynamic),

- ALOHA, S-ALOHA, CSMA, CSMA/CD
- carrier sensing: easy in some technologies (wire), hard in others (wireless)
- CSMA/CD in Ethernet
- CSMA/CA in 802.11

• *rotazione*

- polling from central site, token passing
- Bluetooth, FDDI, token ring

Indirizzi a livello link

Un indirizzo a livello link è associato alla scheda di rete ed è tipicamente permanente.

Detto anche indirizzo fisico/LAN/MAC (Media Access Control)

Per le LAN Ethernet e IEEE 802.11 è lungo 6 byte (2^{48} possibili indirizzi) ed è espresso in esadecimale

Gestione dell'univocità:

IEE assegna i primi 24 bit OUI (Organization Unique Identifier) i restanti sono gestite dalle aziende e assegnati a livello locale. Quando un'azienda vuole costruire adattatori compra un blocco di spazio di indirizzi

Indirizzi MAC e ARP

MAC:

Struttura piatta (non cambia mai), analogo al codice fiscale, 48 bit

IP:

Indirizzo a livello di rete, analogo all'indirizzo postale: struttura gerarchica e deve essere aggiornato quando si cambia rete

Indirizzi LAN

Un adattatore inserisce l'indirizzo MAC di destinazione per spedire un frame.

In LAN broadcast il frame sarà ricevuto ed elaborato da tutti gli adattatori della LAN

Ogni scheda controlla se l'indirizzo MAC corrisponde al proprio; in caso positivo estrae il datagram e

passa al liv superiore, in caso negativo lo scarta

Se un adattatore trasmittente vuole che tutte le schede di rete passino i dati agli strati superiori, immette nel campo di dest. FF-FF-FF-FF-FF-FF (indirizzo di broadcast)

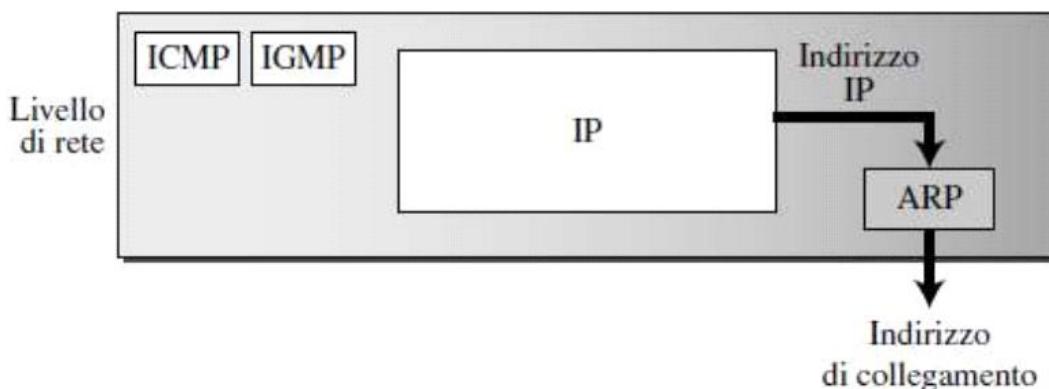
ARP (Address Resolution Protocol)

Risoluzione degli indirizzi

Problema:

All'accensione, una macchina conosce: il suo indirizzo MAC, il suo indirizzo IP. Non sa chi ha attorno. Per questo si usa ARP. Le richieste ARP vengono fatte in broadcast. Conoscendo l'IP ci dà un indirizzo MAC.

Tipicamente è considerato un protocollo di rete, perché il payload viene incapsulato a livello di rete



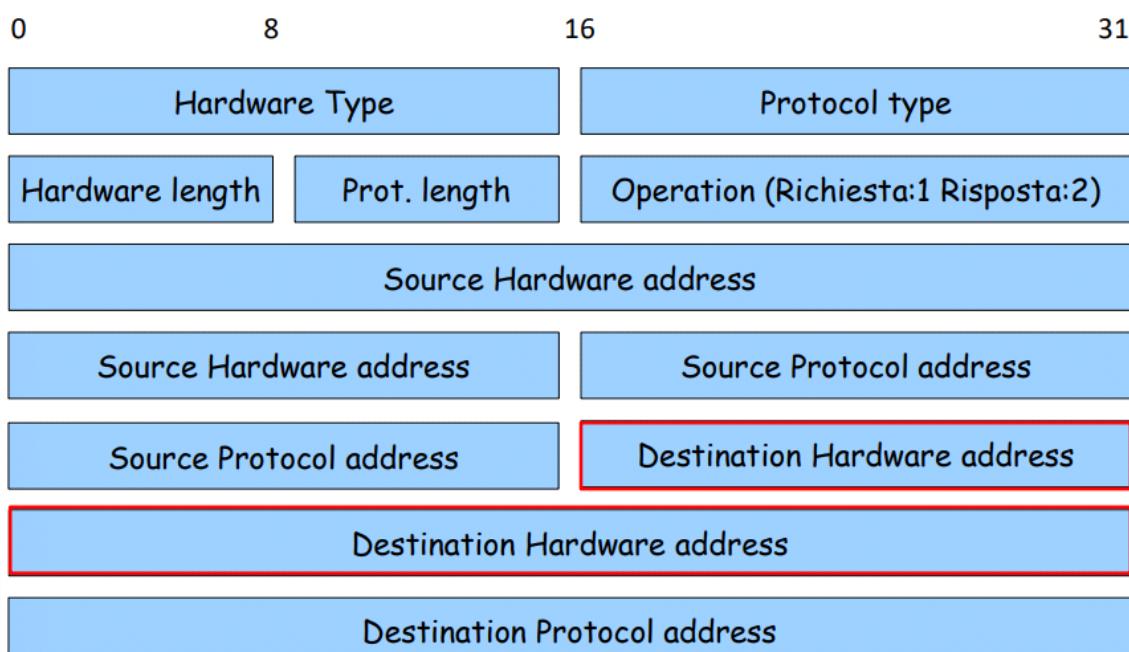
Ogni nodo IP nella LAN ha una tabella ARP:

Tabella ARP:

Contiene corrispondenza tra IP e MAC <Indirizzo IP, Indirizzo MAC, TTL> (TTL indica quando la voce nella tabella andrà eliminata, tipicamente è 20 min)

ARP risolve gli indirizzi IP solo per nodi nella stessa LAN, le tabelle ARP contengono la corrispondenza fra IP e MAC per nodi della stessa sottorete. Non li contiene necessariamente tutte

Pacchetto ARP

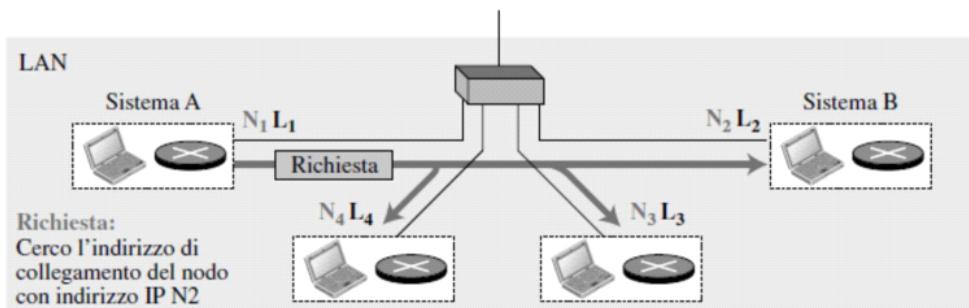


Internet Protocol (IPv4) over Ethernet ARP packet

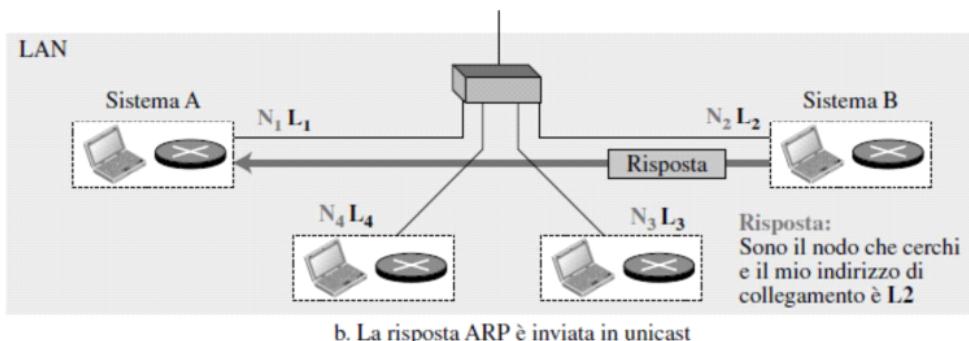
Octet offset	0	1
0	Hardware type (HTYPE)	
2	Protocol type (PTYPE)	
4	Hardware address length (HLEN)	Protocol address length (PLEN)
6	Operation (OPER)	
8	Sender hardware address (SHA) (first 2 bytes)	
10	(next 2 bytes)	
12	(last 2 bytes)	
14	Sender protocol address (SPA) (first 2 bytes)	
16	(last 2 bytes)	
18	Target hardware address (THA) (first 2 bytes)	
20	(next 2 bytes)	
22	(last 2 bytes)	
24	Target protocol address (TPA) (first 2 bytes)	
26	(last 2 bytes)	

- *Hardware type*: protocollo a livello di collegamento (es. Ethernet)
- *Protocol Type*: protocollo di livello superiore
- *Source hw address e source protocol address*: indirizzi del nodo mittente a livello link e superiore. Lunghezza variabile. Campi Hard. Length e prot. Length
- *Destination hw address (vuoto nelle richieste) e dest protocol address*

ARP: funzionamento



a. La richiesta ARP è inviata in broadcast



b. La risposta ARP è inviata in unicast

Il protocollo ARP interagisce direttamente con il livello collegamento.

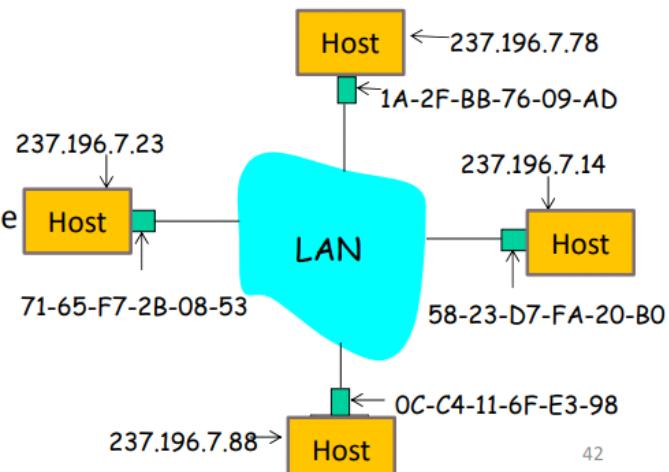
Il pacchetto ARP viene incapsulato in un frame e spedito in broadcast sulla rete.

L'header di livello 2 specifica che il frame contiene un pacchetto ARP

Ogni nodo della rete locale riceve ed elabora il pacchetto con richiesta ARP; il nodo che riconosce il proprio indirizzo IP restituisce un pacchetto di risposta ARP che contiene il proprio indirizzo IP e MAC. Il pacchetto viene mandato in *unicast* al nodo originario

Esempio per Forwarding diretto

- A vuole inviare un datagramma a B, e l'indirizzo MAC di B non è nella tabella ARP di A.
- A trasmette in un pacchetto broadcast il messaggio di richiesta ARP, contenente l'indirizzo IP di B.
 - Indirizzo MAC del destinatario = FF-FF-FF-FF-FF-FF
- Tutte le macchine della LAN ricevono una richiesta ARP.
- B riceve il pacchetto ARP, e risponde ad A comunicandogli il proprio indirizzo MAC.
- il frame viene inviato all'indirizzo MAC di A.

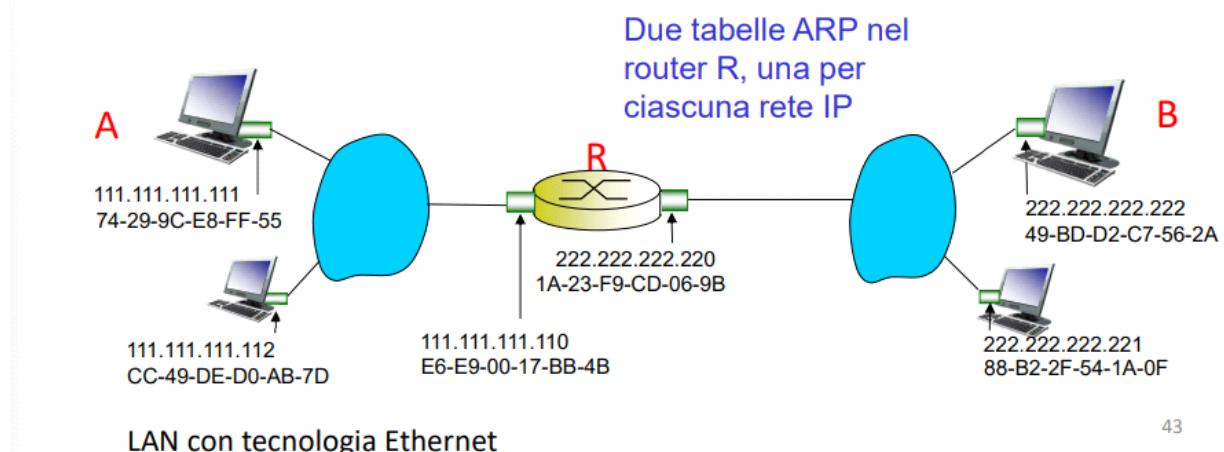


42

Forwarding indiretto

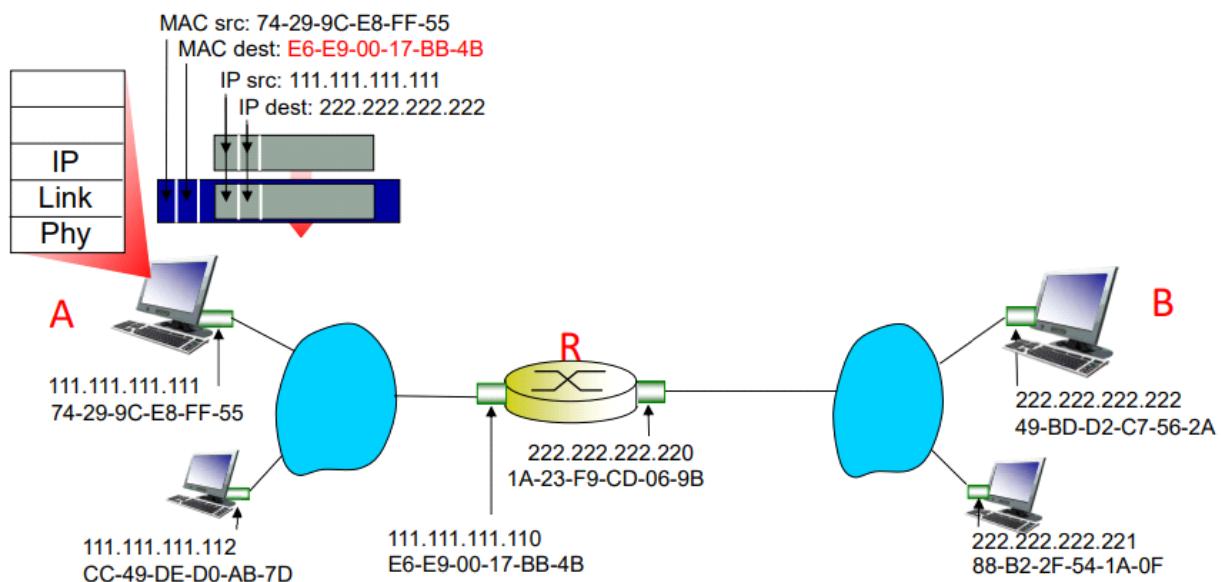
caso: invio di un datagramma con mittente Ip di A e destinatario IP di B

- Attenzione su indirizzamento IP (datagramma) e MAC (frame)
- Ipotesi
 - A conosce l'indirizzo IP di B
 - A conosce l'indirizzo IP del primo router (**come fa?**)
 - A conosce l'indirizzo MAC di R (**come fa?**)

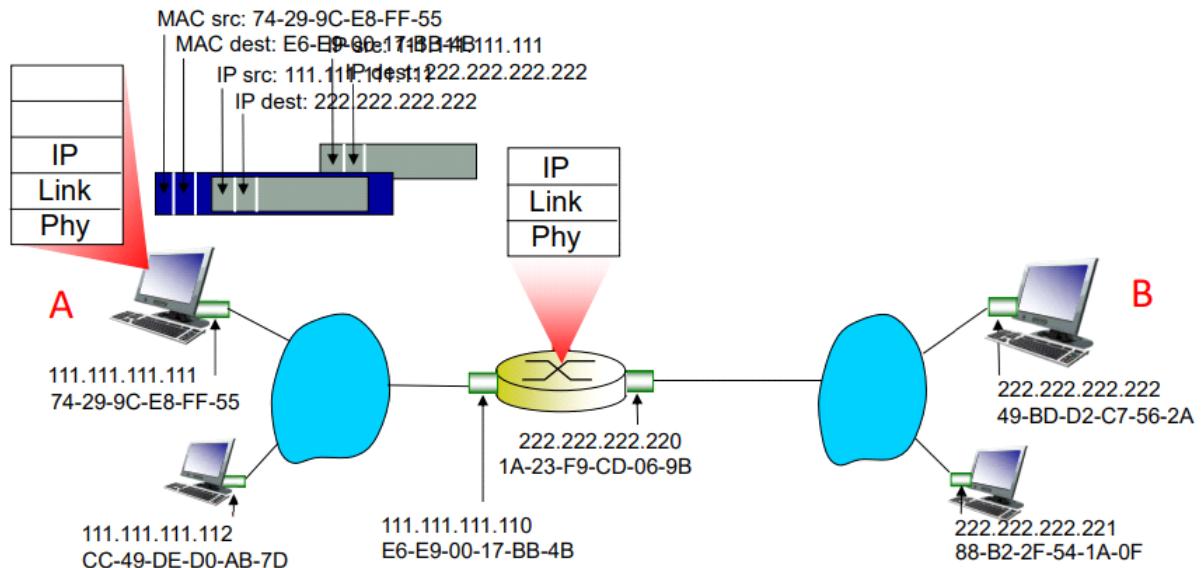


43

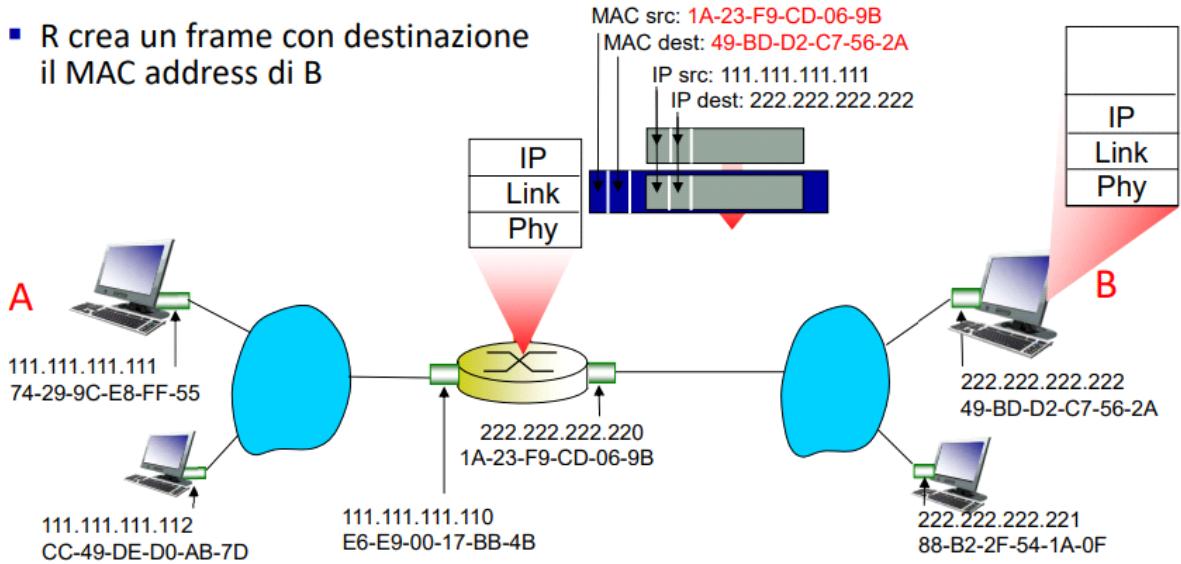
- A crea un datagramma IP con IP sorgente A, destinazione B
- A crea un frame con indirizzo MAC destinazione il MAC address di R, il frame incapsula il datagramma IP (A-to-B)



- frame inviato da A a R
- frame ricevuto da R, decapsulato, il datagramma passa al livello IP



- R estrae il datagramma IP dal frame Ethernet, e vede che la sua destinazione è B.
- R inoltra il datagramma con IP sorgente A, destinazione B
- R usa ARP per ottenere l'indirizzo MAC di B
- R crea un frame con destinazione il MAC address di B



ETHERNET SWITCH VLAN

lunedì 24 aprile 2023 16:44

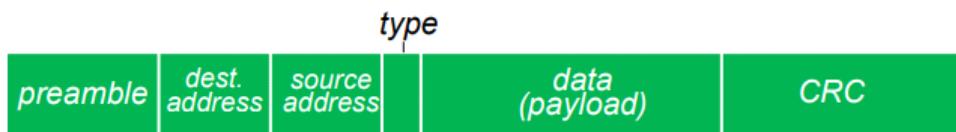
Ethernet:

Tecnologia più utilizzata per le LAN

Topologia fisica:

Topologia a bus col coassiale diffusa fino a metà degli anni 90 seguita dalla topologia a stella al cui centro c'è uno switch (quella che si utilizza adesso)

Pacchetti ethernet:



Preambolo:

I pacchetti ethernet iniziano con un campo di otto byte; i primi sette hanno i bit 10101010 e l'ultimo è 10101011. servono per attivare gli adattatori dei riceventi e sincronizzare gli orologi.

Dati:

l'MTU (maximum transmission unit) può variare da 46 byte a un max di 1500. se il datagram è più grande deve essere frammentato, altrimenti deve essere riempito.

Indirizzo di destinazione: 6 byte

Indirizzo sorgente: 6 byte

Campo tipo (2 byte):

Consente a ethernet di supportare vari protocolli di rete

Controllo CRC

Consente all'adattatore ricevente di rilevare la presenza di un errore nei bit del pacchetto

Ethernet: non affidabile, senza connessione

Connectionless: no handshaking tra nodo mittente e destinatario

Unreliable: nodo in ricezione non invia ack or nack al nodo mittente

- I dati nei frame eliminati (dropped) sono recuperati solo se il trasferimento affidabile è implementato ai livelli superiori
- Protocollo MAC: CSMA/CD con binary backoff

Ci sono molti standard ethernet diversi; hanno in comune protocollo MAC e formato frame

Fast Ethernet

- 100 Mbps
- Mantiene intatti formato e dimensione min/max frame
- Dimensione minima immutata & trasmissione 10 volte più veloce → rete più corta, due soluzioni
 - a) Topologia a bus → hub passivo con topologia a stella e dimensione massima rete 250m (anziché 2500m)
 - b) Usare un commutatore (switch) a livello link, dotato di buffer, e una connessione full-duplex per ogni host → no collisioni

Gigabit Ethernet (802.3z)

- 1000 Mbps = 1 Gbps
- Mantiene invariata lunghezza min/max frame
- Switch al centro della stella. tutti i nodi collegati ai rami della stella → no collisioni

- Mantiene invariata lunghezza min/max frame
- Switch al centro della stella, tutti i nodi collegati ai rami della stella → **no collisioni**

Gigabit Ethernet (802.3z)

- Estende tecnologia, velocità trasferimento, distanza **max copertura**
- 40 Gbps
- Collegamenti punto-punto con switch o mezzi broadcast (es. collegamenti con hub)⁵⁹

Dispositivi di interconnessione

Repeater e Hub

Repeater: operano solo a livello fisico, rigenerano il segnale che ricevono, in passato usati per collegare segmenti di ethernet con topologia a bus

Hub: repeater multi-porta.

Non hanno capacità di filtraggio

Switch di livello link

Operano sia a livello fisico *rigenerando il segnale* che a livello link *verificando indirizzi MAC contenuti in frame*

Hanno una tabella che usano per filtraggio. Non modificano indirizzi MAC in intestazione frame

Ethernet switch

- Link-layer device: ruolo attivo
Store and forward di frame ethernet. Esamina gli indirizzi MAC dei frame in arrivo, inoltra in modo selettivo i frame su uno o più collegamenti. Usa CSMA/CD per accedere al segmento
- Trasparente
Gli host non sono a conoscenza della presenza degli switch
- Plug and play, self learning
Non devono essere configurati

Switch: trasmissioni multiple simultanee

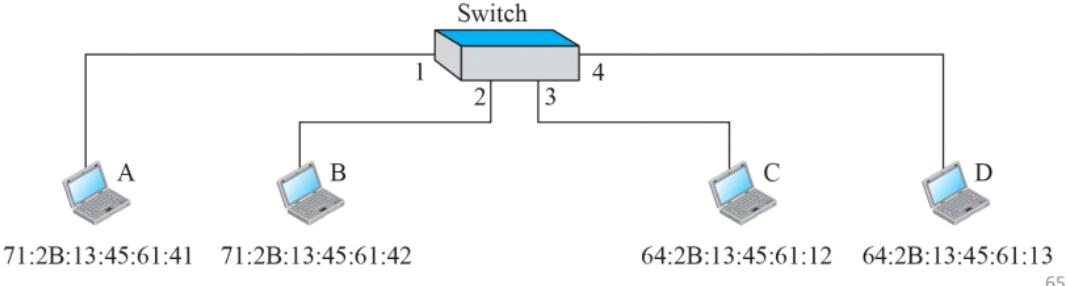
Gli host hanno connessioni dedicate verso lo switch. Gli switch bufferizzano i pacchetti

No collisions; full duplex (permette la comunicazione in entrambe le direzioni simultaneamente): ogni link ha il suo proprio dominio di collisione

Switch con auto-apprendimento

Costruzione graduale della tabella

Indirizzo	Porta	Indirizzo	Porta
a. Originale		71:2B:13:45:61:41	1
b. Dopo che A invia un frame a D			
c. Dopo che D invia un frame a B	1 4	71:2B:13:45:61:41 64:2B:13:45:61:13	1 4
d. Dopo che B invia un frame ad A	2	71:2B:13:45:61:42	2
e. Dopo che C invia un frame a D	3	71:2B:13:45:61:41 64:2B:13:45:61:13 71:2B:13:45:61:42 64:2B:13:45:61:12	1 4 2 3



65

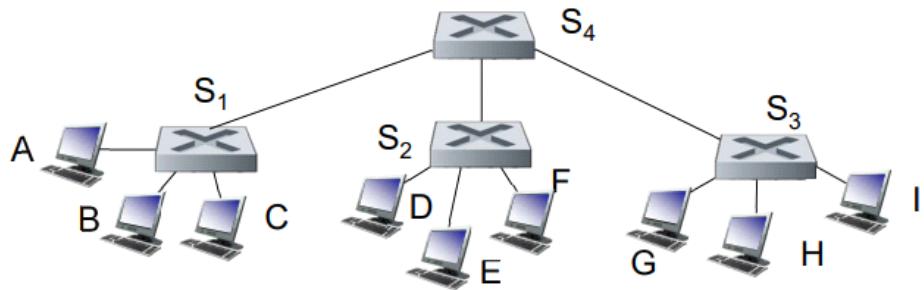
Frame filtering/forwarding

when frame received at switch:

1. record incoming link, MAC address of sending host
2. index switch table using MAC destination address
3. if entry found for destination
 - then {
 - if destination on segment from which frame arrived
 - then drop frame
 - else forward frame on interface indicated by entry
- else flood /* forward on all interfaces except arriving interface */

Interconnecting switches

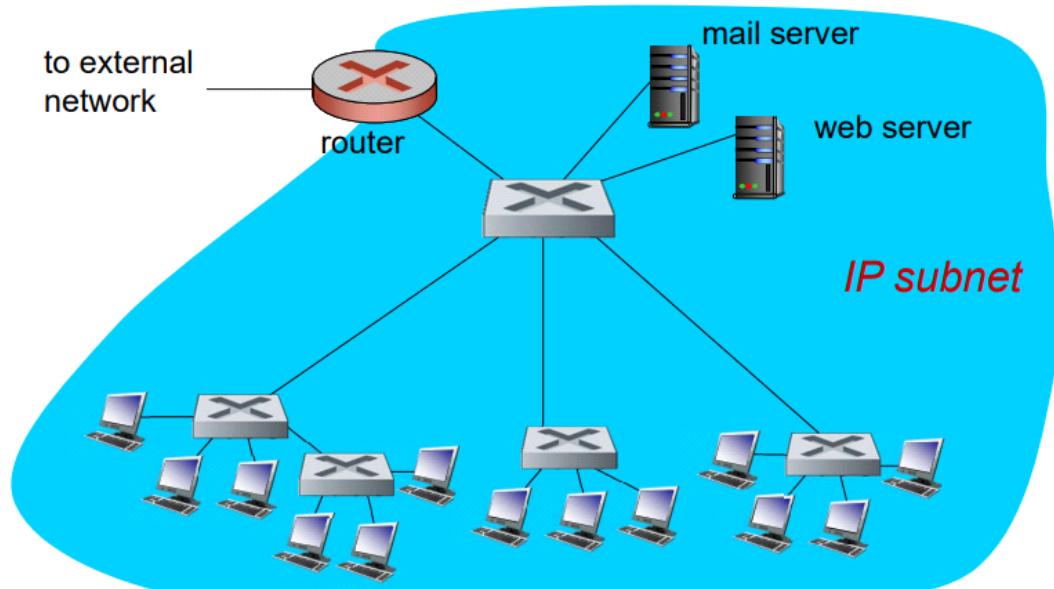
Più switch ad autoapprendimento possono essere collegati:



Q: A invia un frame a G – come fa S_1 ad apprendere che il frame destinato a G deve passare da S_4 e S_3 ?

- **A:** self learning! (esattamente come nel caso dello switch singolo!)

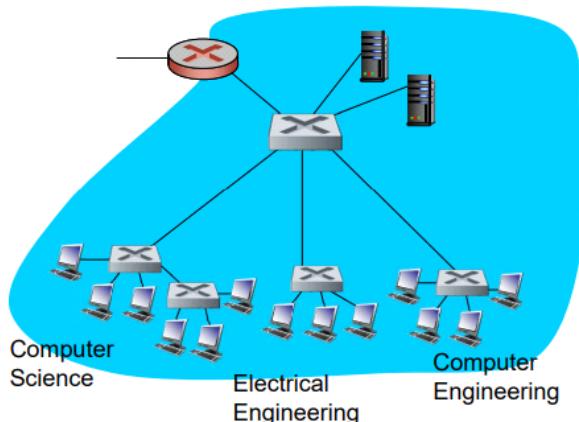
Institutional network



Router:

- Operano
 - Sia a livello fisico: rigenerando segnale
 - Sia a livello link: verificando indirizzi MAC contenuti in frame
 - Sia a livello network: verificando indirizzi IP
- Router \neq repeater e switch/hub
 1. Router hanno 1 indirizzo MAC e 1 indirizzo IP per ogni loro interfaccia
 2. Operano solo sui frame il cui indirizzo destinazione (link) è l'indirizzo (link) dell'interfaccia su cui arrivano
 3. Cambiano indirizzi link contenuti nei frame che inoltrano

VLAN (Virtual local-area Network)



consider:

- CS user moves office to EE, but wants connect to CS switch?
- single broadcast domain:
 - all layer-2 broadcast traffic (ARP, DHCP, unknown location of destination MAC address) must cross entire LAN
 - security/privacy, efficiency issues

Broadcast domain

Il gruppo totale di device che riceve frame broadcast viene chiamato **broadcast domain**

In molte situazioni, un frame broadcast viene utilizzato per un motivo; ad esempio network management o la trasmissione di qualche avviso con significato locale

Se un frame broadcast ha informazioni utili solo a un particolare dipartimento, la capacità di trasmissione è sprecata nelle porzioni di LAN e switch che non ne hanno bisogno

VLAN

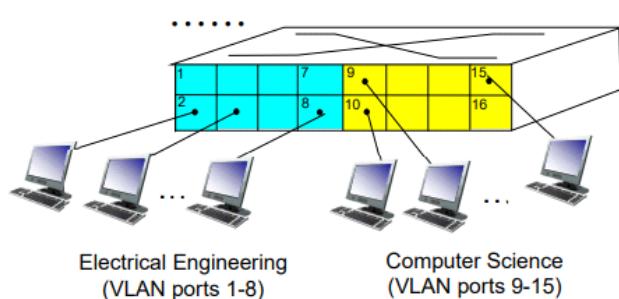
Una VLAN è un sottogruppo logico di una LAN che viene creato tramite software.

Combina le stazioni e i dispositivi di rete in un singolo **broadcast domain** in base a mutuo interesse, indipendentemente dalla LAN fisica

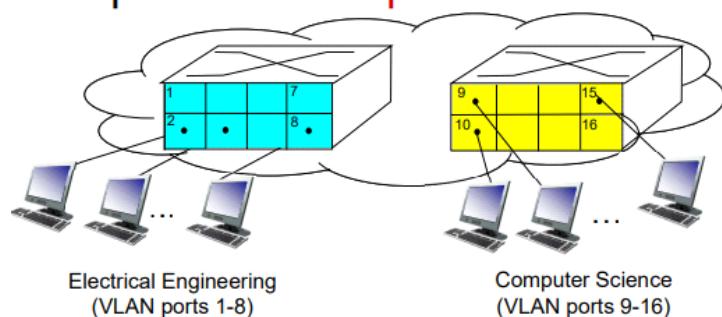
Le VLAN sono implementate tramite switch LAN e funzioni a livello MAC

Per collegare una VLAN all'altra è richiesto un router o uno switch di livello 3.

port-based VLAN: switch ports grouped (by switch management software) so that *single* physical switch



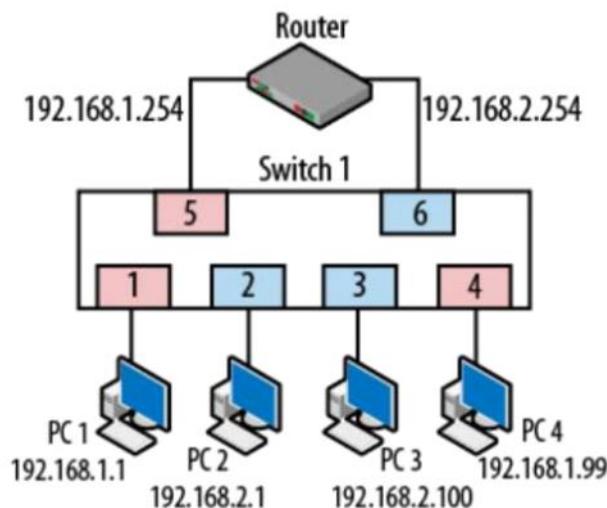
... operates as **multiple virtual switches**



Port-based VLAN

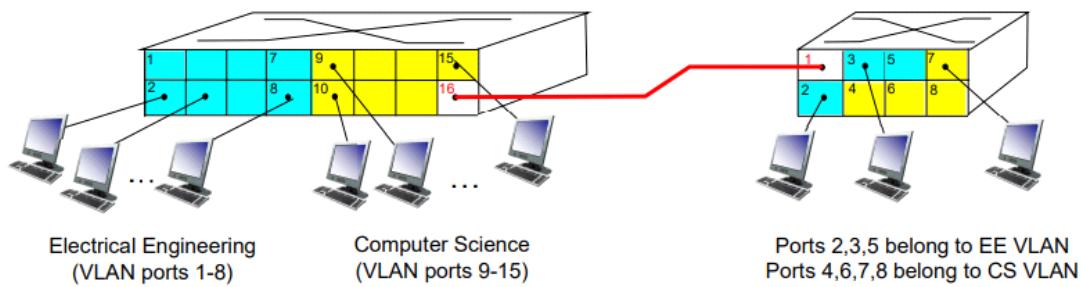
- Traffic isolation: i frame dalle porte da 1 a 8 possono solo raggiungere le porte da 1 a 8. si possono definire VLAN anche in base agli indirizzi MAC degli endpoint invece che le porte dello switch
- Dynamic membership: le porte possono essere assegnate dinamicamente tra le VLAN
- Forwarding between VLANs: fatto tramite routing (con switch separati)

VLAN e sottoreti



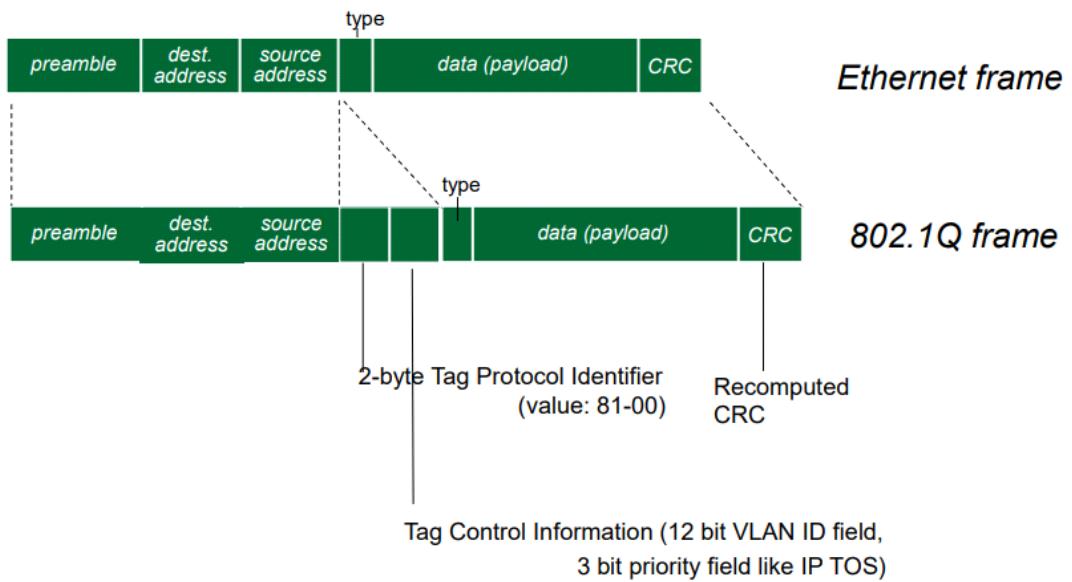
- PC1 and PC4 can communicate directly with each other
- but must use the router to get to PC2 and PC3.
- Frames issued on red VLAN 1 will not be seen by nodes on blue VLAN 2

VLANs su più switch



Trunk port: porta i frame tra le VLAN definite su multipli switch fisici. I frame. I frame che passano intra VLAN attraverso gli switch non usano ethernet 802.1 puro. Il protocollo 802.1q aggiunge/rimuove campi dell'header aggiuntivi per frame che passano attraverso le trunk port.

802.1Q VLAN frame format



APPLICAZIONI P2P

mercoledì 26 aprile 2023 17:10

Paradigma peer-to-peer

Tutti gli host sono peer e agiscono sia da client che da server.

Applicazioni:

- Distribuzione e memorizzazione di contenuti
- Condivisione di risorse di calcolo (elaborazione distribuita)
- Collaborazione e comunicazione
- Telefonia
- Content delivery network
- Piattaforme
- Bitcoin

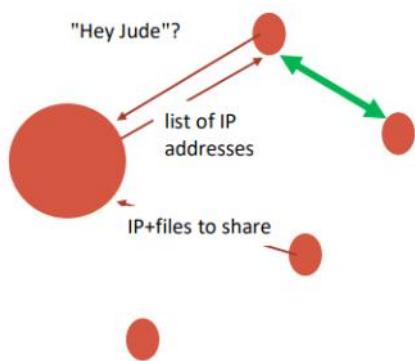
Peer

- Tutti i nodi (peer) hanno la stessa importanza. Sono nodi indipendenti localizzati ai bordi (edge) di Internet
- Nessun controllo centralizzato: ogni peer ha funzione di client e di server e condivide delle risorse
- Servent = server + client
 - o In realtà possono essere presenti dei server centralizzati o nodi con funzionalità diverse
- Sistemi altamente distribuiti
- Nodi altamente dinamici e autonomi: un nodo può entrare o uscire dalla rete P2P in ogni momento
- Operazioni di ingresso/uscita dalla rete anche sofisticate

Coppie di peer comunicano direttamente tra loro. Ogni peer però può non essere sempre attivo oppure cambiare indirizzo ogni volta che si connette

Directory centralizzata

Una directory -> server (si usa client- server). Con elenco di peer e risorse condivise



Problemi: single point of failure, bottleneck per performance, facile da "oscurare"

Reti decentralizzate

Non c'è un servizio di directory centralizzato.

I peer si organizzano in una *overlay network*

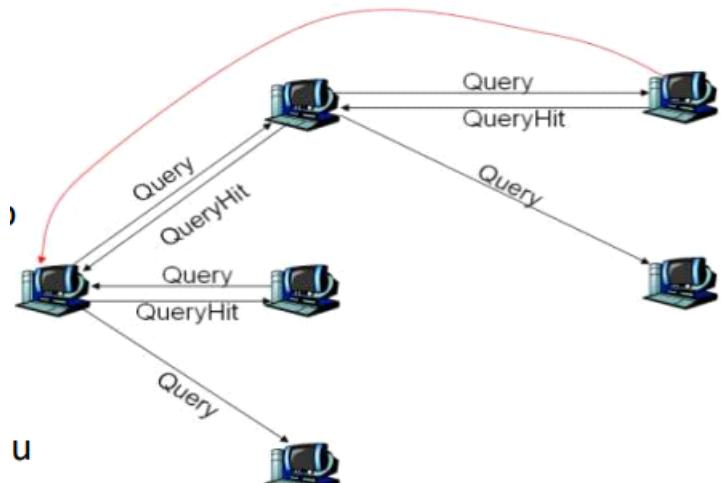
- Reti non strutturate
- Reti strutturate

Reti non strutturate

- Nodi organizzati come un grafo in modo random
- Non ci sono vincoli sul posizionamento delle risorse rispetto alla topologia del grafo.

- La localizzazione delle risorse è resa difficoltosa dalla mancanza di organizzazione della rete
- l'aggiunta di nodi è un'operazione semplice e poco costosa
- Obiettivo: gestire nodi con comportamento fortemente transiente (molte join/leave)

Query flooding



Completamente distribuita

Esempio: Gnutella:

- Peer formano overlay network
 - Archi = connessioni TCP
 - Ogni nodo tipicamente connesso a 10 vicini
- Peer invia query a tutti i suoi vicini
- Se peer riceve query e ha il file richiesto invia messaggio *QueryHit* su reverse path, altrimenti inoltra la query a tutti i suoi vicini

Problemi: flooding, poco scalabile.

Bootstrap: il sw include un primo elenco di nodi

Copertura gerarchica

- Non server con tutti i contenuti (ci sono bootstrap servers)
- I peer non sono tutti uguali (esistono i group leader)

I peer:

Ogni peer è

- Group leader se "potente" in banda o risorse
- Oppure viene assegnato a un group leader

Connessioni TCP tra peer e il group leader e tra alcune coppie di group leader

Il peer group leader tiene traccia del contenuto dei suoi "figli"

I file:

Ogni file è associato a un hash e un descrittore

Client invia una query di keyword al suo group leader

Il group leader risponde con dei "match" del tipo *<hash del file, indirizzo IP>*

Se il leader inoltra la query ad altri leader, questi rispondono con altri match.

Il client sceglie quindi i file da scaricare

BitTorrent

Idea di base: dividere un file in pezzi (da 256 Kb) e far ridistribuire ad ogni peer i dati ricevuti, fornendoli a nuovi destinatari.

Facendo ciò si riduce il carico a ogni sorgente, si riduce la dipendenza dal distributore originale e si fornisce ridondanza.

Condivisione file: torrent.

Insieme di peer (swarm) partecipano alla distribuzione di un certo file, scambiandosi parti (chunk) di file

Ogni peer allo stesso tempo preleva e trasmette più chunk

Tracker: nodo che coordina la distribuzione del file.

Per condividere un file, un peer crea un file .torrent che contiene metadati sul file condiviso e sul tracker.

Un nuovo peer cerca (con motore di ricerca) il torrent e ottiene metafile con info sui chunk e indirizzo IP di tracker. Contatta tracker e riceve indirizzi di alcuni peer dello swarm

Strategie principali:

- "tit for tat" (pan per focaccia): inviare dati ai peer che inviano dati, scegliendo quelli che stanno inviando a frequenza maggiore
- Ciascun peer scambia dati con al più 4 vicini (priorità a chi invia a velocità più alta)
- Ogni peer classifica i suoi vicini in choked e interested
 - o Ogni 10 sec aggiorna la classifica
 - o Ogni 30 sec sceglie un choked a caso e lo promuove
- Rarest chunks first

Reti strutturate

- Vincoli sul grafo e sul posizionamento delle risorse sui nodi del grafo
- l'organizzazione della rete segue principi drigidi
- l'aggiunta o la rimozione di nodi è un'operazione costosa
- Obiettivo: migliorare la localizzazione delle risorse

Reti strutturate - DHT

- Sistemi con Distributed Hash Table (DHT)
- Ad ogni peer è assegnato un ID ed conosce un certo numero di peer
- Ad ogni risorsa condivisa (pubblicata) viene assegnato un ID, basato su una funzione hash applicata al contenuto della risorsa ed al suo nome
- Routing della risorsa pubblicata verso il peer che ha l'ID più simile a quello della risorsa
- La richiesta per la risorsa specifica sarà instradata verso il peer che ha l'ID più simile a quello della risorsa

SICUREZZA

mercoledì 26 aprile 2023 18:13

Segretezza -> cifratura

Integrità (non essere modificato il messaggio) -> funzione hash

Autenticazione: MAC (hash su messaggio + chiave), firma digitale

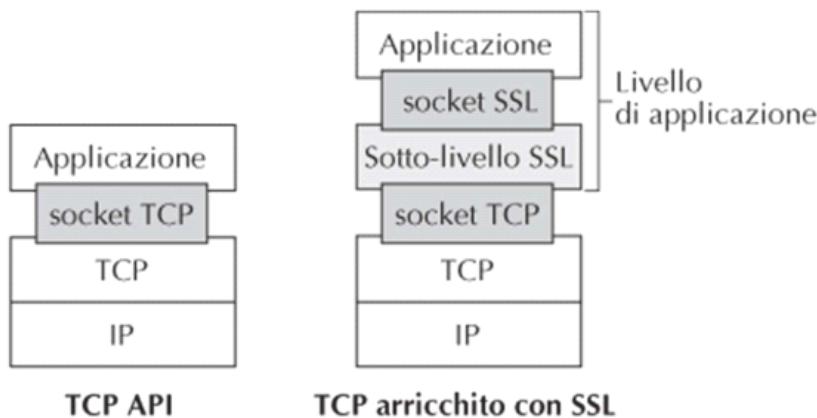
SSL/TLS

Sicurezza del livello di trasporto. Protocollo di sicurezza ampiamente diffuso

Standard: TLS

Fornisce:

- Riservatezza
- Integrità
- Autenticazione



SSL fornisce delle API alle applicazioni. Librerie/classi SSL C e Java

SSL: segreto condiviso. Si scambia inf. ase di handshake per generare la chiave

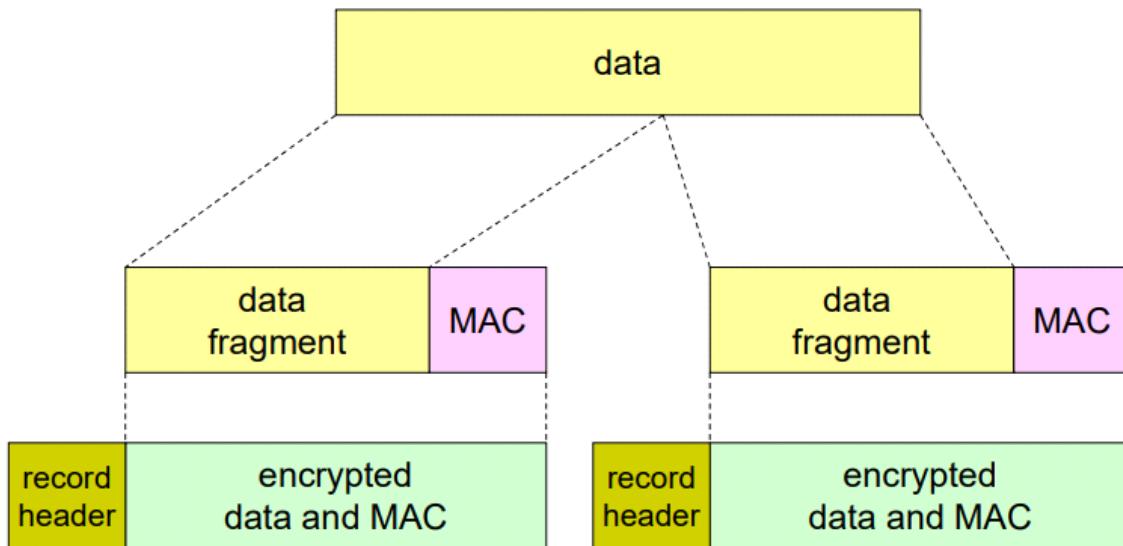
SSL cipher suite:

cipher suite

- public-key algorithm
- symmetric encryption algorithm
- MAC algorithm

Supporta varie cipher suite su cui il client e server di mettono d'accordo

Record



- SSL suddivide il flusso di dati in record
- Aggiunge un MAC
- Cifra record + MAC
- record header: content type; version; length

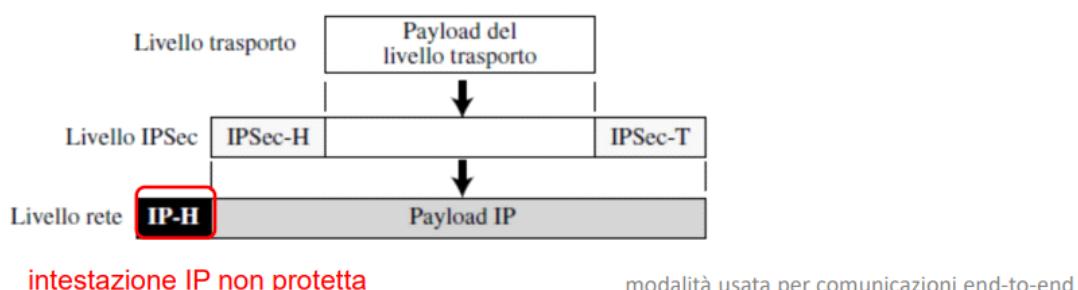
IPSec

Protocolli per fornire sicurezza a livello di rete

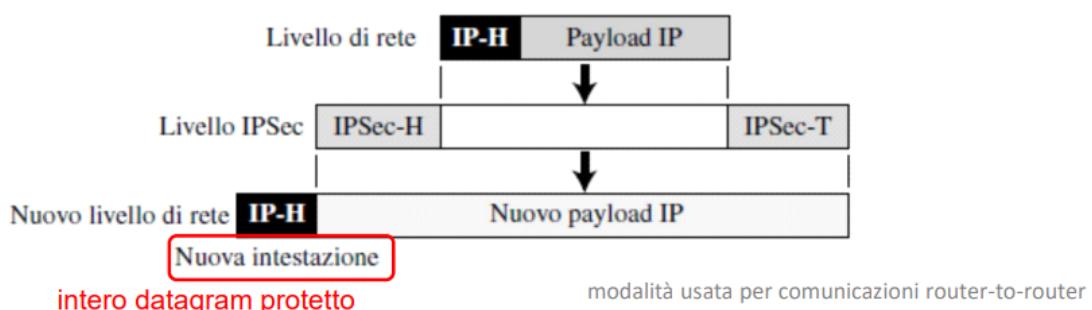
Autenticazione e confidenzialità dei pacchetti IP

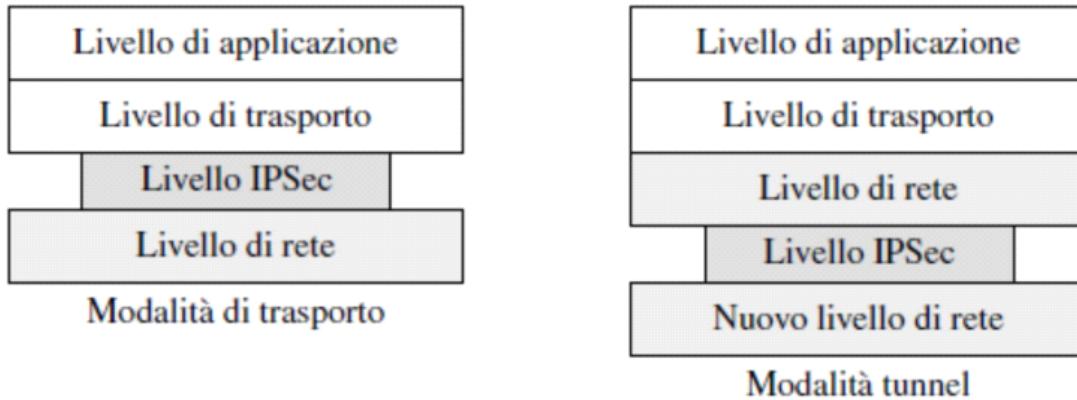
- Modalità trasporto
- Modalità tunnel

Modalità **trasporto**: protegge dati passati da liv. trasporto a liv. rete



Modalità **tunnel**: protegge l'intero pacchetto IP

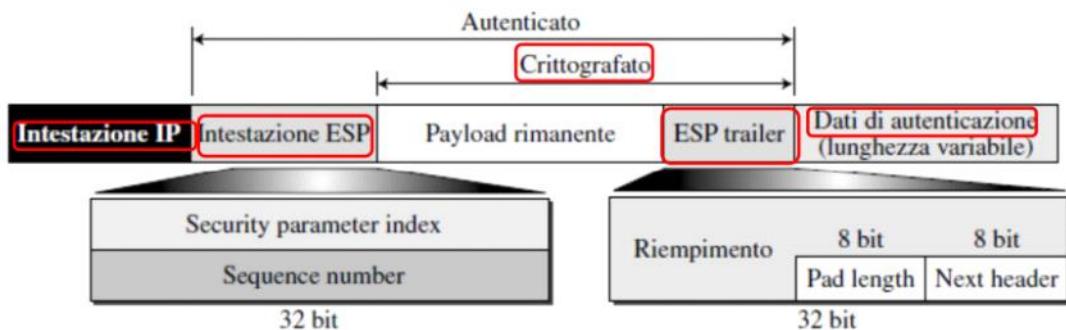




IPsec: protocollo ESP (Encapsulating Security Payload)

Obiettivi: Autenticazione sorgente, Integrità, Riservatezza

1. Viene aggiunto trailer ESP
2. Payload e trailer ESP vengono crittografati
3. Viene aggiunta intestazione ESP
4. Intestazione ESP, payload e trailer ESP vengono usati per generare dati autenticazione (che vengono aggiunti dopo trailer ESP)
5. Viene aggiunta intestazione IP (impostando campo protocollo a 50)



VPN: rete privata sulla internet pubblica