



Reti di Calcolatori

Reti di Calcolatori - 2010/2011

Questo riassunto e' stato realizzato in preparazione dello scritto e del successivo orale di reti. Le informazioni sono state ricavate dal libro, appunti e web e da me successivamente rielaborate; perciò troverete sicuramente errori di ortografia, sfondoni in campo informatico e altri tipi di distrazioni, quindi vi consiglio un lettura coscienziosa.

Bona...



Capitolo 1 Internet e reti d calcolatori 7

Cosa e' una rete	7
<i>Topologia di rete</i>	7
<i>Modalità' di Trasmissione</i>	7
<i>Mezzi di trasmissione</i>	8
<i>Reti locali LAN</i>	8
<i>Reti Metropolitane MAN</i>	8
<i>Reti geografiche WAN</i>	9

Che cosa e' Internet	9
-----------------------------	----------

<i>Internet Hardware</i>	10
<i>Chi amministra la rete</i>	10
<i>Cos'e' un protocollo</i>	10
<i>Switching di circuito</i>	11
<i>Switching di pacchetto</i>	12

Perdite e ritardi in packet-switched network	12
-----------------------------------------------------	-----------

<i>Ritardi di Elaborazione</i>	12
<i>Ritardi in un Nodo</i>	13
<i>Perdita di Pacchetti</i>	13
<i>Ritardo end-to-end</i>	14

Struttura a livelli - OSI (Open System Interconnect) Reference Model	14
-----------------------------------------------------------------------------	-----------

<i>Modello IOS/OSI</i>	14
<i>Manca la parte del libro sui livelli che e' differente da questa</i>	16
<i>Data Encapsulation</i>	16

Il modello TCP/IP	16
--------------------------	-----------

<i>TCP/IP vs OSI</i>	17
----------------------	----

Network e End-User Devices	17
-----------------------------------	-----------

Capitolo 2 Livello di applicazione 19

I principi delle applicazioni di rete	19
----------------------------------------------	-----------

<i>Architetture delle applicazioni di rete</i>	19
------------------------------------------------	----

Processi Comunicanti	20
-----------------------------	-----------

Web e HTTP	22
-------------------	-----------

<i>Terminologia</i>	22
<i>Web (HTTP) Server</i>	22
<i>URL</i>	22
<i>Panoramica su HTTP</i>	24



<i>Connessioni HTTP</i>	24
<i>Analisi temporale delle connessioni non persistenti</i>	25
<i>Analisi temporale delle connessioni persistenti</i>	25
<i>Analisi temporale delle connessioni persistenti senza pipeline</i>	25
<i>Analisi temporale delle connessioni persistenti con pipeline</i>	26
<i>Messaggi HTTP</i>	26
<i>Interazione utente-server: i Cookie</i>	28
<i>Caching Web (HTTP Proxy Server)</i>	29
<i>Get Condizionale</i>	30
Trasferimento di file: FTP	31
<i>Comandi e risposte FTP</i>	33
<i>Perché Out-of-Band di controllo?</i>	33
<i>Approfondimento funzionamento generale</i>	33
<i>Problemi relativi alla sicurezza</i>	34
Posta Elettronica	34
<i>SMTP</i>	35
<i>Protocolli di accesso alla posta</i>	36
<i>POP3 [RFC 1939]</i>	37
<i>IMAP [RFC 3501]</i>	37
Domain Name System (DNS)	38
<i>Caratteristiche Principali</i>	39
<i>DNS in dettaglio</i>	39
<i>Approfondimenti</i>	42
<i>Domande e Problemi</i>	43
Domande	43
Problemi	44
Capitolo 3 Livello di trasporto	47
Parametri dal livello applicazione al livello di trasporto	47
Servizi e Protocolli di Trasporto	47
<i>Relazione tra livello di trasporto e livello di rete</i>	47
<i>Protocollo del livello di trasporto in Internet</i>	47
Multiplexing e demultiplexing	47
Trasporto senza connessione: UDP	48
<i>Si usa UDP perché:</i>	48
<i>Error control (Checksum UDP)</i>	49

Principi del trasferimento Affidabile	50
Trasferimento dati affidabile	50
Rdt 1.0: trasferimento affidabile su canale affidabile	51
Rdt 2.0: canale con errori nei bit	51
Rdt 2.1	52
Rdt 2.2: un protocollo senza NAK	53
Rdt 3.0: canali con errori e perdite	53
Rdt 3.0: con funzionamento stop-and-wait	54
Protocolli con pipeline	54
Go-Back-N	55
Ripetizione Selettiva	57
Trasporto orientato alla connessione: TCP	60
Connessione TCP	61
Struttura dei segmenti TCP	61
Ritrasmissione in TCP	63
Stima del tempo di andata e ritorno	64
Trasferimento dati affidabile	65
Ritrasmissione Rapida	66
Controllo di Flusso	67
Gestione delle connessioni TCP	68
Controllo di congestione TCP	70
Domande e Problemi	73
Domande e Problemi	74
Capitolo 4 Livello Network	75
Introduzione	75
Inoltro e instradamento	75
Modelli di servizi di rete	75
Reti a datagramma	76
IP Service Model: Perchè Best-Effort?	77
Protocollo Internet (IP)	78
Formato dei datagrammi	78
Frammentazione dei datagrammi IP	81
Indirizzamento IPv4	82
UPnP	87
ICMP	87
IPv6	88
Algoritmi di instradamento	89



<i>Instrandamento in Internet</i>	94
<i>Instrandamento Broadcast e Multicast</i>	99
Domande e Problemi	100
Capitolo 5 Livello DataLink	102
Livello di link: introduzione e servizi	102
<i>Servizi offerti dal livello DataLink</i>	102
<i>Chi implementa il DataLink Layer</i>	103
Tecniche di rilevazione e correzione degli errori	104
<i>Controllo di Parità</i>	104
<i>Somma di controllo</i>	105
<i>Controllo a Ridondanza Ciclica</i>	105
Protocolli di accesso multiplo	106
<i>I protocolli al accesso multiplo</i>	106
<i>Protocollo a suddivisione del canale (Channel partitioning)</i>	106
<i>Protocollo ad accesso casuale (Random access)</i>	107
<i>Protocolli con rilevamento delle portate</i>	108
<i>Protocollo a rotazione. (tak-in-turn)</i>	109
<i>Tecnologie LAN</i>	110
Indirizzi a Livello di Link	111
<i>Indirizzi MAC e ARP</i>	111
<i>Protocollo per la risoluzione degli indirizzi (ARP)</i>	111
<i>Invio verso un nodo esterno alla sottorete</i>	113
Ethernet	113
<i>Tipologia a stella</i>	113
<i>Struttura dei pacchetti Ethernet</i>	113
<i>Servizio senza connessione non affidabile</i>	114
<i>Ethernet utilizza il protocollo di accesso multiplo CSMA/CD</i>	114
<i>Efficienza di Ethernet</i>	115
<i>La probabilità di eventuali collisioni viene così calcolata:</i>	115
Sicurezza nelle Reti	116
Definizione	116
Le principali minacce nello scambio messaggi	116
<i>Sicurezza dei protocolli di rete</i>	116
Minacce alla sicurezza in Internet	117
<i>Packet sniffing (to sniff = odorare)</i>	117



Reti di Calcolatori

<i>IP Spoofing</i>	117
<i>Denial of Service (DoS)</i>	117
Strumenti Utilizzati	118
Sniffing su reti	118
Crittografia	119
<i>Algoritmi di crittografia</i>	119
La cifratura a chiave asimmetrica	119
<i>Crittografia a chiave Pubblica</i>	120
<i>Chiavi di sessione</i>	123
Integrità dei messaggi	123
<i>Funzioni Hash Crittografiche</i>	123
<i>Codice di autenticazione dei messaggi</i>	124
<i>Firme Digitali</i>	124
<i>Certificazione della chiave pubblica</i>	125
Autenticazione end-to-end	125
<i>ap 3.1</i>	125
<i>ap 4.0</i>	125
<i>ap 5.0</i>	126
E-mail sicure	126
<i>Rendere sicure le connessioni TCP: SSL</i>	127

Capitolo 1 Internet e reti di calcolatori

Cosa e' una rete

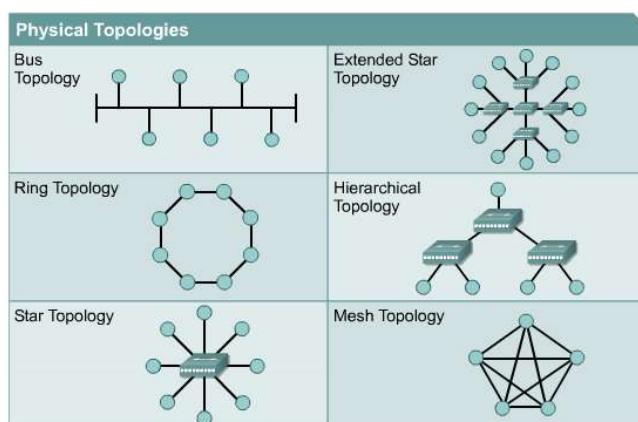
Un insieme **interconnesso** di computer **autonomi** che permette l'accesso alle informazioni, la condivisione di risorse e la facilitazione delle comunicazioni.

Fatta di hardware come apparati di interconnessione, apparati per il controllo della trasmissione ecc..

Fatta di software per codifica e formattazione dei dati, rilevazione degli errori e correzione, instradamento delle informazioni fra i computer in rete e servizi e applicazioni.

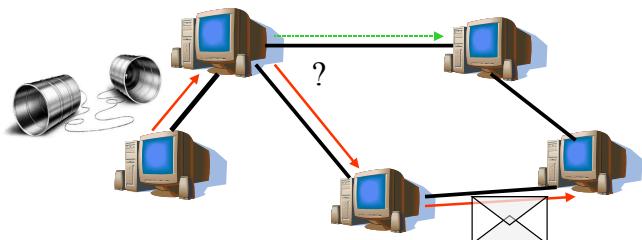
Topologia di rete

Definisce la struttura della rete:



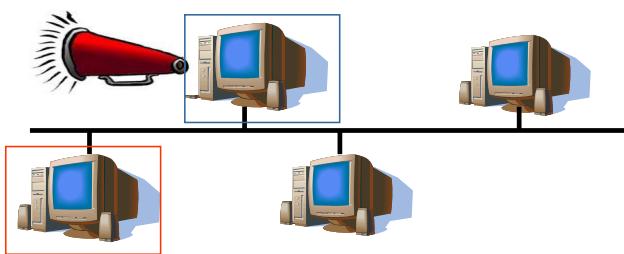
- La topologia fisica, che e' il reale layout dei fili e/o dei media
- La topologia logica, che definisce come i media vengono usati dagli hosts per spedire i dati.

Modalità' di Trasmissione



Point-to-point

- Le connessioni sono fra coppie di nodi
- Per passare dal mittente al destinatario il messaggio deve attraversare nodi intermedi
- In genere ci possono esser più cammini e deve esser scelto il migliore



Broadcast

- Il canale di comunicazione e' condiviso fra tutte le macchine
- I messaggi inviati sono ricevuti da tutte le macchine
- Nel pacchetto e' indicato l'indirizzo del destinatario
- Solo la macchina destinataria accetta il pacchetto trasmesso

Mezzi di trasmissione

Si dividono in due grandi categorie:

Distanza fra le unità di elaborazione	Ambiente
0.1m	Circuito
1m	Sistema
10m	Stanza
100m	Edificio
1Km	Campus
10Km	Città
100Km	Nazione
1000Km	Continente
10000Km	Pianeta

Calcolatori multiprocessore
 Rete Locale (LAN)
 Rete Metropolitana (MAN)
 Rete Geografica (WAN)
 Internet

1. Wired

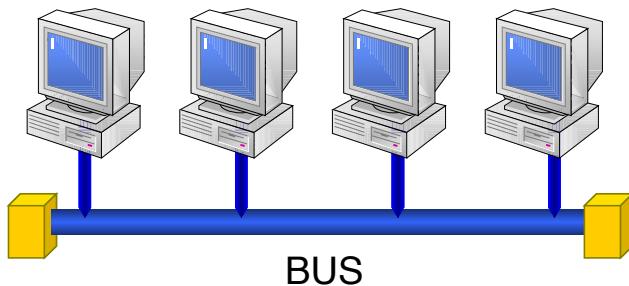
- 1.1. Rame, utilizzato per i doppini e i cavi coassiali

- 1.2. Vetro, utilizzato per le Fibre ottiche

2. Wireless

- 2.1. Suoni, luce, raggi infrarossi, radiofrequenza e microonde

Reti locali LAN



Rete di comprensorio, senza attraversamento di suolo pubblico.

Caratteristiche:

- Velocità trasmissiva molto elevata (> 10 Mb/s)
- Distanze ridotte (< km)
- Mezzi trasmissivi anche non conformi alle raccomandazioni CCITT
- Realizzata e gestita dal proprietario del comprensorio.

Reti Metropolitane MAN

Rete in ambito urbano con disponibilità di canali trasmissivi veloci: sono grosse LAN

Caratteristiche :

- Velocità trasmissiva v elevata ($2 \text{ Mb/s} < v < 140 \text{ Mb/s}$)
- Limitata ad una città o a poche città vicine.
- Mezzi trasmissivi conformi agli standard CCITT, generalmente in fibra ottica.
- Realizzata e gestita dalle società telefoniche e trasmissione dati pubblica

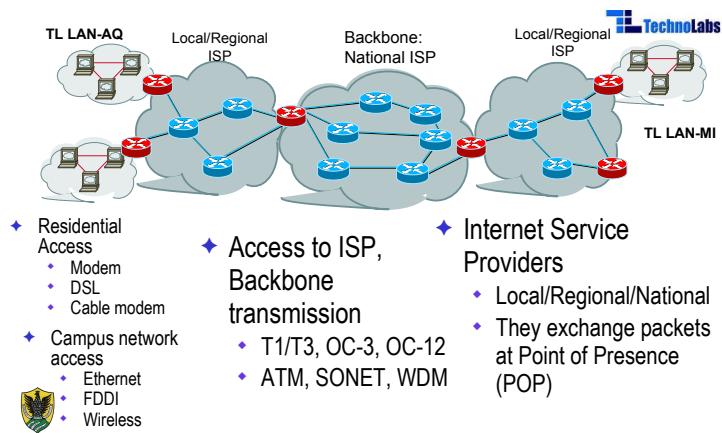
Reti geografiche WAN

Realizzate in ambito nazionale o internazionale, utilizza qualsiasi mezzo trasmissivo, e interconnettono LAN/MAN

Caratteristiche:

- Velocità trasmissiva **v** medio-bassa ($9.6 \text{ Mb/s} < v < 34 \text{ Mb/s}$)
- Distanze elevate (alti costi e basse velocità)
- Mezzi trasmissivi conformi agli standard CCITT.

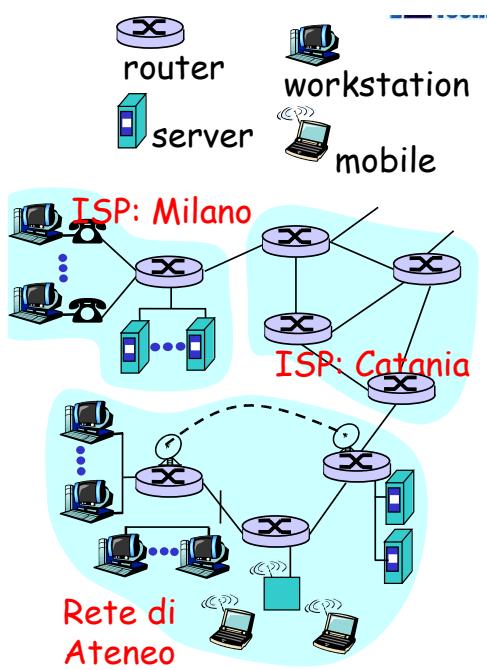
Che cosa e' Internet



Sistema di informazione globale che

- è logicamente interconnesso da un Address space unico e globale, basato sull'Internet Protocol (IP);
- è in grado di supportare la comunicazione tramite la suite Transmission Control Protocol/Internet Protocol (TCP/IP) o le sue successive estensioni/sviluppi, e/o altri protocolli compatibili con l'IP; e
- fornisce, utilizza o rende accessibili, sia pubblicamente che privatamente, servizi di comunicazione di alto livello stratificati e basati sulla correlata infrastruttura qui descritta.

Internet Hardware



Milioni di dispositivi computazionali connessi in rete: **host, end-system, PC, workstation, server, PDA, cellulari ecc.**

Collegamenti con Fibra ottiche, Point radio, satelliti.

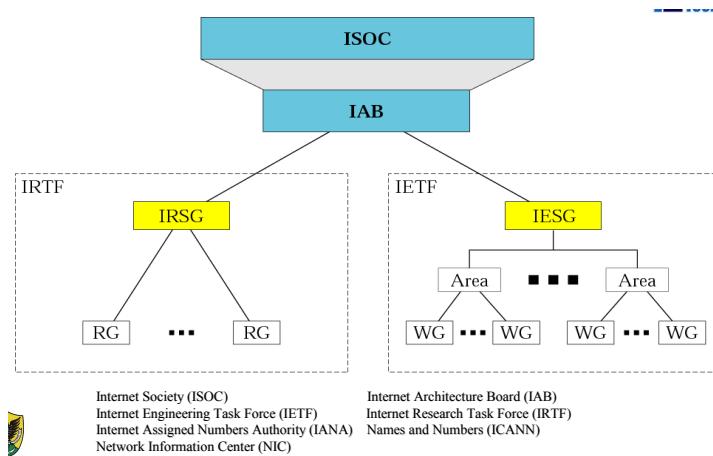
Router: compito di inoltrare i pacchetti lungo la rete.

Protocolli di comunicazione: meccanismi per la trasmissione dei messaggi (TCP, IP, UDP, HTTP, FTP, PPP...)

Internet “network of network” con una struttura gerarchica.

End-System, sono programmi e applicazioni di rete connessi con LINK.

Chi amministra la rete

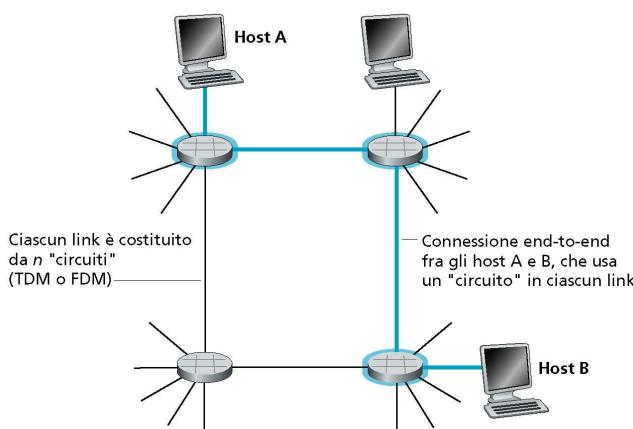
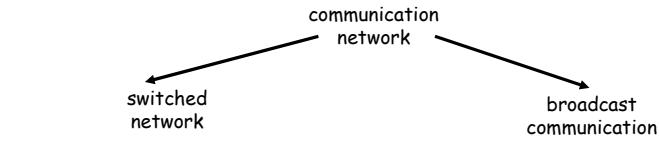


Cos'e' un protocollo

I protocolli sono un insieme formalizzato di regole che definiscono il **formato** e l'**ordine**, dei messaggi inviati e ricevuti tra entità della rete e le azioni che vengono fatte per la trasmissione e ricezione dei messaggi stessi

La rigidità dei protocolli permette di fare attacchi (DoS).

Esistono due approcci fondamentali per spostare dati nella rete di collegamenti e switch:



Legenda:



Reti Broadcast, dove i nodi condividono un canale di comunicazione; l'informazione trasmessa da un nodo e' ricevuta da tutti gli altri nodi della rete, esempi ne sono la tv e la radio

Rete Switched, l'informazione e' trasmessa a un piccolo sub-set dei suddetti nodi. Esse a sua volta si suddividono in due categorie: *Switching di circuito* e *Switching di pacchetto*.

Switching di circuito

Un circuito dedicato per ogni chiamata. Ovvero le risorse sono riservate End-to-End alla chiamata.

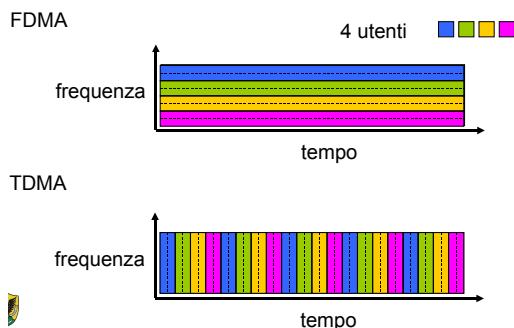
Prestazioni garantite

E' richiesta la fase di "Call Setup/handshake" dove vengono allocate le risorse, le quali rimangono inattive (idle) se non usate dal chiamante (no sharing).

Caratteristiche: Banda di trasmissione (Quando due host desiderano comunicare, la rete stabilisce una connessione end-to-end dedicata), Risorse dedicate, Performance elevata, necessita di una fase di handshake.

Ciascuno dei commutatori di circuito dispone di n circuiti, in modo da poter supportare n connessioni simultaneamente; questa caratteristica pero' limita la connessione, infatti ogni connessione ottiene solo $1/n$ della larghezza della banda per la durata della stessa.

Le risorse sono fatte a pezzi, cioè viene diviso o la *Frequenza* (FDM - Frequency Division Multiplexing) o il *Tempo* (TDM - Time Division Multiplexing).



Con FDM, lo spettro di frequenza di un collegamento viene suddiviso tra le connessioni stabilite tramite il collegamento. La larghezza di banda viene detta **ampiezza di banda (bandwidth)**.

Con TDM, il tempo viene suddiviso in **frame** di durata fissa che sono a loro volta ripartiti in un numero fisso di **slot** temporali. Quando la rete stabilisce una connessione attraverso un collegamento, le dedica uno slot di tempo in ogni frame. E questi slot sono dedicati unicamente a quella connessione.

Problematiche: i circuiti dedicati sono inattivi durante i periodi di silenzio (**silent period**).

Switching di pacchetto

Ogni flusso dati end-to-end e' **diviso in pacchetti**, tra origine e destinazione i pacchetti viaggiano attraverso collegamenti e commutatori di pacchetto. Ogni nodo riceve l'intero pacchetto, lo memorizza per un breve tempo e poi lo spedisce al prossimo nodo (**Store-and-Forward Networks**). Tale meccanismo introduce il **ritardo di Store-and-forward** in ingresso di ciascuno collegamento.

Supponendo che esistano Q collegamenti tra i due host e con frequenza pari a R bps e che il primo voglia inviare L bit, il ritardo totale = QL/R .

Ogni commutatore di pacchetto connette più collegamenti, e per ciascuno di questi esso mantiene un **buffer di output**. Ciò introduce i **ritardi di coda**.

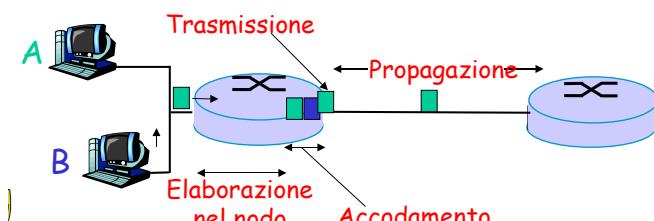
Inoltre nell'eventualità che il buffer si riempia, si verificheranno delle **perdite di pacchetti (packet loss)**.

Ogni pacchetto e' inviato attraverso la rete da nodo a nodo seguendo dei cammini (routing).

Dati sono suddivisi in packets	Problematiche
Packet degli utenti A e B condividono le risorse di rete	Richiesta di risorse può essere superiore della disponibilità
ogni packet utilizza la banda al massimo della sua capacità	congestione: code dei pacchetti/buffer
"resources used as <i>needed</i> ". Non è necessaria una allocazione iniziale di tutte le risorse	"store and forward": packet fanno un passo alla volta <ul style="list-style-type: none"> • Trasmissione su un link • Attesa e trasmissione al link successivo

Manca ISP

Perdite e ritardi in packet-switched network



Si hanno 4 sorgenti di ritardo per ogni hop. E i ritardi non sono prescindibili dato che non conosciamo in partenza il viaggio che fa il pacchetto.

Processamento al nodo: Correzione di errore sui bit e calcolo del link di uscita

Attesa in coda: Attesa per la trasmissione , la quale dipende dalla congestione del router.

I tipi di ritardi sono:

Ritardi di Elaborazione

- tempo richiesto per esaminare l'intestazione del pacchetto e per determinare dove dirigerlo = **delab** (~ microsec).

Ritardo di Accodamento

- tempo che deve attendere un pacchetto che deve aspettare nel buffer prima di esser inviato.
- La lunghezza di questo ritardo dipende dal numero di pacchetti precedentemente arrivati che non sono stati ancora trasmessi. Se la coda e' vuota il ritardo e' nullo (~ microsec, ~millisec).

Ritardo di Trasmissione

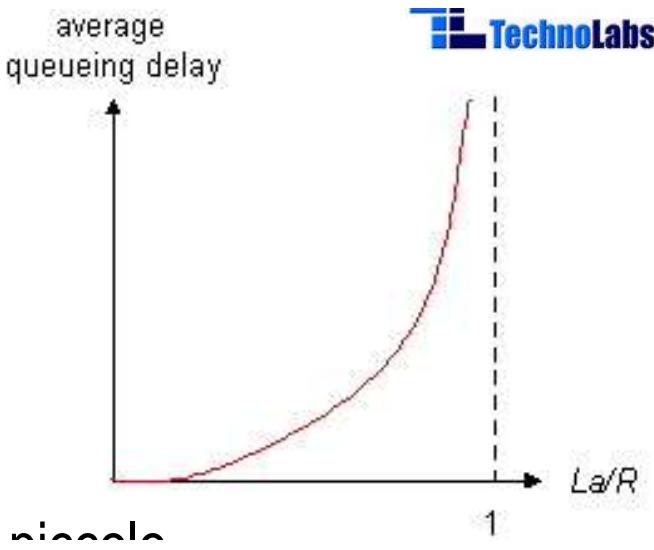
- tempo per immettere tutti i bit sul canale di propagazione. Viene usata la politica FIFO.
- R= frequenza trasmittiva del collegamento/banda sul link (bps)
- L=lunghezza pacchetto (bit)

- Tempo per trasmettere pacchetto sul link = $d_{trans} = L/R$

Ritardo di Propagazione

- tempo che impiega un pacchetto per passare da un router ad un altro, *dipende dal mezzo fisico*. (~ millisec).
- d = lunghezza link fisico
- s = velocità di propagazione nel mezzo (~ 2×10^8 m/sec)
- Ritardo di propagazione nel mezzo = $d_{prop} = d/s$

Intensità del traffico



• R=banda del link (bps)

• L=lunghezza del pacchetto (bit)

• a=frequenza media (rate) di arrivo dei pacchetti nella coda (packets/sec)

Intensità del traffico = La/R

• $La/R \sim 0$: ritardo medio di coda piccolo

• $La/R = 1$: ritardo medio di coda grande

• $La/R > 1$: arrivano più pacchetti di quanti possano essere smaltiti, il tempo di attesa in coda tende a diventare infinito! (in realtà perdita)

L'intensità del traffico influisce sul **dqueue**. Se i pacchetti arrivano a cadenza periodica, nessuno avrà ritardi, se arrivano a blocchi ma periodicamente si possono avere dei significativi ritardi di coda.

Il ritardo di coda può variare da pacchetto a pacchetto, pertanto per caratterizzare il ritardo di accodamento si fa uso solamente di misure statistiche quali: **Ritardo di accodamento medio, varianza del ritardo di accodamento, probabilità che dqueue > soglia**.

Ritardi in un Nodo

$$dnodo = delab + dqueue + dtrans + dprop$$

- delab = processing delay / elaborazione
 - tipicamente qualche microsecondo o meno
- dqueue = queuing delay
 - dipende dalla congestione
- dtrans = transmission delay
 - = L/R , significativo per links a bassa velocità
- dprop = propagation delay
 - Da qualche microsecondo fino a decine di msec

Perdita di Pacchetti

- Code: ai link di uscita sono associati uno o più buffer che hanno capacità finita, quando un pacchetto arriva ad una coda piena, il pacchetto viene buttato (lost)
- I pacchetti persi possono essere ritrasmessi dal nodo precedente, dalla sorgente (end system), o non essere ritrasmessi affatto

Ritardo end-to-end

Ipotizzando l'esistenza di $N-1$ router tra l'host origine e quello di destinazione e che la rete non sia congestionata, il ritardo di elaborazione a ciascun router e presso l'origine sia d_{lab} , la frequenza di trasmissione in uscita a ogni router e all'origine sia di R bps e la propagazione su ciascun collegamento sia d_{prop} .

Tutti i ritardi si accumulano e danno come ritardo end-to-end:

$$\text{end-to-end} = N(d_{lab} + d_{trans} + d_{prop})$$

Struttura a livelli - OSI (Open System Interconnect) Reference Model

Ogni livello implementa un servizio mediante funzionalità interne e usando i servizi messi a disposizione dagli strati inferiori. *Stratificazione*.

La stratificazione permette una più facile organizzazione e individuazione delle funzionalità, la modularità facilita la manutenzione e la modifica dei sistemi.

La modifica dell'implementazione dei servizi resi da uno strato è trasparente (purchè non si modifichi l'interfaccia); Es., cambiare il corriere non altera il funzionamento complessivo del servizio postale (ogni livello espone le interfacce ma nasconde la sua implementazione).

Principi base:

- **Separation of Concern:** Separazione degli interessi e delle responsabilità, fare ciò che compete, delegando ad altri tutto ciò che è delegabile.
- **Information Hiding:** Nascondere tutte le informazioni che non sono indispensabili a che il committente possa compiutamente definire l'operazione.

Modello IOS/OSI



In tale modello sono previsti 7 livelli: 3 Livelli di rete (fisico, link, rete) e 4 Livelli di utente (Trasporto, sessione, presentazione, applicazione).

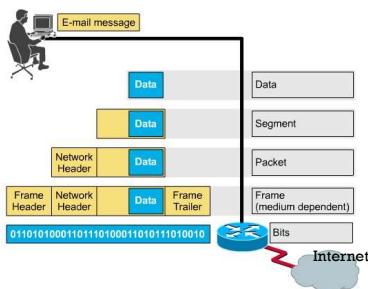
Livello Fisico	Livello Dati (Datalink)	Livello Rete
<p>Compito: trasmettere sequenze binarie (bit) sul canale trasmittivo</p> <p>Lo standard prescrive:</p> <ul style="list-style-type: none"> • le caratteristiche fisiche dell'interfaccia e del mezzo • modalità di trasmissione (HD, FD, S) la forma e la dimensione dei connettori • la sincronizzazione dei bit 	<p>Compito: creare trame (frame) e trasmetterle con "sufficiente" affidabilità tra due entità direttamente connesse, rilevare errori di trasmissione ed eventualmente correggerli</p> <ul style="list-style-type: none"> • smistamento • controllo flusso • controllo errore • controllo accesso <p>Rilevazione e correzione errori: codici autocorreggenti, ritrasmissione, etc.</p>	<p>Compito: gestire l'instradamento (routing) di frame attraverso sistemi intermedi, ed eventualmente trovare percorsi alternativi in caso di guasti</p> <p>Indirizzamento: logico</p> <p>Algoritmi di instradamento: definizione e/o apprendimento (completo o parziale) della topologia della rete, calcolo del percorso su base locale e/o globale, riconfigurazione in caso di guasti, etc.</p> <p>Non necessariamente garantisce: affidabilità della trasmissione delle frame, non duplicazione alla destinazione, rispetto alla destinazione del loro ordine di invio</p> <p>È responsabile della consegna mittente-destinatario di un pacchetto</p>
<p>Livello Trasporto</p> <p>Compito: trasferire l'informazione (il messaggio) end-to-end affidabilmente e trasparentemente, ottimizzando l'uso delle risorse</p> <p>Affidabilità: tutte le trame arrivano a destinazione, in copia unica e in ordine</p> <p>Trasparenza: "forma" dell'informazione qual era alla sorgente conservata a destinazione</p> <p>Ottimizzazione: traffico ripartito sui canali disponibili, prevenzione della congestione della rete</p> <p>Funzioni: segmentazione, riassemblaggio, controllo della connessione, controllo flusso, controllo errore</p>	<p>Livello Sessione</p> <p>Compito: gestire il dialogo end-to-end tra due programmi applicativi che debbono comunicare</p> <p>Dialogo: garantire la mutua esclusione nell'utilizzo di risorse condivise, intercalare domande e risposte garantendo la consequenzialità</p> <p>Sincronizzazione: stabilire punti intermedi nella comunicazione rispetto ai quali entrambe le parti abbiano la garanzia che quanto accaduto "prima" sia andato a buon fine</p>	<p>Livello Presentazione</p> <p>Compito: gestire la sintassi dell'informazione lungo l'intero percorso end-to-end, convertendo l'uno nell'altro i vari formati</p> <p>traslazione, crittografia, compressione</p>

Livello Applicazione	Benefici del Layering	
Compito: definire i servizi attraverso cui l'utente utilizza la rete, con tutte le relative interfacce di accesso	Vengono introdotti livelli intermedi che forniscono un'astrazione comune per le varie tecnologie.	
Servizi di utente: terminale virtuale, trasferimento di file, posta elettronica, servizi di directory, etc.		
Servizi di sistema operativo: risoluzione di nomi, localizzazione di risorse, sincronizzazione degli orologi tra sistemi diversi, controllo di diritti di accesso, etc.		

Manca la parte del libro sui livelli che è differente da questa

Data Encapsulation

Data Encapsulation è il processo di aggiungere un header ai dati che fluiscono giù per il modello OSI. Ogni livello OSI può aggiungere il proprio header ai dati ricevuti dal livello superiore (o dal programma software 'sopra' il livello Applicazione).



I 5 Steps di Data Encapsulation sono:

- I livelli Application, Presentation e Session creano DATA dall'input degli utenti.
- Il livello Transport converte i DATA in SEGMENTS
- Il livello Network converte i SEGMENTS in PACKETS (o datagrams)
- Il livello Data Link converte i PACKETS in FRAMES
- Il livello Physical converte i FRAMES in BITS.

Lato TX l'informazione fluisce dall'alto in basso ed ogni livello spezzetta l' informazione ricevuta dal livello superiore e aggiunge l'header.

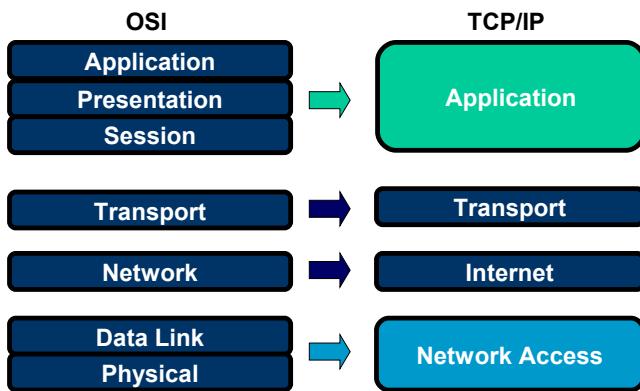
Lato RX l'informazione fluisce verso l'alto del modello strippando l'header ad ogni livello e riassemblando i pezzi di informazione.

Ogni livello contiene una **Protocol Data Unit (PDU)**, le quali sono usate per la conversazione tra peer-to-peer.

Il modello TCP/IP

Il modello TCP/IP è stato sviluppato dal Department of Defense (DoD) negli anni 60 come un progetto militare atto ad assicurare la comunicazione dei dati anche sotto attacco.

Da allora, TCP/IP è diventato lo standard de facto usato per la comunicazione dei dati su Internet.



TCP/IP vs OSI

- TCP/IP è una descrizione di protocolli già esistenti ed è quindi meno generale e flessibile
- I protocolli TCP/IP erano già largamente usati in ambito accademico quando fu proposto il modello OSI
- Il modello e i protocolli OSI hanno dei difetti
 - Il livello sessione è poco utile e quello presentazione poco significativo
 - I livelli rete e datalink sono molto complessi e richiedono una strutturazione in sottolivelli
 - Lo standard è complesso e i protocolli difficili da implementare
- TCP/IP non distingue bene fra servizi, interfacce e protocolli andando così a racchiudere il tutto nel livello applicazione.

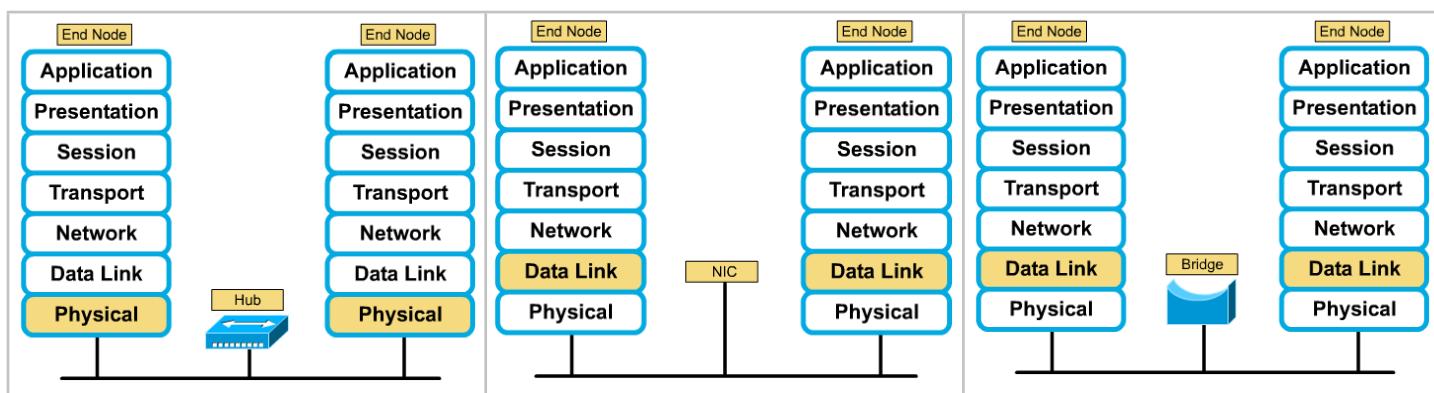
Network e End-User Devices

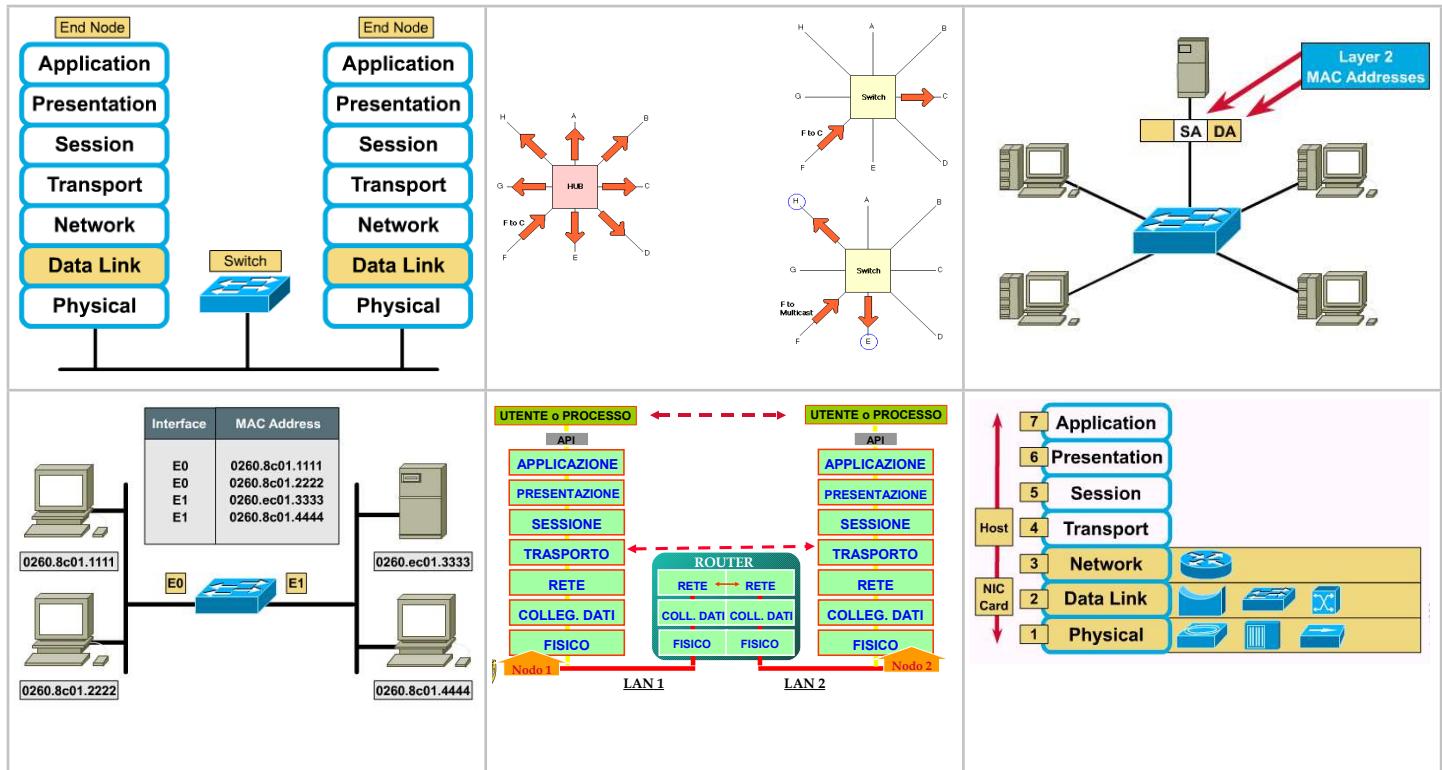
Network Devices	
Repeater	Bridge
Small Hub (10BASE-T)	Workgroup Switch
100BASE-T Hub	Router
Hub	Network Cloud

Equipments connessi direttamente ad un segmento di rete sono detti

Device. Essi a sua volta possono esser suddivisi in due classi:

- end-user devices.
- network devices.





Capitolo 2 Livello di applicazione

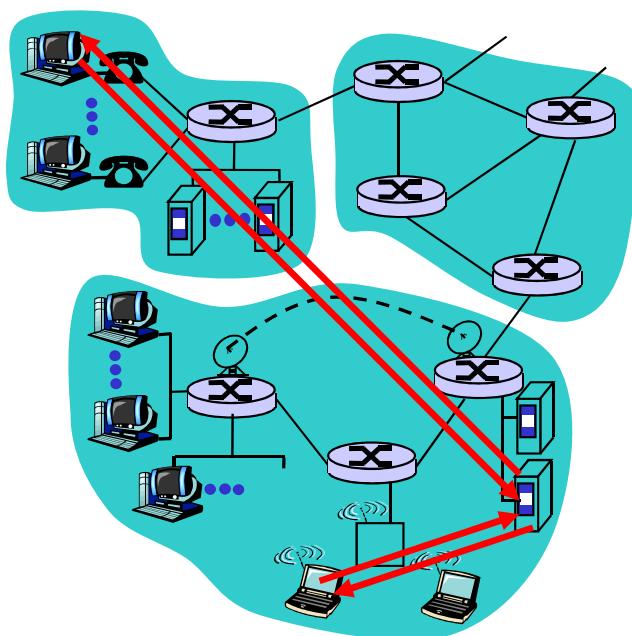
I principi delle applicazioni di rete

Il cuore dello sviluppo delle applicazioni di rete e' costituito dalla compilazione dei programmi che sono eseguiti dai sistemi terminali e comunicano tra loro via rete.

Architetture delle applicazioni di rete

Per lo sviluppatore l'architettura di rete e' fissata e fornisce alle applicazioni uno specifico insieme di servizi, ma il suo compito e' **progettare l'architettura dell'applicazione e stabilire la sua organizzazione sui vari end-system**.

Le principali architetture utilizzate sono:



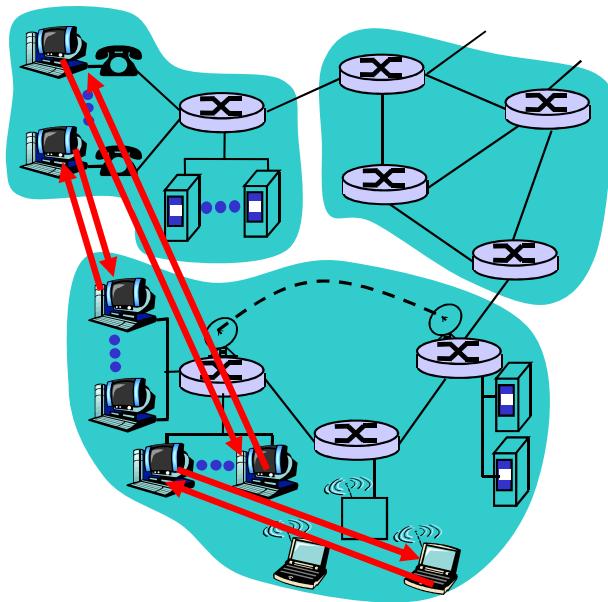
Architettura client-server: la quale prevede appunto:

Un server: un host sempre attivo (server) che risponde alle richieste di servizio di molti altri host (client). Il server dispone di un indirizzo IP fisso, diffusamente conosciuto.

Un client: i quali comunicano direttamente con il server e mai direttamente con altri client, può contattare il server in qualsiasi momento, Always-on o Sometimes-on, può avere indirizzi IP dinamici.

Spesso un solo server non e' in grado di soddisfare tutte le richieste per questo motivo in tali tipi di architetture spesso vengono usati dei cluster di host (server farm) per creare un potente server virtuale.

Gli applicativi basati su tale architettura fanno spesso un uso intensivo dell'infrastruttura.



Architettura P2P: la quale prevede che l'infrastruttura di server sempre attivi sia minima o del tutto assente, infatti essa sfrutta la comunicazione diretta tra coppie arbitrarie di host chiamati peer , collegati in modo intermittente e con IP dinamici. il loro punto di forza e' la scalabilita' e ciò determina che esse non necessitano di una significativa infrastruttura del server e banda nel server.

Ibridi (client - server e P2P): uno dei più famosi fu

Napster, il quale prevedeva lo scambio di messaggi secondo la logica P2P, ma la ricerca dei file e' centralizzata (logica client - server), i peer registrano il loro contenuto presso un server centrale e i peer chiedono allo stesso server di localizzare il contenuto.

Messaggistica Instantanea, la chat tra due utenti e' del tipo P2P, ma l'individuazione della presenza/location e' centralizzata mediante la registrazione dell'IP in fase di connessione.

Processi Comunicanti

I processi su due end-system comunicano scambiandosi **messaggi** attraverso la rete, processi client e server: il processo che inizia la comunicazione viene identificato come **client** mentre il processo che attende di esser contattato e' identificato come **server**.

Nell'applicazioni P2P gli host hanno processi client e processi server attivi contemporaneamente. Il peer che scarica e' il client, il peer che mette a disposizione il file e' il server.

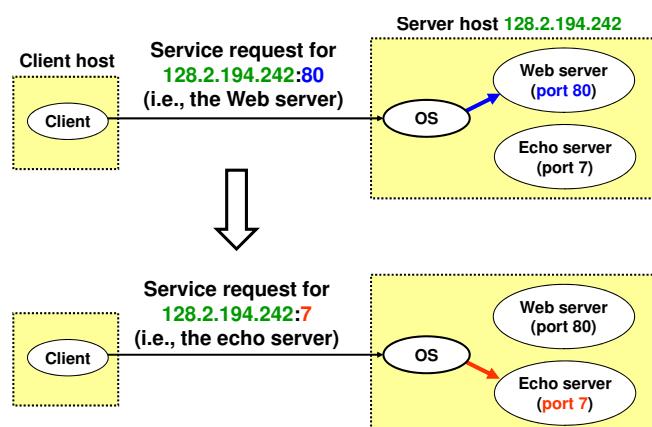
I socket

Ogni messaggio generato da un processo deve esser inoltrato al livello di rete sottostante, e lo fa' inviando/ricevendo messaggi a/da la sua **socket**.

Una socket e' analoga ad una porta e il processo presuppone l'esistenza di un'infrastruttura esterna che trasporterà il messaggio attraverso la rete fino a destinazione.

Le socket sono **API(application programming interface)** tra l'applicazioni e la rete, esse gli mettono ad disposizione la possibilità di scegliere il protocollo di trasporto e la capacita' di determinare alcuni parametri.

Processi di indirizzamento



L'identificatore comprende sia l'indirizzo IP che i numeri di porta associati al processo in esecuzione su un host.

Esempi di numeri di porta: HTTP server: 80, Mail server: 25

I numeri di porta dei processi base sono well-known e sono stati assegnati dallo IANA, quando create una nuova applicazione, dovete assegnarle un nuovo numero di porta.

Protocollo a livello di Applicazione

Un protocollo applicativo definisce:

- *Tipi di messaggi scambiati*, ad esempio messaggi di richiesta e di risposta
- *Sintassi dei tipi di messaggio*, quali sono i campi nel messaggio e come sono descritti
- *Semantica dei campi*, ovvero significato delle informazioni nei campi
- *Regole* per determinare quando e come un processo invia e risponde ai messaggi

E' importante distinguere tra applicazioni di rete e protocolli a livello di applicazione. **Un protocollo a livello di applicazione e' solo una parte(sostanziosa) di un'applicazione di rete.**

Possiamo distinguere i protocolli in due grandi categorie:

1. Protocolli di pubblico dominio:

Definiti nelle RFC , Consente l'interoperabilità , Ma il protocollo applicativo è un pezzo (grande) dell'applicazione; ad esempio, HTTP [RFC 2616] => Web e SMTP [RFC 2821] => e-mail.

2. Protocolli proprietari:

P2P usa essenzialmente protocolli proprietari, ad esempio kazaa, winmx,gnutella.

Servizi di trasporto disponibili per le applicazioni

Trasferimento dati affidabile

alcune applicazioni (ad esempio, audio) possono tollerare qualche perdita, altre applicazioni (ad esempio, trasferimento di file, telnet) richiedono un trasferimento dati affidabile al 100%, se un protocollo fornisce questo tipo di servizio di garanzia nella consegna dati, si dice che fornisce un **trasferimento dati affidabile**.

Throughput

Throughput disponibile = nel contesto di una sessione di comunicazione tra due processi lungo il cammino di rete e' la frequenza alla quale il processo mittente puo' inviare i bit al processo ricevente. Ma considerando il fatto che ci possono esser differenti sessioni, il

Throughput disponibile può fluttuare nel tempo.

- alcune applicazioni (ad esempio, quelle multimediali) per essere "efficaci" richiedono un'ampiezza di banda minima, ovvero hanno dei requisiti di Throughput (chiedono un throughput minimo garantito); *applicazioni sensibili alla banda*.
- altre applicazioni utilizzano l'ampiezza di banda che si rende disponibile, *applicazioni elastiche*.

Temporizzazione

alcune applicazioni interattive in tempo reale (ad esempio, telefonia Internet, giochi interattivi) per essere "realistiche" richiedono piccoli ritardi, ovvero applicano stretti vincoli sulla consegna dei dati.

Sicurezza

Servizi di trasporto offerti da Internet

Internet, come ogni rete TCP/IP, mette a disposizione delle applicazioni due protocolli di trasporto: **UDP e TCP**, i quali offrono un modello di servizio diverso.

Servizio di UDP

vecchio(NON orientato alla connessione (*connectionless*): trasferimento dati inaffidabile fra i processi d'invio e di ricezione; *non offre*: handshake di connessione, affidabilità, controllo di flusso, controllo della congestione, temporizzazione né ampiezza di banda minima)

Web e HTTP

Terminologia

Una pagina web è costituita da oggetti. Un oggetto può essere un file HTML, un'immagine JPEG, un'applet Java, un file audio, ...

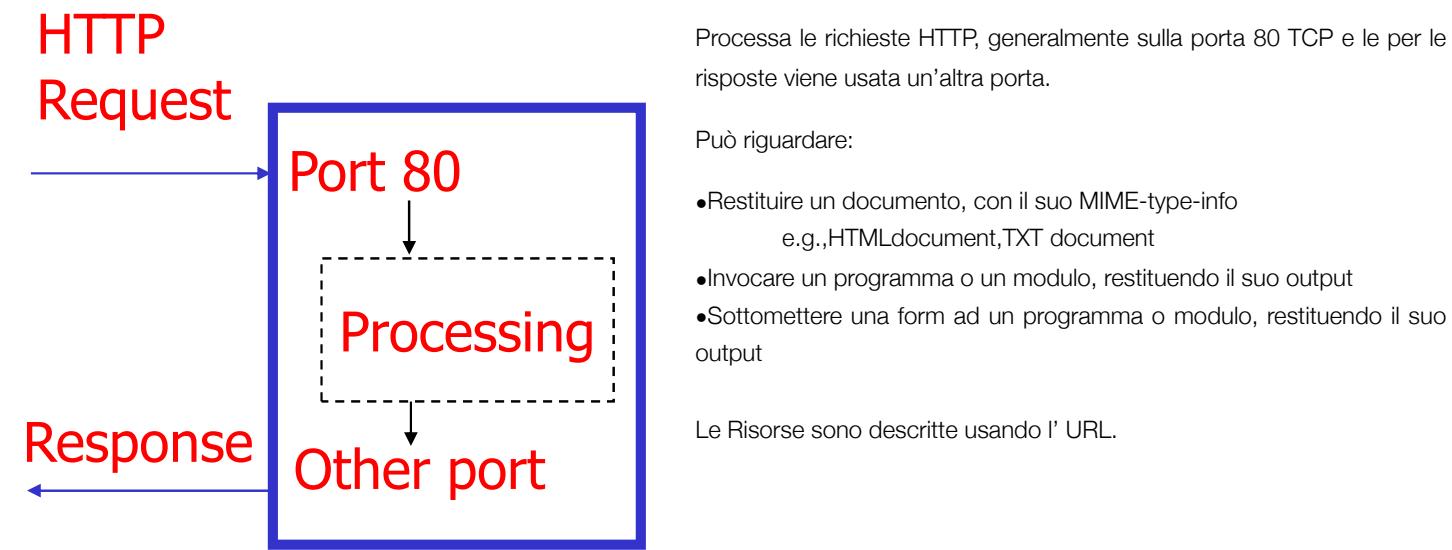
Una pagina web è formata da un file base HTML che include diversi oggetti referenziati. Ogni oggetto è referenziato da un URL.

Esempio di URL (Universal Resource Locator):

`http://www.someschool.edu/someDept/pic.gif`

protocollo nome dell'host nome del percorso

Web (HTTP) Server



URL

Protocol :// Host : Port / Path

URL: Uniform Resource Locator è una sequenza di caratteri che identifica univocamente l'indirizzo di una risorsa in Internet, come un documento o un'immagine.

- Un modo di codificare informazioni di protocollo, login, DNS (o IP) address, di path in una singola stringa
- E' un caso speciale di Uniform Resource Identifier (URI)
 - URL è un URI per un sito da cui possono essere ricavate informazioni varie URN è un URI per un nome

Sintassi generale :

- {partition/protocol}://{userid}:{password}@{domain:port} /{path}
- http://me:too@my.com:1234/index.html#anchor



Reti di Calcolatori

- news://nntp.upenn.edu
- imap://email:me@my.com/folder1
- file:///D:/ity10250/Desktop/DOCS.HTM

Definizione dei campi

Ogni Uniform Resource Locator si compone normalmente di sei parti, alcune delle quali opzionali:

protocollo://<username:password@>nomehost<:porta></percorso><?queryString>

- protocollo

Describe il protocollo da utilizzare per l'accesso al server. I protocolli più comuni sono l'**HTTP**, **HTTPS**, **FTP**, **MMS** ecc. Se il protocollo non viene specificato, generalmente il browser utilizza "HTTP://" come predefinito.

- username:password@ (opzionale)

Subito dopo il protocollo, è possibile specificare l'autenticazione (username e password) per l'accesso alla risorsa. Alcuni server consentono di specificare nell'URL le informazioni per l'autenticazione nel formato "username:password@". Tuttavia, l'autenticazione in URL è estremamente rischiosa in quanto le credenziali di accesso vengono inviate al server in chiaro.

Questo sistema di autenticazione espone inoltre gli utenti al **phishing**. Infatti, i server che non necessitano di autenticazione non considerano in alcun modo i contenuti a sinistra del simbolo "@"; perciò, un URL creato ad arte può portare un utente ad una destinazione completamente diversa da ciò che ci si può aspettare. Ad esempio: l'url <http://www.google.it%7Csearchq=wikipedia@www.microsoft.com> a prima vista sembra portare su Google, ma in realtà la destinazione effettiva è www.microsoft.com.

Da diversi anni, un update di Internet Explorer (832894) ha disattivato la funzione di autenticazione rendendo inefficaci i tentativi di phishing. Altri browser come Firefox hanno mantenuto la funzione, avvertendo tuttavia l'utente del possibile tentativo di phishing.

- nomehost

Rappresenta l'indirizzo fisico del server su cui risiede la risorsa. Può essere costituito da un nome del dominio o da un indirizzo IP.

- porta (opzionale)

Indica al sistema operativo dell'host remoto la porta del processo server al quale affidare la richiesta. Vi è necessità di indicare questo parametro quando il processo server è in ascolto su una porta non conforme allo standard definito. In caso contrario può essere omesso ed il client provvederà a completare questo campo dell'URL con il valore standard associato al protocollo indicato nella richiesta.

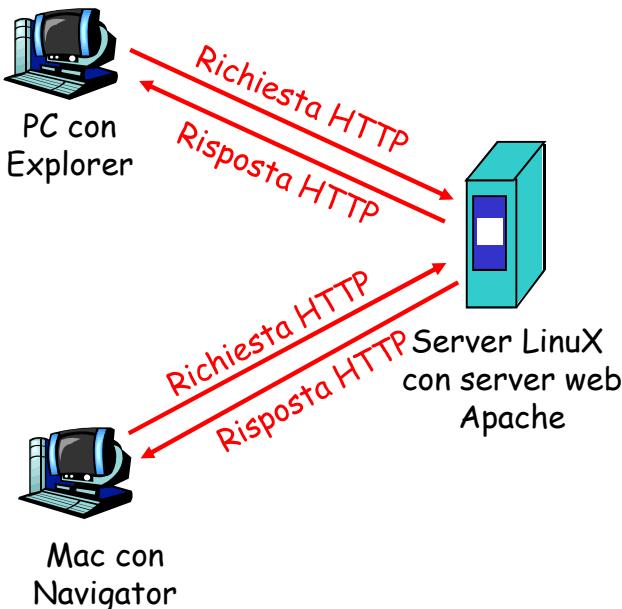
- percorso (opzionale)

Percorso (pathname) nel file system del server che identifica la risorsa (generalmente una pagina web, una immagine o un file multimediale). Se il nome del file non viene specificato, il server può essere configurato per restituire il file predefinito.

- queryString (opzionale)

Se richiesto, al termine dell'url è possibile aggiungere una query string separandola utilizzando il simbolo "?". La query string è una stringa di caratteri che consente di passare al server uno o più parametri. Di norma, la query string ha questo formato: [...]?
parametro1=valore¶metro2=valore2.

Panoramica su HTTP



HTTP: hypertext transfer protocol

Protocollo a livello di applicazione è usato come principale sistema per la trasmissione d'informazioni sul web.

- Modello client/server
- client: il browser (user agent) che richiede, riceve, "visualizza" gli oggetti del Web
- server: il server web invia oggetti in risposta a una richiesta
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068

Usa TCP:

- Il client inizializza la connessione TCP (crea una socket) con il server, sulla porta 80
- Il server accetta la connessione TCP dal client
- Messaggi HTTP scambiati fra browser (client HTTP) e server web (server HTTP), HTTP delega a TCP riguardo l'affidabilità.

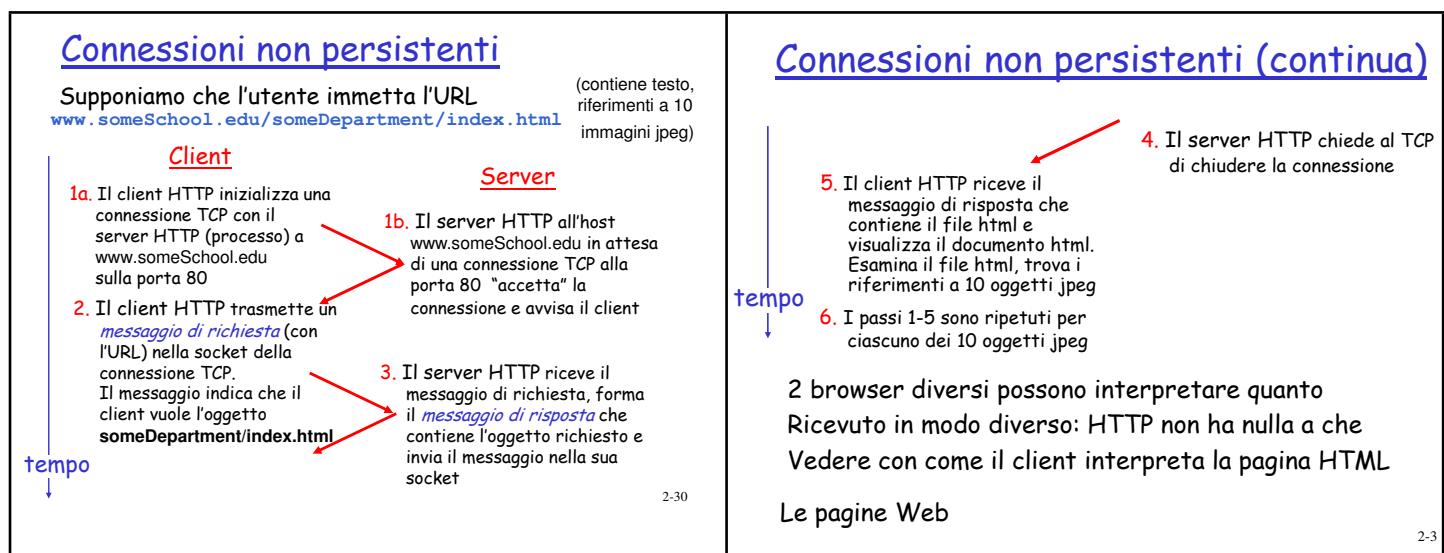
- Connessione TCP chiusa

HTTP è un protocollo "senza stato" (stateless), il server non mantiene informazioni sulle richieste fatte dal client, infatti lo stato viene mantenuto sul browser del client.

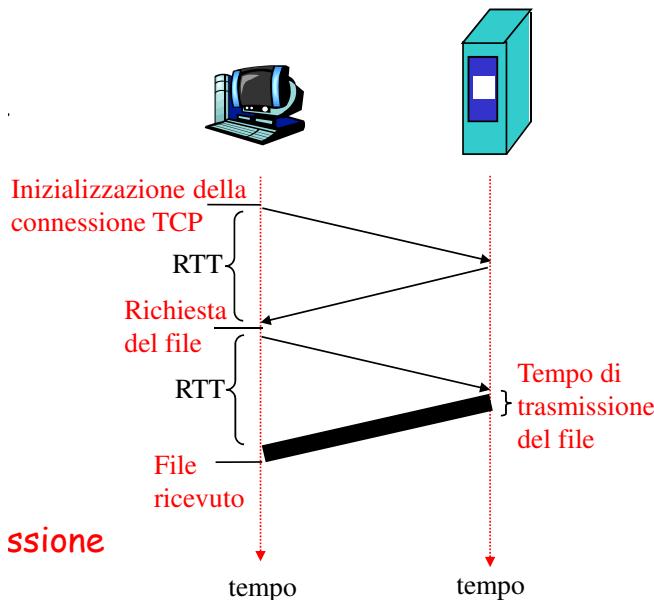
Connessioni HTTP

Connessioni Non Persistenti	Connessioni Persistenti
Ogni coppia richiesta/risposta viene inviata su connessioni TCP separate. HTTP 1.0	Ogni coppia richiesta/risposta sono inviate in un'unica connessione TCP. Senza pipeline, manda e aspetta, con Pipelining manda le richieste una dopo l'altra senza aspettare la risposta delle richieste pendenti.

Connessioni non Persistenti:



Schema del tempo di risposta



Definizione RTT (round trip time): tempo impiegato da un pacchetto di dimensione trascurabile per andare dal client al server e ritornare al client.

Tempo di risposta:

- un RTT per inizializzare la connessione TCP
- un RTT perche' ritornino la richiesta HTTP e i primi byte della risposta HTTP
- tempo di trasmissione del file

$$\text{totale} = 2\text{RTT} + \text{tempo di trasmissione.}$$

Connessioni Persistenti

Lo svantaggio delle connessioni non persistenti sono: il fatto che richiede 2 RTT per un oggetto, overhead del sistema operativo per ogni connessione TCP; inoltre molti browser spesso aprono connessioni TCP parallele per caricare gli oggetti referenziati.

Le connessioni persistenti sono caratterizzate:

- Il server lascia la connessione TCP aperta dopo l'invio di una risposta
- I successivi messaggi tra gli stessi client/server vengono trasmessi sulla connessione aperta

Tali tipi di connessioni si dividono ulteriormente in:

Connessioni Persistenti Senza Pipeline	Connessioni Persistenti Con Pipeline
Il client invia una nuova richiesta solo quando ha ricevuto la risposta precedente, <i>un RTT per ogni oggetto referenziato</i> .	e' la modalità di default in HTTP 1.1, il client invia le richieste non appena riscontra un oggetto referenziato, <i>un solo RTT per tutti gli oggetti referenziati</i> .

Analisi temporale delle connessioni non persistenti

Analisi temporale delle connessioni persistenti

Analisi temporale delle connessioni persistenti senza pipeline

Analisi temporale delle connessioni persistenti con pipeline

Messaggi HTTP

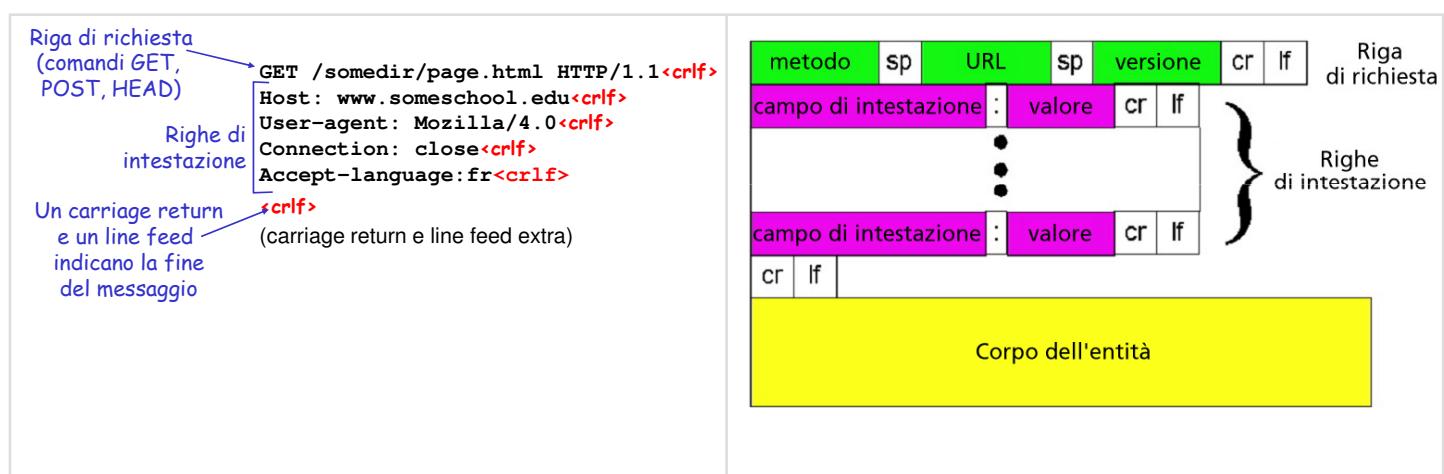
I messaggi HTTP sono essenzialmente due: **richiesta e risposta**.

Messaggi di Richiesta HTTP

Una richiesta HTTP è un insieme di linee inviate al server dal navigatore. Essa comprende :

- **Una linea di richiesta:** è una linea che precisa il tipo di documento richiesto, il metodo che deve essere applicato, e la versione del protocollo usato. La linea comprende tre elementi che devono essere separati da uno spazio :
- **Il metodo**
- **L'URL**
- **La versione del protocollo** usato dal client (generalmente HTTP/1.0)
- **I campi d'intestazione della richiesta:** si tratta di un insieme di linee facoltative che permettono di dare delle informazioni supplementari sulla richiesta e/o il client (Navigatore, sistema operativo,...). Ognuna di queste linee è composta da un nome che qualifica il tipo di intestazione, seguito da due punti (:) e dal valore dell'intestazione
- **Il corpo della richiesta:** è un'insieme di linee opzionali che devono essere separate dalle linee precedenti da una linea vuota e che permettono ad esempio l'invio di dati con un comando POST al momento dell'invio dei dati al server con un formulari

Sono in formato ASCII quindi leggibili dall'utente.



Un esempio di richiesta:

```
GET /beta.html HTTP/1.1
Referer: http://www.alpha.com/alpha.html
Connection: Keep-Alive
User-Agent: Mozilla/4.61 (Macintosh; I; PPC)
Host: www.alpha.com:80
Accept: image/gif, image/jpeg, image/png, /*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1, *, utf-8
```

Request Line: Method URI HTTP-Version\r\n

La request line contiene 3 tokens (word), il carattere “spazio”, separa i tokens e CRLF termina la richiesta.

i 3 token sono :

- Method:
 - GET, recupera le informazioni identificare dall'URI
 - HEAD, recupera meta-informationi sull'URI, ovvero risponde con un messaggio HTTP che tralascia pero' i dati
 - PUT, Memorizza informazioni nella locazione indicata dall'URI
 - POST, invia informazioni ad un URI e riceve i risultati
 - DELETE, rimuove l'entity identificata dall'URI
 - TRACE, usato per tracciare l'HTTP forwarding attraverso proxies,tunnels ecc
 - OPTION, usato per determinare le capacita' del server, o le caratteristiche di una data risorsa.
- URI
 - Universal Resource Identifier
 - RFC 2396
 - Absolute URI:
 - scheme://hostname[:port]/path
 - <http://www.cs.rpi.edu:80/blah/foo>
 - Relative URI:
 - /path
 - /blah/foo (in questo caso non viene menzionato il server)

Le “Header Lines”

Dopo la request line ci sono un certo numero (possibilmente zero) di HTTP header lines. Ogni header line contiene un **nome** di un **attributo** seguito da ":" seguito a sua volta da uno **spazio** e da un **valore**.

Per i metodi GET e HEAD, la fine degli headers corrisponde con la fine della richiesta.

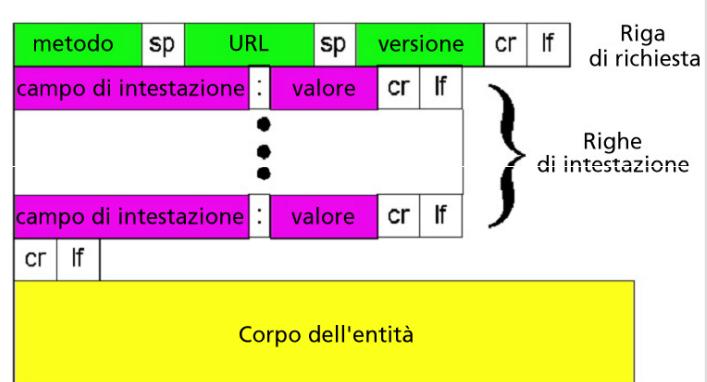
Messaggi di Risposta HTTP

Una risposta HTTP è un insieme di linee inviate dal server al navigatore. Essa comprende :

- **una linea di stato:** è una linea che precisa la versione del protocollo usata e lo stato di trattamento della richiesta attraverso un codice e un testo esplicativo. La linea comprende tre elementi che devono essere separati da uno spazio :
- La **versione** del protocollo usata
- Il **codice dello stato**
- Il **significato del codice**

- I campi d'intestazione della risposta:** si tratta di un insieme di linee facoltative che permettono di dare delle informazioni supplementari sulla risposta e/o il server. Ognuna di queste linee è composta da un nome che qualifica il tipo di intestazione, seguito da due punti (:) e dal valore dell'intestazione
- Il corpo della risposta:** contiene il documento richiesto

```
HTTP/1.0 200 OK
Date: Mon, 28 Jun 2004 10:47:31 GMT
Server: Apache/1.3.29 (Unix) PHP/4.3.4
X-Powered-By: PHP/4.3.4
Vary: Accept-Encoding,Cookie
Cache-Control: private, s-maxage=0, max-age=0, must-revalidate
Content-Language: it
Content-Type: text/html; charset=utf-8
Age: 7673
X-Cache: HIT from wikipedia.org
Connection: close
```

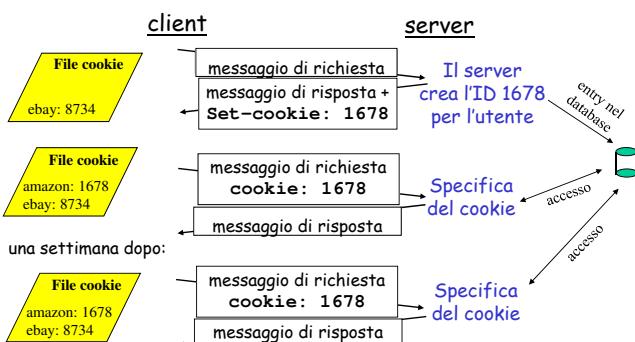


Status Code

- Lo status code è un numero di tre cifre, di cui la prima indica la classe della risposta, e le altre due la risposta specifica. Esistono le seguenti classi:
- 1xx: Informational. Una risposta temporanea alla richiesta, durante il suo svolgimento.
- 2xx: Successful. Il server ha ricevuto, capito e accettato la richiesta.
- 3xx: Redirection. Il server ha ricevuto e capito la richiesta, ma sono necessarie altre azioni da parte del client per portare a termine la richiesta.
- 4xx: Client error. La richiesta del client non può essere soddisfatta per un errore da parte del client (errore sintattico o richiesta non autorizzata).
- 5xx: Server error. La richiesta può anche essere corretta, ma il server non è in grado di soddisfare la richiesta per un problema interno (suo o di applicazioni CGI).

Interazione utente-server: i Cookie

Sono adottati da HTTP per autenticare gli utenti allo scopo di limitare gli accessi al server, fornire i contenuti in funzione delle loro identità e tener traccia degli utenti.



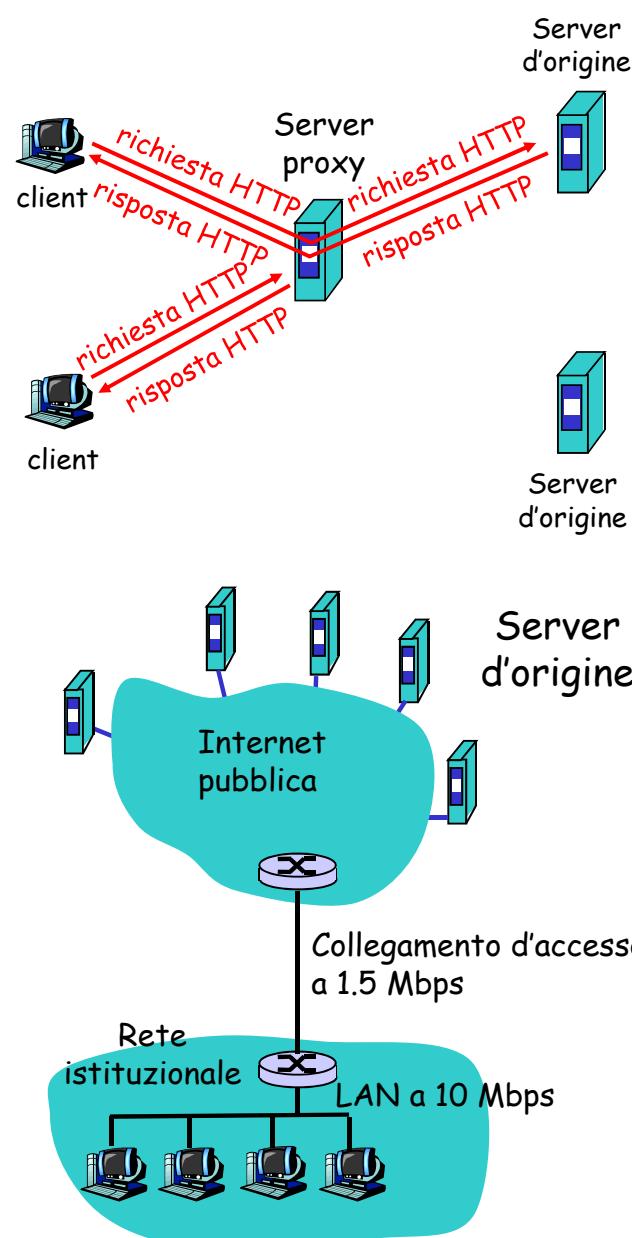
Quattro componenti:

- Una riga di intestazione nel messaggio di risposta HTTP
- Una riga di intestazione nel messaggio di richiesta HTTP
- Un file cookie mantenuto sul sistema terminale dell'utente e gestito dal browser dell'utente (al momento della prima richiesta HTTP da parte del client, il server crea un identificativo univoco(ID) e una entry nel database per ID).
- Un database sul sito

I cookie possono contenere autorizzazioni, carta per gli acquisti, raccomandazioni, stato della sessione dell'utente.. ma si noti che il client e il browser non possono mai leggere i cookie, i quali sono preparati e consumati dai server.

Caching Web (HTTP Proxy Server)

E' un entità di rete con l'obiettivo e' quello di soddisfare la richiesta del cliente senza coinvolgere il server d'origine.



L'utente configura il browser per l'accesso al Web tramite il proxy

Il browser trasmette tutte le richieste HTTP al proxy e

- se l'oggetto e' nella cache, gli viene restituito
- altrimenti, la cache richiede l'oggetto al server d'origine lo salva nella cache e poi lo inoltra al client.

La cache opera come client e come server e tipicamente la cache è installata da un ISP (università, aziende o ISP residenziali).

Perché il caching Web??

- Riduce i tempi di risposta alle richieste dei client, in particolare se l'ampiezza di banda costituisce il collo di bottiglia.
- Riduce il traffico sul collegamento di accesso a Internet.
- Internet arricchita di cache consente ai provider "scadenti" di fornire dati con efficacia (ma così fa la condivisione di file P2P)

Esempio di caching

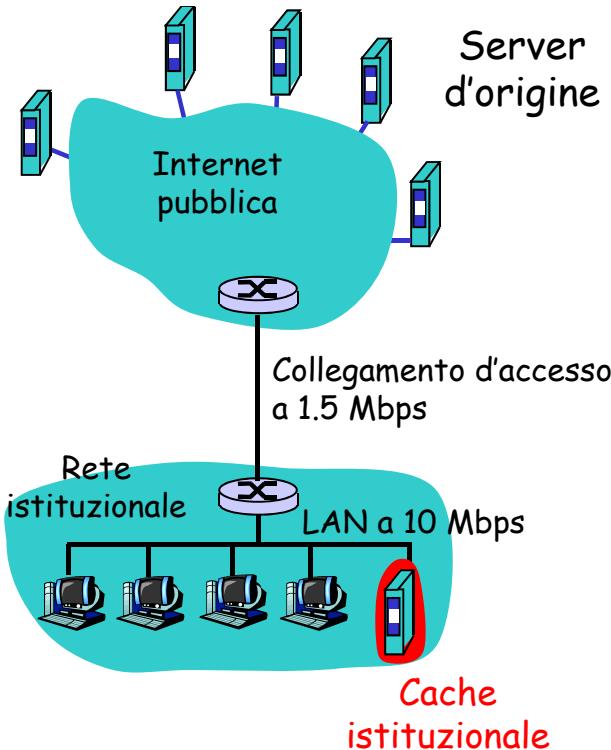
Ipotesi

- Dimensione media di un oggetto = 100.000 bit
- Frequenza media di richieste dai browser istituzionali ai server d'origine = 15/sec
- Ritardo dal router istituzionale a qualsiasi server d'origine e ritorno al router = 2 sec

Conseguenze

- utilizzazione sulla LAN = 15%
- utilizzazione sul collegamento d'accesso = 100%
- ritardo totale = ritardo di Internet + ritardo di accesso + ritardo della LAN = 2 sec + minuti + millisecondi

Soluzione possibile



- aumentare l'ampiezza di banda del collegamento d'accesso a 10 Mbps, per esempio

Conseguenze

- utilizzazione sulla LAN = 15%
- utilizzazione sul collegamento d'accesso = 15%
- ritardo totale = ritardo di Internet + ritardo di accesso + ritardo della LAN = 2 sec + msec + msec
- l'aggiornamento spesso è molto costoso

Soluzione Alternativa:

Installare la cache

- supponiamo una percentuale di successo (hit rate) pari a 0,4

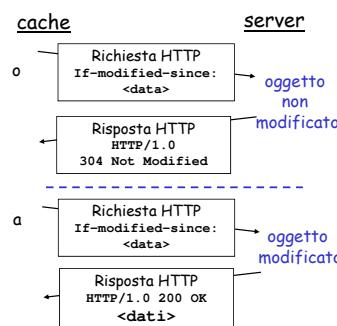
Conseguenze

- il 40% delle richieste sarà soddisfatto quasi immediatamente
- il 60% delle richieste sarà soddisfatto dal server d'origine
- l'utilizzazione del collegamento d'accesso si è ridotta al 60%, determinando ritardi trascurabili (circa 10 msec)
- ritardo totale medio = ritardo di Internet + ritardo di accesso + ritardo della LAN = $0,6 \cdot (2,01)$ sec + millisecondi < 1,4 sec

Get Condizionale

E' un messaggio di richiesta HTTP di tipo GET il quale include una riga d'intestazione *If-Modified-Since*:. Ha come obiettivo quello di verificare che la copia in cache sia valida/aggornata rispetto a quella presente sul server origine.

- cache: specifica la data della copia dell'oggetto nella richiesta HTTP: *If-modified-since: <data>*
- server: la risposta non contiene l'oggetto se la copia nella cache è aggiornata: HTTP/1.0 304 Not Modified



Q1. Indicare –giustificando la risposta– quanti pacchetti IP può ricevere in meno nel caso ottimo un cliente HTTP se specifica il campo *If-Modified-Since* nella richiesta di un file di dimensione $(K+0.5) \cdot M$, dove K è un intero e M è il valore di MSS nella connessione stabilita col server.

R1. Il caso ottimo è quello in cui il cliente possiede una versione aggiornata del file e non si verificano perdite, corruzioni né frammentazioni dei pacchetti. In tale caso il cliente riceverà la risposta HTTP ("304 Not Modified") in 1 solo pacchetto IP. Se invece il cliente non possiede una versione aggiornata del file, il server dovrà inviare almeno $(K+1)$ pacchetti IP per trasmettere il file (oltre a eventuali altri pacchetti dovuti a ritrasmissioni). Nel caso ottimo il cliente riceverà quindi almeno K pacchetti in meno.

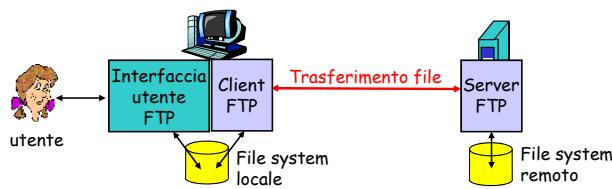
Trasferimento di file: FTP

FTP e' un Protocollo per la trasmissione di dati tra host basato su TCP.

Perché abbiamo bisogno di un FTP Service? Lo scopo e' quello di trasferire file tra due host (locale-remoto): Promuovere il file sharing (programmi e/o dati) , Incoraggiare l'uso indiretto/implicito di host remoti , Trasferire i dati in modo affidabile e efficiente e rimuovere in maniera trasparente l'incompatibilita' tra differenti sistemi si stoccaggio file tra host

Inoltre vi sono dei **problem**i riguardanti i File Transfer in particolare si ha che Sistemi eterogenei usano differenti: Operating Systems, Character Sets, Naming Conventions, Directory Structures, File Structures and Formats, **Data representation**.

FTP deve indirizzare e risolvere questi problemi!!

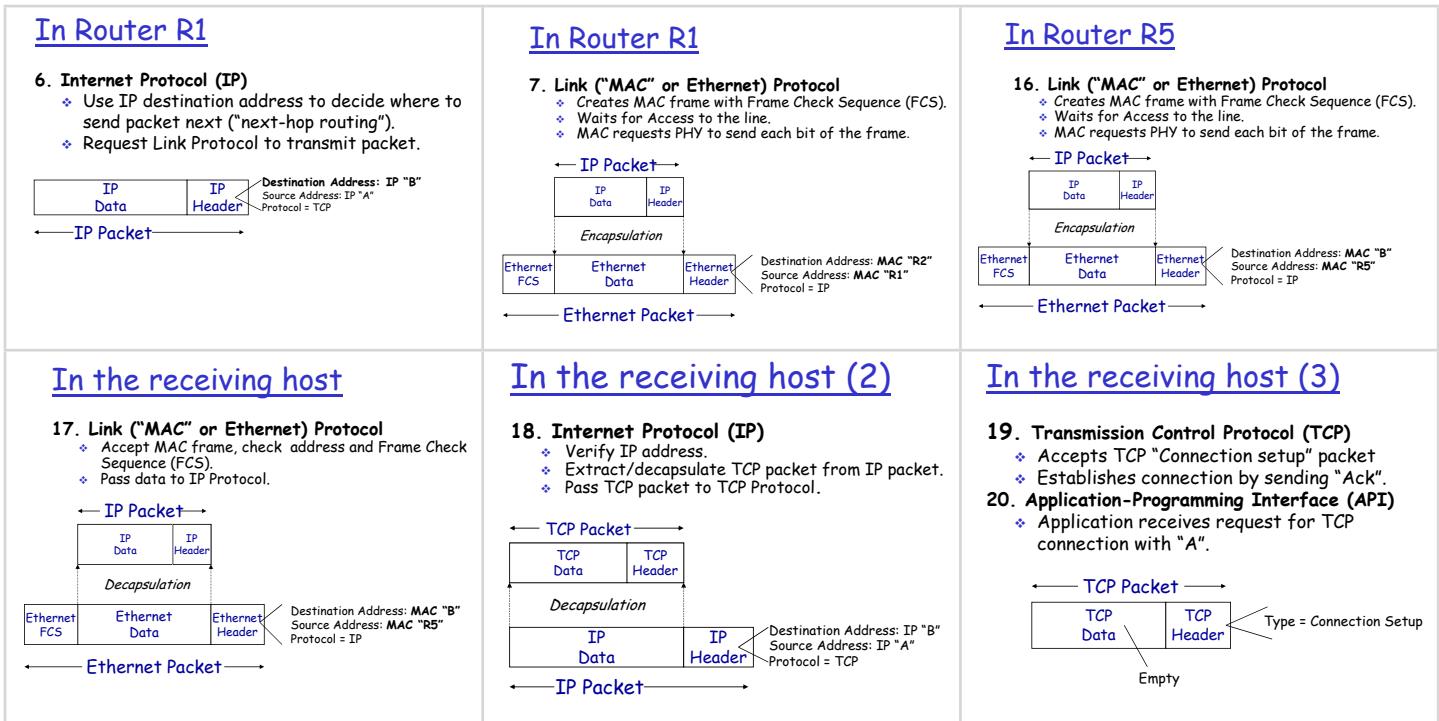


FTP e' un protocollo per il trasferimento file da un host locale a un host remoto e viceversa. Utilizza il modello **client-server**, il client e' il lato che inizia il trasferimento ed il server e' l'host remoto.

Il server FTP e' associato alla porta **21**, sulla quale si ha la connessione di controllo TCP; mentre la connessione dati TCP avviene sulla porta **20** del server FTP.

Il modello FTP

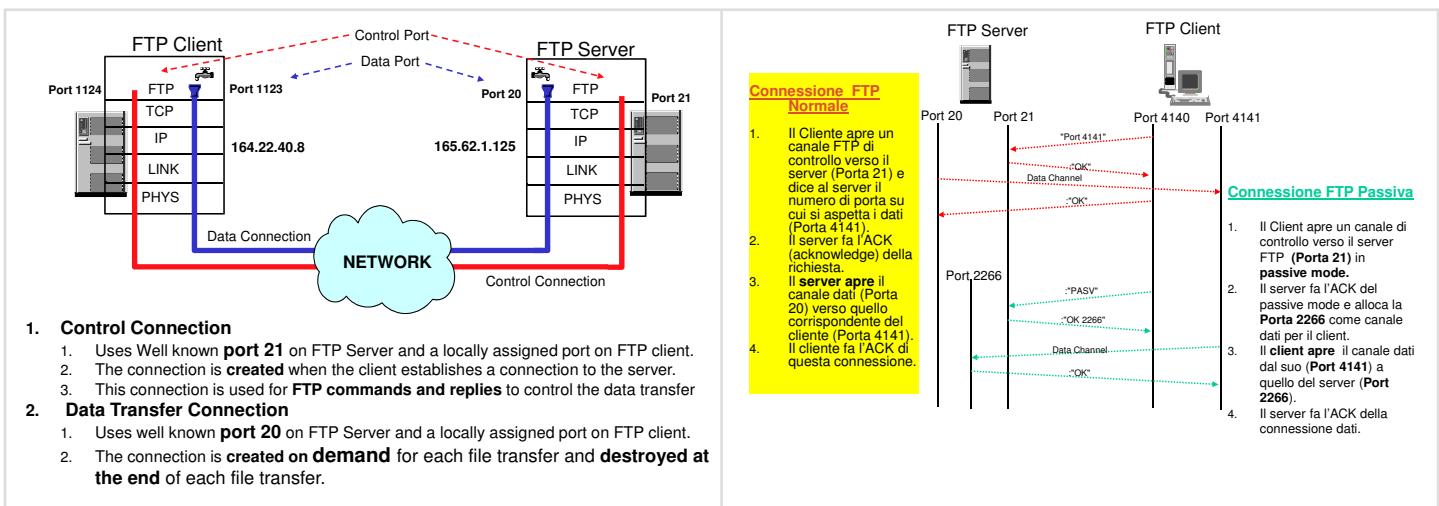
<p>PI: Protocol Interpreter DTP: Data Transfer Protocol</p>	<p>Example: FTP su Internet Usando TCP/IP ed Ethernet</p>	<ol style="list-style-type: none"> Application-Programming Interface (API) <ul style="list-style-type: none"> Application requests TCP connection with "B" Transmission Control Protocol (TCP) <ul style="list-style-type: none"> Creates TCP "Connection setup" packet TCP requests IP packet to be sent to "B" <p>TCP Packet</p>
<p>3. Internet Protocol (IP)</p> <ul style="list-style-type: none"> Creates IP packet with correct addresses. IP requests packet to be sent to router. <p>TCP Packet</p> <p>Encapsulation</p> <p>Destination Address: IP "B" Source Address: IP "A" Protocol = TCP</p> <p>IP Packet</p>	<p>4. Link ("MAC" or Ethernet) Protocol</p> <ul style="list-style-type: none"> Creates MAC frame with Frame Check Sequence (FCS). Wait for Access to the line. MAC requests PHY to send each bit of the frame. <p>IP Packet</p> <p>Encapsulation</p> <p>Destination Address: MAC "R1" Source Address: MAC "A" Protocol = IP</p> <p>Ethernet Packet</p>	<p>In Router R1</p> <p>5. Link ("MAC" or Ethernet) Protocol</p> <ul style="list-style-type: none"> Accept MAC frame, check address and Frame Check Sequence (FCS). Pass data to IP Protocol. <p>IP Packet</p> <p>Decapsulation</p> <p>Destination Address: MAC "R1" Source Address: MAC "A" Protocol = IP</p> <p>Ethernet Packet</p>



FTP: connessione di controllo, connessione dati

I vari step per lo scambio di dati sono i seguenti:

- Il client FTP contatta il server FTP sulla porta 21 (connessione di controllo), specificando TCP come protocollo di trasporto
- Il client ottiene l'autorizzazione sulla connessione di controllo
- Il client cambia la directory remota inviando i comandi sulla connessione di controllo
- Quando il server riceve un comando per trasferire un file, apre una connessione dati TCP con il client sulla porta 20
- Dopo il trasferimento di un file, il server chiude la connessione dati (NON quella di controllo)
- Il server apre una seconda connessione dati TCP per trasferire un altro file.
- Connessione di controllo: **"fuori banda"** (**out of band**)
- Il server FTP mantiene lo "stato": directory corrente, autenticazione precedente



Comandi e risposte FTP

Comandi Comuni	Codici di ritorno Comuni
<ul style="list-style-type: none"> Inviati come testo ASCII- 7 bits- sulla connessione di controllo USER username PASS password LIST - elenca i file della directory corrente RETR filename - recupera (get) un file dalla directory corrente STOR filename - memorizza (put) un file nell'host remoto 	<ul style="list-style-type: none"> Codice di stato ed espressione (come in HTTP) 331 Username OK, password required 125 data connection already open; transfer starting 425 Can't open data connection 452 Error writing file

FTP deve mantenere lo stato dell'utente, deve associare la connessione di controllo a uno specifico utente e tener conto della directory corrente. Perciò un server può aprire poche connessioni FTP contemporaneamente.

Perché Out-of-Band di controllo?

- Evita la necessità di segnare la fine del trasferimento dei dati**
 - trasferimento dati si conclude con la chiusura della connessione dati
 - Eppure, la connessione di controllo rimane in piedi
- Interruzione di un trasferimento di dati**
 - può interrompere il trasferimento senza uccidere la connessione di controllo
 - ... che evita che l'utente debba accedere nuovamente
 - Fatto con un ABOR sulla connessione di controllo
- il trasferimento di file tra due host**
 - le connessioni dati potrebbero andare ad host diversi
 - ... inviando un diverso indirizzo IP del client al server
 - Ad esempio, coordinate dell'utente per il trasferimento tra due server

Approfondimento funzionamento generale

In un canale dati di tipo **attivo** il client apre una porta tipicamente random, tramite il canale comandi rende noto il numero di tale porta al server e attende che esso si connetta. Una volta che il server ha attivato la connessione dati al client FTP, quest'ultimo effettua il binding della porta sorgente alla porta 20 del server FTP. A tale scopo possono venire impiegati i comandi PORT o EPRT, a seconda del protocollo di rete utilizzato (tipicamente [IPv4](#) o [IPv6](#)).

In un canale dati di tipo **passivo** il server apre una porta tipicamente random (> 1023), tramite il canale comandi rende noto il numero di tale porta al client e attende che esso si connetta. A tale scopo possono venire impiegati i comandi PASV o EPSV, a seconda del protocollo di rete utilizzato (tipicamente [IPv4](#) o [IPv6](#)).

Sia il canale comandi sia il canale dati sono delle connessioni TCP; FTP crea un nuovo canale dati per ogni file trasferito all'interno della sessione utente, mentre il canale comandi rimane aperto per l'intera durata della sessione utente, in altre parole il canale comandi è persistente mentre il canale dati è non persistente.

Un server FTP offre svariate funzioni che permettono al client di interagire con il suo filesystem e i file che lo popolano, tra cui:

- Download/upload di file.
- Resume di trasferimenti interrotti.
- Rimozione e rinomina di file.
- Creazione di directory.
- Navigazione tra directory.



Reti di Calcolatori

FTP fornisce inoltre un **sistema di autenticazione** (N.B. in chiaro) degli accessi. Il client che si connette potrebbe dover fornire delle credenziali a seconda delle quali gli saranno assegnati determinati privilegi per poter operare sul filesystem. L'autenticazione cosiddetta "anonima" prevede che il client non specifichi nessuna password di accesso e che lo stesso abbia privilegi che sono tipicamente di "sola lettura".

Problemi relativi alla sicurezza

La specifica originale di FTP non prevede alcuna cifratura per i dati scambiati tra client e server. Questo comprende nomi utenti, password, comandi, codici di risposta e file trasferiti i quali possono essere "snifati" o visionati da malintenzionati in determinate situazioni (esempio: ambienti Intranet).

Il problema è comune a diversi altri protocolli utilizzati prima della diffusione di SSL quali HTTP, TELNET e SMTP. Per ovviare al problema è stata definita una nuova specifica che aggiunge al protocollo FTP originale un layer di cifratura SSL/TLS. Più una nuova serie di comandi e codici di risposta. Il protocollo prende il nome di FTPS ed è definito nella [RFC-4217](#). Da non confondersi con SFTP che è comunque una valida alternativa per ovviare al problema descritto.

Posta Elettronica

E' un servizio di comunicazione asincrono, senza coordinazione tra mittente e destinatario. E' veloce, bassi costi, mailing list con possibilità di includere video, foto, audio, file html ecc, i contro però sono dovuti agli spam, virus, segretezza e forged e-mail.

Tre componenti principali:

- user agent (agente utente)
- mail server (server di posta)
- simple mail transfer protocol: SMTP

User Agent

Detto anche mail reader, consentono all'utente di leggere, rispondere, inoltrare, salvare e comporre messaggi di posta elettronica. I messaggi in uscita o in arrivo sono memorizzati sul mail server.

Il termine "client" viene utilizzato perché il servizio di posta elettronica si basa su un'architettura [client-server](#). Il client si occupa della composizione e della lettura, il server si occupa della trasmissione dei messaggi.

Esiste una particolare categoria di client e-mail definita "client [webmail](#)". Si tratta di particolari programmi eseguiti da siti web che permettono di gestire i propri messaggi da qualsiasi postazione: da casa, dall'ufficio, in un Netcafé, ecc.

Mail Server

Nucleo del sistema postale. Le principali operazioni sono la ricezione della posta, che viene consegnata nella casella di posta dei destinatari; lo smistamento della posta consegnata dagli utenti verso altri server di posta; l'accesso da parte degli utenti alle proprie caselle di posta.

La ricezione e lo smistamento si svolgono oggi prevalentemente tramite il protocollo [SMTP](#). Per l'accesso, i protocolli più usati sono [POP3](#) che scarica la posta sul computer dell'utente, e [IMAP](#) che mantiene la posta sul server in modo che l'utente possa accedervi da diversi computer.

E' composto da:

- Casella di posta (mailbox) contiene i messaggi in arrivo per l'utente, ogni utente ha la propria.
- Coda di messaggi da trasmettere (in uscita).
- Protocollo SMTP tra server di posta per inviare messaggi di posta elettronica
 - client: server di posta trasmittente
 - "server": server di posta ricevente

SMTP

Rappresenta il principale protocollo a livello di applicazione per la posta elettronica su Internet. Usa TCP per trasferire in modo affidabile i messaggi di posta elettronica dal server di posta del mittente a quello del destinatario aprendo una connessione sulla porta 25

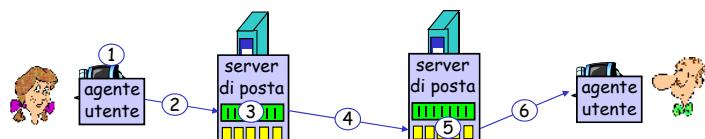
Per associare il server SMTP a un dato nome di dominio ([DNS](#)) si usa un Resource Record di tipo MX (Mail eXchange) ([Tipi di record DNS](#)).

Poiché SMTP è un protocollo testuale basato sulla codifica [ASCII](#) (in particolare ASCII NVT), non è permesso trasmettere direttamente testo composto con un diverso set di caratteri e tantomeno file binari. Lo standard [MIME](#) permette di estendere il formato dei messaggi mantenendo la compatibilità col software esistente. Per esempio, al giorno d'oggi molti server SMTP supportano l'estensione [8BITMIME](#), la quale permette un trasferimento di un testo che contiene caratteri accentati (non-ASCII) senza bisogno di trascodificarlo. Altri limiti di SMTP, quale la lunghezza massima di una riga, impediscono la spedizione di file binari senza [trascodifica](#). (Nota che per i file binari inviati con [HTTP](#) si utilizza il formato [MIME](#) senza bisogno di una trascodifica.)

Opzioni del trasferimento:

- trasferimento diretto: il server trasmittente al server ricevente
- tre espressioni per il trasferimento
 - handshaking (saluto)
 - trasferimento di messaggi
 - chiusura
 - Utilizza connessioni persistenti
- interazione comando/risposta
 - comandi: testo ASCII
 - risposta: codice di stato ed espressione
- i messaggi (intestazione e corpo) devono essere nel formato [ASCII a 7 bit](#)

Scenario classico



- 1) Alice usa il suo agente utente per comporre il messaggio da inviare a bob@someschool.edu
- 2) L'agente utente di Alice invia un messaggio al server di posta di Alice; il messaggio è posto nella coda di messaggi
- 3) Il lato client di SMTP apre una connessione TCP con il server di posta di Bob
- 4) Il client SMTP invia il messaggio di Alice sulla connessione TCP
- 5) Il server di posta di Bob pone il messaggio nella mail box di Bob
- 6) Bob invoca il suo agente utente per leggere il messaggio

Problemi che si possono verificare: errore nel proprio MailServer [MS non accessibile al momento], errore di scrittura dopo la 3, errore a sx della 6 MS lancia un errore.

```

S: 220 www.example.com ESMTP Postfix
C: HELO mydomain.com
S: 250 Hello mydomain.com, pleased to meet you
C: MAIL FROM: <sender@mydomain.com>
S: 250 sender@mydomain.com ... Sender ok
C: RCPT TO: <friend@example.com>
S: 250 friend@example.com ... Recipient Ok
C: DATA
S: 354 End data with "." on a line by itself
C: Subject: messaggio di prova
C: From: sender@mydomain.com
C: To: friend@example.com
C:
C: Ciao,
C: questa è una prova.
C:
S: 250 Ok: queued as 12345
C: QUIT
S: 221 Bye

```

Sicurezza

Una delle limitazioni del protocollo SMTP originario è che non gestisce l'[autenticazione](#) dei mittenti. Oltre al rischio di spam, esiste la possibilità di inviare e-mail facendo apparire come mittente l'indirizzo corrispondente ad un altro account. Senza accedere all'account di terzi, è possibile stabilire una connessione al mail-server e scrivere un messaggio in codice SMTP contenente i comandi relativi a mittente e destinatario, dare i relativi parametri e il corpo della e-mail.

Per ovviare a questi problemi è stata sviluppata un'estensione chiamata [SMTP-AUTH](#).

Nonostante questo, lo [spam](#) rimane ancor oggi un grave problema. Tuttavia, non si ritiene praticabile una revisione radicale del protocollo SMTP, per via del gran numero di implementazioni del protocollo attuale

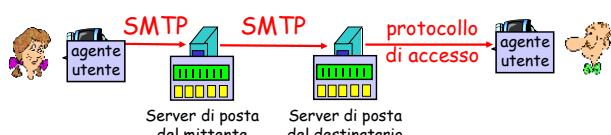
Confronto con HTTP

HTTP: pull protocol

SMTP: push protocol

Entrambi hanno un'interazione comando/risposta in ASCII, con codici di stato. HTTP in ogni oggetto e' incapsulato nel suo messaggio di risposta, in SMTP piu' oggetti vengono trasmessi in un unico messaggio.

Protocolli di accesso alla posta



Lo scenario che troviamo e' il seguente:

- **SMTP**: consegna/memorizzazione sul server del destinatario
- Protocollo di accesso alla posta: ottenere i messaggi dal server
 - **POP**: Post Office Protocol [RFC 1939]
 - autorizzazione (agente <-> server) e download
 - **IMAP**: Internet Mail Access Protocol [RFC 1730]
 - più funzioni (più complesse)
 - manipolazione di messaggi memorizzati sul server
 - **HTTP WebMail**: Hotmail , Yahoo! Mail, ecc.

La spedizione: SMTP	La ricezione: pop3 e Imap
<ul style="list-style-type: none"> Per inviare un messaggio si utilizza un programma in grado di utilizzare il protocollo SMTP. Quando il server SMTP riceve il messaggio, che include il mittente (autenticato) ed il destinatario inizia la ricerca del percorso su cui instradare il messaggio. Per far ciò analizza la parte relativa al dominio del destinatario (la stringa che segue il simbolo @) e ricava l'indirizzo IP. Instrandando tramite i vari DNS il messaggio verso l'appropriato percorso, il messaggio arriva a destinazione. 	<ul style="list-style-type: none"> Sul server del destinatario esistono varie caselle postali: una per ogni account attivo, create in seguito alla registrazione degli utenti del servizio di posta elettronica gestito dal dominio. Su questo server un software di gestione dei messaggi ricevuti smista la corrispondenza memorizzando in ciascuna casella i dati ricevuti e li mantiene fino a quando l'utente non si collega tramite un client di lettura di e-mail. I software client più diffusi sono Outlook express, Outlook, Eudora, FireBird e Messenger. Questi programmi, oltre ai parametri di spedizione, contengono anche le configurazioni per la ricezione dei messaggi tramite protocollo POP3 o Imap.

POP3 [RFC 1939]

Post Office Protocol, e' un protocollo di accesso alla posta estremamente semplice pero' le sue funzionalità sono piuttosto limitate. POP3 entra in azione quando l'agente utente apre una connessione **TCP** verso il mail server sulla porta **110**. Una volta stabilita la connessione si succederanno 3 fasi :

```

S:+OK <22593.1129980067@example.com>
C:USER pippo
S:+OK
C:PASS pluto
S:+OK
C:LIST
S:+OK
 1 817
 2 124
.
C:RETR 1
S:+OK
  Return-Path: <pippo@example.org>
  Delivered-To: pippo@example.org
  Date: Sat, 22 Oct 2005 13:24:54 +0200
  From: Mario Rossi <mario@rossi.org>
  Subject: xxxx
  Content-Type: text/plain; charset=ISO-8859-1

  testo messaggio
.
C:DELE 1
S:+OK
C:QUIT
S:+OK

```

- autenticazione, il client manda nome utente e password in **chiaro**
 - transazione, l'agente utente chiede al server di elencare le dimensioni dei messaggi memorizzati, quindi li recupera i messaggi e può marcare i messaggi per la cancellazione
 - aggiornamento, ha luogo dopo che il client ha inviato il comando quit, che conclude la sessione. A questo punto il server cancella i messaggi marcati per la cancellazione.
- Il server può dare due tipi di risposta ai comandi inviati dall'agente utente:
- +OK per indicare la correttezza del comando precedente, spesso trasporta anche dati
 - ERR utilizzo del server per indicate che qualcosa non ha funzionato nel comando precedente.
- Ci sono due tipi di transazione: **Scarica e cancella, scarica e mantieni**.

Durante una sessione tra agente utente e il server di posta, il server POP3 mantiene alcune informazioni di stato; in particolare, tiene traccia dei messaggi dell'utente marcati come cancellati. Comunque , il server POP3 non trasporta informazioni di stato tra sessioni POP3.

IMAP [RFC 3501]

E' un protocollo di accesso alla posta, ma presenta *maggiori potenzialità rispetto a POP3* ed e' quindi assai più complesso.

La porta predefinita del demone IMAP sull'host è la 143. Se si utilizza una connessione sicura tramite SSL, allora la porta è la 993.

IMAP consente di mantenere una **gerarchia di cartelle sul server remoto**. Fornisce quindi comandi per creare una nuova cartella, spostare i messaggi e effettuare ricerche nelle cartelle, inoltre e' possibile *mantenere informazioni sullo stato* (nomi della cartella, associazione tra i messaggi e le cartelle).

Permette di scaricare solo l'intestazione o parte del messaggio ad esempio in caso di connessione lente.

Ecco un elenco delle caratteristiche dell'IMAP ma non del POP:

- Accesso alla posta sia online che off-line
 - Mentre si utilizza il POP3, il client si connette per scaricare i nuovi messaggi e poi si disconnette. Con l'IMAP il client rimane connesso e risponde alle richieste che l'utente fa attraverso l'interfaccia; questo permette di risparmiare tempo se ci sono messaggi di grandi dimensioni.
- Più utenti possono utilizzare la stessa casella di posta
 - Il protocollo POP assume che un solo client (utente) è connesso ad una determinata mailbox (casella di posta), quella che gli è stata assegnata. Al contrario l'IMAP4 permette connessioni simultanee alla stessa mailbox, fornendo meccanismi per controllare i cambiamenti apportati da ogni utente.
- Supporto all'accesso a singole parti [MIME](#) di un messaggio
 - La maggior parte delle e-mail sono trasmesse nel formato MIME, che permette una struttura ad albero del messaggio, dove ogni ramo è un contenuto diverso (intestazioni, allegati o parti di esso, messaggio in un dato formato, eccetera). Il protocollo IMAP4 permette di scaricare una singola parte MIME o addirittura sezioni delle parti, per avere un'anteprima del messaggio o per scaricare una mail senza i file allegati.
- Supporto per attributi dei messaggi tenuti dal server.
 - Attraverso l'uso di attributi, tenuti sul server, definiti nel protocollo IMAP4, ogni singolo client può tenere traccia di ogni messaggio, per esempio per sapere se è già stato letto o se ha avuto una risposta.
- Accesso a molteplici caselle di posta sul server
 - Alcuni utenti, con il protocollo IMAP4, possono creare, modificare o cancellare mailbox (di solito associate a cartelle) sul server. Inoltre, questa gestione delle mailbox, permette di avere cartelle condivise tra utenti diversi.
- Possibilità di fare ricerche sul server
 - L'IMAP4 permette al client di chiedere al server quali messaggi soddisfano un certo criterio, per fare, per esempio, delle ricerche sui messaggi senza doverli scaricare tutti.
- Supporto di un meccanismo per la definizione di estensioni
 - Nelle specifiche dell'IMAP è descritto come un server può far sapere agli utenti se ha delle funzionalità extra. Molte estensioni dell'IMAP sono molto diffuse, ad esempio l'[IMAP Idle](#) [1].
- L'IMAP è principalmente utilizzato nelle grandi [network](#) come università o aziende, dove un utente cambia postazione spesso: con il POP3, sarebbe necessario scaricare i messaggi ogni volta che si cambia pc, mentre con l'IMAP si possono scaricare solo i nuovi messaggi o accedere ad un messaggio specifico senza dover scaricare gli altri.

Domain Name System (DNS)

Ad ogni risorsa TCP/IP può essere assegnato un nome simbolico. Per fare ciò sono necessari: un metodo per associare il nome simbolico di una macchina all'indirizzo (o agli indirizzi) IP (**risoluzione diretta**) e un metodo per associare ad un indirizzo IP il nome simbolico della macchina (**risoluzione inversa**)

Domain Name System (DNS), definito presso ISI - USC 1984, RFC 882, RFC 883, RFC 973 (obsolete), [RFC 1034](#), [RFC 1035](#), [RFC 1123](#), [RFC 1537](#), [RFC 1912](#).

Il DNS utilizza il protocollo di trasporto UDP e la porta 53. È un servizio utilizzato per la risoluzione di nomi di host in indirizzi IP e viceversa. Il servizio è realizzato tramite un database distribuito, costituito dai server DNS.

Il nome DNS denota

- il protocollo che regola il funzionamento del servizio,
- i programmi che lo implementano,
- i server su cui questi girano,
- l'insieme di questi server che cooperano per fornire il servizio.

L'operazione di convertire un nome in un indirizzo è detta risoluzione DNS e convertire un indirizzo IP in nome è detto risoluzione inversa.

Caratteristiche Principali

- La possibilità di attribuire un nome testuale facile da memorizzare a un server (ad esempio un sito [world wide web](#)) migliora di molto l'uso del servizio, in quanto noi esseri umani troviamo più facile ricordare nomi testuali (mentre gli host e i [router](#) sono raggiungibili utilizzando gli indirizzi IP numerici). Per questo, il DNS è fondamentale per l'ampia diffusione di internet anche tra utenti non tecnici, ed è una delle sue caratteristiche più visibili.
- È possibile attribuire più nomi allo stesso indirizzo IP (o viceversa) per rappresentare diversi servizi o funzioni forniti da uno stesso host (o più host che erogano lo stesso servizio). Questa flessibilità risulta utile in molti casi:
- Nel caso il server debba sostituire il server che ospita un servizio, o si debba modificare il suo indirizzo IP, è sufficiente modificare il record DNS, senza dover intervenire sui client.
- Un utilizzo molto popolare di questa possibilità è il cosiddetto [virtual hosting](#) basato sui nomi, una tecnica per cui un [web server](#) dotato di una singola interfaccia di rete e di singolo indirizzo IP può ospitare più siti web, usando l'indirizzo alfanumerico trasmesso nell'header [HTTP](#) per identificare il sito per cui viene fatta la richiesta.
- Utilizzando nomi diversi per riferirsi ai diversi servizi erogati da un host, è possibile spostare una parte dei servizi su un altro host, e spostare i client su questo nuovo host modificando il suo record nel DNS.
- Facendo corrispondere più indirizzi IP a un nome, il carico dei client viene distribuito su diversi server, ottenendo un aumento delle prestazioni complessive del servizio e una tolleranza ai guasti (ma è necessario assicurarsi che i diversi server siano sempre allineati, ovvero offrano esattamente lo stesso servizio ai client).
- La risoluzione inversa è utile per identificare l'identità di un host, o per leggere il risultato di un [traceroute](#).
- Il DNS viene usato da numerose tecnologie in modo poco visibile agli utenti, per organizzare le informazioni necessarie al funzionamento del servizio.
- Accesso veloce ai dati con database in memoria centrale e meccanismo di caching.

Composto da tre componenti principali:

- Spazio dei nomi e informazioni associate (**Resource Record**)
- Nameserver (**Application server che mantiene i dati**)
- Resolver (**client per l'interrogazione del nameserver**)

DNS in dettaglio

Host e router di Internet:

- indirizzo IP (32 bit) - usato per indirizzare i datagrammi
- "nome", ad esempio, www.yahoo.com – usato dagli esseri umani
- "MAC address", ad es. 00:11:22:33:44:55 – usato a livello DL

Domain Name System:

- Database distribuito implementato in una gerarchia di server DNS
- Protocollo a livello di applicazione che consente agli host, ai router e ai server DNS di comunicare per risolvere i nomi (tradurre indirizzi/nomi)
 - nota: funzioni critiche di Internet implementate come protocollo a livello di applicazione
 - complessità nelle parti periferiche della rete
 - Basato sul modello client / server.

Servizi DNS

- Traduzione degli hostname in indirizzi IP
- Host aliasing
 - un host può avere più nomi

- Mail server aliasing
- Distribuzione locale
 - server web replicati: insieme di indirizzi IP per un nome canonico

I motivi che hanno portato a non centralizzare i DNS sono stati tra i tanti la nascita di un singolo punto di guasto, il problema relativo al volume di traffico, se centralizzato sarebbe distante per molti host e la manutenzione sarebbe problematica.

Inoltre un database centralizzato su un singolo server DNS non è scalabile!!

I server DNS effettuano gli zone transfer usando il protocollo di trasporto TCP e la porta 53. Questa modalità viene usata anche quando una query ha una risposta molto lunga.

Nomi DNS

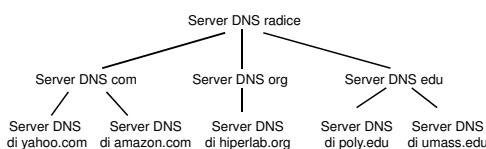
Quando un nome di dominio termina con il carattere **Punto [.]**, il nome viene detto *completo* (FQDN). es: ftp.univaq.it Tale definizione specifica che solo i nomi completi possono corrispondere a un indirizzo IP.

Quando un nome di un dominio non è *completo*, viene detto *parziale* (PQDN) es: informatica Tali nomi non corrispondono ad un IP, ma vengono utilizzati in relazione ad un dominio per esser trasformati in FQDN e quindi in IP.

informatica + univaq.it = informatica.univaq.it => 193.204.130.2

HOSTNAME => INDIRIZZO IP

Database distribuiti e gerarchici



Il client vuole l'IP di www.amazon.com e quindi si avranno i seguenti passaggi, contattare il server radice per trovare il server DNS **com**, interrogare il server DNS **com** per ottenere il server DNS amazon.com e infine interrogare il server DNS amazon.com per ottenerne l'indirizzo ip.

I server DNS sono distribuiti per il mondo e organizzati in maniera gerarchica e per questo esistono 3 classi di server DNS:

1. Server Radice

Ne esistono 13 in internet, sono replicati e non sono singoli. Vengono contattati da un server DNS locale dopo di che contatta un server DNS autorizzato se non conosce la mappatura, ottiene la mappatura e restituisce la mappatura al server DNS locale.

2. Server TLD

Top-level domain, si occupano dei domini di *alto livello* con, org, net, ecc e di tutti i *domini locali* di alto livello, quali uk, it, jp ecc.

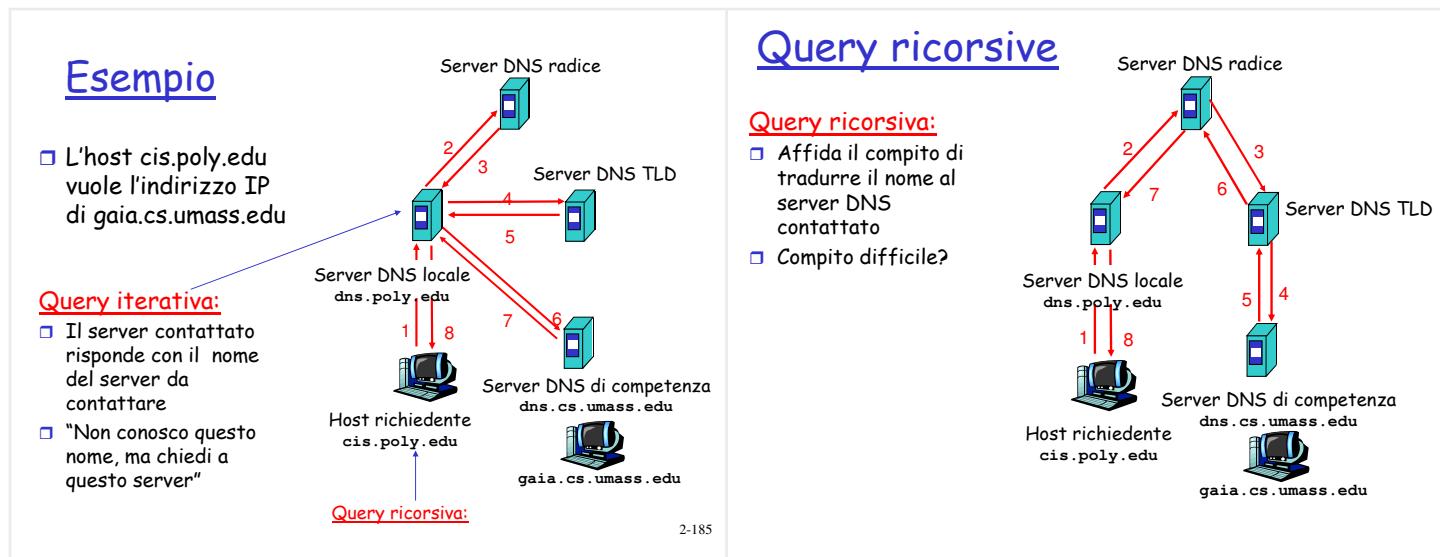
3. Server di Competenza

Authoritative server, ogni organizzazione dotata di un host internet pubblicamente accessibile (server web, di posta) deve fornire i record DNS di pubblico dominio che mappano i nomi di tali host in indirizzi IP. Possono esser mantenuti dall'organizzazione o dal service provider.

Server DNS Locale

Non appartengono alla gerarchia dei server, ma è comunque fondamentale nell'architettura DNS. Ciascun ISP (università, società, ISP residenziale) ha un server DNS locale detto anche *default name server*.

Quando un host effettua una richiesta DNS, la query viene inviata al suo server DNS locale il quale opera da proxy e inoltra la query in un gerarchia di server DNS.



Name server locale definisce che i resolver sono configurati con l'indirizzo IP di un Name Server Locale (solitamente nella stessa rete) e le richieste fatte dal resolver al NS locale sono ricorsive: egli si aspetta che il NS locale gli esaudisca la richiesta.

Caching e aggiornamento record

Una volta che un server DNS impara la mappatura, la mette nella **cache**, nella quale le informazioni vengono invalidate dopo un certo periodo di tempo e tipicamente un server DNS locale memorizza anche l'indirizzi IP dei server TDL cio' diminuisce il carico di lavoro dei server Radice.

Record DNS

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

identificati con il nome di *record di risorsa RR*.

```
$TTL      43200
@           IN      SOA     ns.mesys.it.
hostmaster.mesys.it. (
                      2002053101 ; serial
                      86400 ; refresh
                      3600 ; retry
                      604800 ; expire
                      86400 ; default_ttl
)
@           IN      MX      5      mail.mesys.it.
@           IN      NS      ns.mesys.it.
@           IN      NS      dns2.nic.it.
localhost   IN      A       127.0.0.1
ns          IN      A       151.4.83.2
ns1         IN      A       151.4.83.3
mail        IN      A       151.4.83.2
www         IN      CNAME   turtle.mesys.it.
ftp          IN      CNAME   dolphin.mesys.it.
```

• **Type=A** Indica la corrispondenza tra un nome ed uno (o più) indirizzi IP

name è il nome dell'host

value è l'indirizzo IP

realy1.bar.foo.com , 145.37.93.126, A)

• **Type=NS** utilizzato per indicare quali siano i server DNS autorevoli per un certo dominio, ovvero per delegarne la gestione.

name è il dominio (ad esempio foo.com)

value è il nome dell'host del server di competenza di questo dominio

(foo.com , dns.foo.com , NS)

- **Type=CNAME** Sono usati per creare un alias, ovvero per fare in modo che lo stesso calcolatore sia noto con più nomi. Uno degli utilizzi di questo tipo di record consiste nell'attribuire ad un host che offre più servizi un nome per ciascun servizio. In

questo modo, i servizi possono poi essere spostati su altri host senza dover riconfigurare i client, ma modificando solo il DNS.

name è il nome alias di qualche nome “canonico” (nome vero) www.ibm.com è in realtà servereast.backup2.ibm.com
value è il nome canonico
(*foo.com , realy1.bar.foo.com, CNAME*)

- **Type=MX (Mail eXchange) indica a quali server debba essere inviata la posta elettronica per un certo dominio.**

value è il nome del server di posta associato a name
(*foo.com , mial.bar..foo.com, MX*)

Messaggi DNS

Protocollo DNS: domande (query) e messaggi di risposta entrambi con lo stesso formato.

Identificazione	Flag	12 byte
Numero di domande	Numero di RR di risposta	
Numero di RR autorevoli	Numero di RR addizionali	
Domande (numero variabile di domande)		
Risposte (numero variabile di record di risorsa)		
Competenza (numero variabile di record di risorsa)		
Informazioni aggiuntive (numero variabile di record di risorsa)		

Intestazione del messaggio:

- Identificazione: numero di 16 bit per la domanda, la risposta alla domanda usa lo stesso numero.
- Flag

Domanda o Risposta
Richiesta di ricorsione
Ricorsione disponibile
Risposta di competenza

I vari campi

- Domande: campo per il nome richiesto e il tipo di domanda
- Risposte: RR alla risposta alla domanda
- Record per i server di competenza
- Informazioni extra

Inserire record nel database DNS

Esempio: abbiamo appena avviato la nuova società “Network Utopia”

1. Registriamo il nome networkutopia.com presso registrar (ad esempio, Network Solutions)
 - 1.1. Forniamo a registrar i nomi e gli indirizzi IP dei server DNS di competenza (primario e secondario)
 - 1.2. Registrar inserisce due RR nel server TLD com: (networkutopia.com, dns1.networkutopia.com, NS) (dns1.networkutopia.com, 212.212.212.1, A)
2. Inseriamo nel server di competenza un record tipo A per www.networkutopia.com e un record tipo MX per networkutopia.com

Risoluzione Inversa

Risalire al nome del dominio noto l’IP ! consiste nell’ottenere l’indirizzo IP, es. 13.14.15.16, esso viene trasformato in un nome di dominio: 16.15.14.13.in-addr.arpa.

E poi viene effettuata una normale risoluzione di questo nome di dominio. Questa risoluzione anziché fornire un IP fornisce un nome di dominio che è quello relativo all’IP di partenza.

Approfondimenti

Sicurezza



- **man-in-the-middle:** le query vengono modificate in modo da reindirizzare tutto o parte del traffico verso un server che agisce da **proxy** trasparente, intercettando il traffico degli utenti. Questo permette il monitoraggio del traffico degli utenti, e quindi anche il furto di informazioni sensibili e/o credenziali. Una soluzione open source è [ArpON](#) "ARP handler inspection". ArpON è un demone portabile che rende il protocollo ARP sicuro contro attacchi Man in The Middle (MITM) attraverso tecniche ARP Spoofing, ARP Cache Poisoning, ARP Poison Routing (APR). Blocca anche attacchi derivati quali Sniffing, Hijacking, Injection, Filtering come: DHCP Spoofing, DNS Spoofing, WEB Spoofing, Session Hijacking e SSL/TLS Hijacking & co attacks.
- **redirezione degli errori:** alle query per nomi inesistenti viene risposto con l'indirizzo IP di un server, che tipicamente ospita un motore di ricerca e tenta di aiutare gli utenti a trovare il sito cercato.

Realizzazione

- I DNS implementano uno spazio gerarchico dei nomi, per permettere che parti di uno spazio dei nomi, conosciute come "zone", possano essere delegate da un [name server](#) ad un altro name server che si trova più in basso nella gerarchia.
- I nomi di dominio sono soggetti a determinate restrizioni: per esempio ogni parte del nome (quella cioè limitata dai punti nel nome) non può superare i 63 caratteri e il nome complessivo non può superare i 255 caratteri.
- I nomi di dominio sono anche limitati ad un sottoinsieme di caratteri [ASCII](#); in questo modo si impedisce di scrivere nomi e parole con caratteri che non tutti hanno sulla propria tastiera. Per superare questa limitazione, il sistema di [IDNA](#) è basato sul modello [Punycode](#), rileva stringhe [Unicode](#) in un insieme di caratteri DNS validi, venne approvato dall'[ICANN](#) e adottato da alcuni [registri](#).

Ricorsione

- In generale, per ottenere la risoluzione di un nome è necessario partire dalla radice, interrogare uno dei root server nel dominio di primo livello, ottenere il server che lo gestisce, interrogarlo nel dominio di secondo livello, fino a raggiungere il server autorevole per il nome desiderato. Questa tecnica è detta "ricorsione".
- [\[modifica\]](#)

Caching

- Alcuni server si prestano ad effettuare query ricorsive per conto di alcuni client. Una volta che hanno ottenuto una risposta, memorizzano in una [cache](#) tutte le informazioni che hanno imparato, fino alla loro scadenza. Alcune implementazioni del servizio DNS permettono di realizzare i cosiddetti servers caching only, ovvero privi di database proprio, ma utili per reindirizzare ad un server autorevole le query di risoluzione. Tale caratteristica è utile soprattutto quando la risoluzione deve essere effettuata attraverso collegamenti lenti (con velocità inferiore a 500 kbps) o [firewall](#).

Domande e Problemi

1. Elencare 5 applicazioni internet non proprietarie e i protocolli a livello applicazione che utilizzano?
2. Qual'e' la differenza tra un'architettura di rete e un'architettura di applicazione?

Domande

3. In una sessione di comunicazione tra una coppia di processi, chi e' il client e chi il server?

In una sessione di comunicazione viene identificato come client il processo che inizia la sessione ovvero il primo che avvia la comunicazione. Il server invece e' il processo che viene interpellato e il quale deve rispondere alle "Richieste" del client.

4. Per le applicazioni P2P siete d'accordo con l'affermazione "Non esiste la nozione di lato client e server in una sessione di comunicazione?"

Sì, perché in realtà ogni host della comunicazione esegue sia operazioni da client che da server, infatti oltre che richiedere un determinato servizio ad altri, nello stesso momento si mette lui a disposizione come fornitore di altri servizi

5. Quali informazioni usa un processo in esecuzione su un host per identificare un processo di un altro host?

Le informazioni principali utilizzate sono due:

- l'indirizzo IP
- socket del processo

6. Supponete di voler fare una transizione da un client remoto a un server il piu' velocemente possibile. userete TCP o UDP? perche?

UDP, dato che tra i due livelli di trasporto e' il piu' leggero (non connection oriented, non ha meccanismi di controllo pesanti e costosi)

7. Elencate un'applicazione che richieda di non perdere alcun dato e che contemporaneamente molto sensibile alla tempistica?

Le comunicazioni bancarie possiedono entrambe le caratteristiche.

8. Confrontate UDP e TCP in base alle 4 grandi classi di servizi messi a disposizione da un servizio di trasporto

/	trasferimento dati affidabile	Throughput	Temporizzazione	Sicurezza
TCP	si			si
UDP	no			minima

9. mmmm

10. Cosa si intende per protocollo di handshaking?

Con il termine handshaking intendiamo quel protocollo utilizzato quando vogliamo instaurare in modo affidabile una connessione tra due host. Generalmente utilizzata da TCP e comunemente chiamato 3 way handshaking dato che esso prevede lo scambio di tre messaggi, syn , syn/ack , ack. Questo protocollo ci permette di creare correttamente una connessione.

11. Perche' HTTP,FTP,SMTP,POP3 e IMAP girano su TCP e non UDP?

Il motivo principale e' che TCP e' un protocollo di trasporto affidabile e in secondo luogo i protocolli del livello applicazione elencati privilegiano una trasmissione affidabile ad una piu' veloce ma meno affidabile.

In modo da poter delegare tutti i controlli a TCP e procedere nelle loro attivita'.

12. Come descriveresti il meccanismo dei cookies di un sito di e-commerce?

Il meccanismo dei cookies e' un meccanismo che prevede uno scambio d'informazioni tra il client HTTP e il server per creare/utilizzare il codice cookies. in particolare avremo che alla prima richiesta effettuata dal client (presupponendo che si sia loggato) ad esso gli verrà assegnato un cookie (set-cookie) il quale sarà riutilizzato dal client (browser) ad ogni successiva richiesta. tale tecnica insieme al meccanismo delle sessioni permette di mantenere "uno stato" sul lato client della comunicazione. ricordiamo infatti che HTTP è senza stato!!

13. Perche' si dice che FTP invia informazioni di controllo "fuori banda"?

viene così definita in quanto FTP prevede due connessioni TCP una sulla porta numero 20 la cosiddetta connessione di controllo e una sulla porta 21 connessione dati.

In particolar modo sulla connessione di controllo vengono scambiati essenzialmente messaggi di controllo/comandi i quali verranno eseguiti sulla connessione dati e per la presenza di due connessioni che FTP viene definito in tale modo/

Problemi

(6) Qual'e' il meccanismo di segnalazione usato tra client e server per indicare che una connessione persistente sta' per essere chiusa??
Tale chiusura puo' essere segnalata dal client, dal server o da entrambi??

L'implementazione della chiusura non e' specificata nel protocollo HTTP/1.1, ovvero essa puo' esser iniziata dal client o dal server, in particolare vi sono dei momenti in cui e' obbligatorio non chiudere la connessione, ma per il resto entrambi possono segnalarla/iniziarla per primi.



Il meccanismo classico utilizzato per la segnalazione della chiusura della connessione e' porre nell'Header tra le righe di intestazioni il campo: "Connection: Close"

Inoltre il server puo' chiudere la connessione per due motivi:

- a) allo scadere di un timeout, applicato sul tempo totale di connessione o sul tempo di inattività di una connessione;
- b) allo scadere di un numero massimo di richieste da servire sulla stessa connessione.

(7) Supponendo di dover ottenere una pagina tramite l'url relativo. Per ottenere l'IP relativo, che non si trova in cache, e' necessari una ricerca DNS. Supponendo di dover effettuare un numero di visite pari a n server DNS prima di ricevere l'indirizzo ip del dns. Le visite successive incorrono in RTT di RTT1...RTTn. Supponendo inoltre che l'RTT0 sia il tempo tra l'host locale e il server che contiene l'oggetto. Quanto tempo intercorre tra il click sull'url della pagina web e la ricezione dell'oggetto? Supponendo che il tempo di trasmissione sia nullo quanto intercorre?

Dovendo effettuare n visite dei server DNS dobbiamo sommare i ritardi relativi, una volta ottenuto l'ip del dns dobbiamo effettuare la richiesta HTTP per ottenere l'oggetto ovvero andar a sommare RTT0:

RTT0 +RTT1 + + RTTn a cui andiamo a sommare il tempo di trasmissione dell'oggetto per rispondere alla prima domanda e non lo consideriamo per rispondere alla seconda.

(8) In riferimento al problema 6 supponiamo che i file HTML referenzi tre oggetti molto piccoli sullo stesso server. Ignorando il tempo di trasmissione, quanto tempo intercorre con

- a. HTTP non persistente e senza connessioni TCP parallele
- b. HTTP non persistente e con connessioni TCP parallele
- c. HTTP persistente

Per quanto riguarda il punto a. in questo caso siamo nell'approccio classico dell'HTTP/1.0 ovvero una richiesta/risposta per connessione TCP in modo seriale. In questo caso per ogni singolo oggetto avremo :

3-wayhandshakedelTCP

il client invia il messaggio di richiesta HTTP sulla connessione

Il server invia il messaggio di risposta HTTP sulla connessione

la chiusura della connessione TCP

Cio' comporta un Overhead per l'insaturazione e l'abbattimento della connessione TCP, per ogni trasferimento di risorsa infatti si hanno 2 round trip time in piu'.

A cio' va aggiunto l'Overhead per il meccanismo di slow start di TCP, infatti la finestra ha dimensione pari a 1 all'inizio di ogni nuova connessione TCP.

In conclusione per il punto a avremo $3RTT * (\text{numero oggetti}) = 12 RTT$

Per quanto riguarda il punto b. si differenzia dal punto a. avendo la possibilità di connessioni parallele. Che implementa una soluzione parziale al problema.

In questo caso abbiamo un'apertura simultanea di più connessioni TCP, una per ogni oggetto richiesto.

Le problematiche di questo tipo di approccio sono:

Aumento congestione sulla rete

Il server riesce a servire un numero minore di client diversi

La richiesta delle pagine è interrotta dal client

Non è garantita la riduzione della latenza dato che ogni richiesta è indipendente

infine nella rfc 2616 è specificato che un client non dovrebbe avere più di due connessioni parallele per uno stesso server

In conclusione avremo un ritardo di 3 RTT per la richiesta del primo oggetto (pagina HTML) a cui andremo a sommare altri 3RTT dovuti alle richieste effettuate in parallelo per i 3 oggetti

Per quanto riguarda il punto c. essa si basa sull'utilizzo di connessioni persistenti ovvero si avranno trasmissioni HTTP multipli uno dopo l'altro, in modo seriale.

In questo modo si ammortizza l'overhead del tcp, ma si mantiene più a lungo lo stato TCP sul server (time_wait)

In conclusione per il punto c avremo un ritardo di $RTT(3\text{-wayhandshakedelTCP}) + 4RTT$ (pagina + oggetti) e $RTT(\text{chiusura connessione TCP})$.

(9) E' un bordello!!!!

(10) Idem

(11) idem

(12) Qual'e' la differenza tra MAIL FROM: in SMTP e From: nel messaggio di posta elettronica stesso?

La differenza principale e' che fanno parte di due intestazioni differenti, una propria di SMTP e una definita dall'rfc 822 relativa ai messaggi di posta. E quindi si ha che il MAIL FROM fa parte dell'handshaking SMTP tra la comunicazione tra il client SMTP e il sever SMTP per lo scambio del messaggio , mentre il From: presente nel messaggio serve per specificare l'email del mittente.

(13) Quel e' lo scopo del comando POP3 UIDL?

Lo scopo di tale comando e' quello d'individuare l'id-univoco di ogni messaggio presente nella mailbox o di un particolare messaggio passando come parametro il numero sequenziale. Nelle rfc viene elencato tra i comandi "facoltativi ma consigliati", inoltre tale id-univoco non e' memorizzato nella mailbox ma dev'essere calcolato come un hash del messaggio e quindi il client che utilizza tale comando deve saper trattare i casi in cui messaggi differenti abbiano lo stesso id-univoco. Inoltre l'id-univoco e' una stringa consistente da 1 a 70 caratteri che logicamente individua in modo univoco un messaggio in una mailbox.

(14) Produrre una transizione di una sessione POP in cui andiamo a scaricare due messaggi, in modalita' "scarica-e-mantieni"

```
C: list
S: 1 456
S: 2 566
S: .
C: retr 1
S: bla bla....
S: ..... bla
S: .
C: ret 2
S: bla bla....
S: ..... bla
S: .
C: quit
S: +ok ....
```

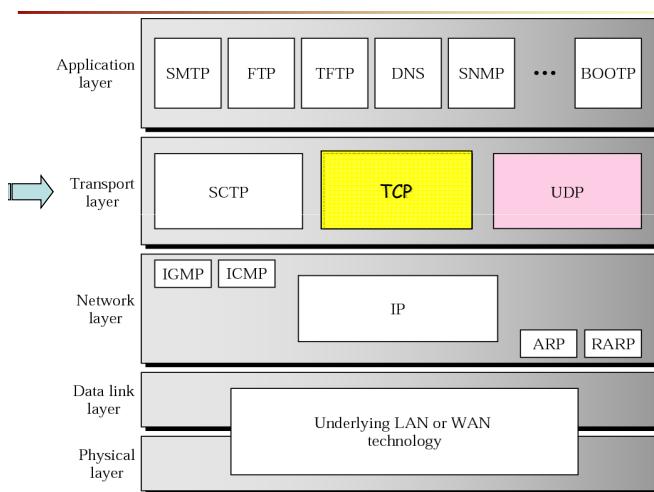
Capitolo 3 Livello di trasporto

Parametri dal livello applicazione al livello di trasporto

Servizi e Protocolli di Trasporto

Forniscono la **comunicazione logica tra processi applicativi di host differenti**. I protocolli di trasporto vengono eseguiti nei sistemi terminali.

Più protocolli di trasporto sono a disposizione delle applicazioni come per Internet si hanno UDP e TCP.



Relazione tra livello di trasporto e livello di rete

Il livello di rete offre una comunicazione logica tra host, mentre il livello di trasporto offre una comunicazione logica tra processi che girano su host differenti, essa si basa sui servizi del livello di rete.

Protocollo del livello di trasporto in Internet

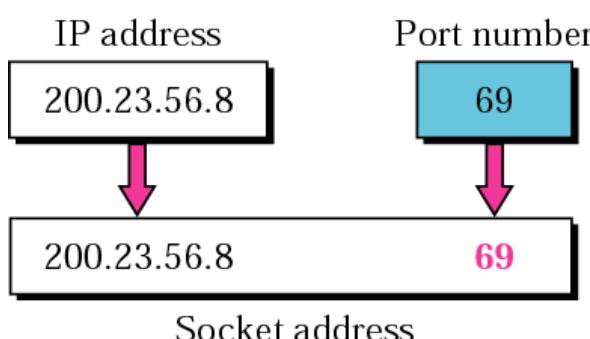
Transmission Control Protocol, caratterizzato da una consegna garantita e in ordine, controllo di congestione, controllo di flusso e setup della connessione. “*TCP converte il sistema inaffidabile, IP, a livello di rete in un sistema affidabile a livello trasporto*”

User Datagram Protocol, definito come un'estensione senza fronzoli del servizio “best-effort” di IP. “*UDP invece estende l'inaffidabilità di IP a livello trasporto*”.

I **servizi non offerti** sono la garanzia su ritardi e la garanzia su ampiezza di banda, dato che essi non sono un servizio offerto da IP.

Multiplexing e demultiplexing

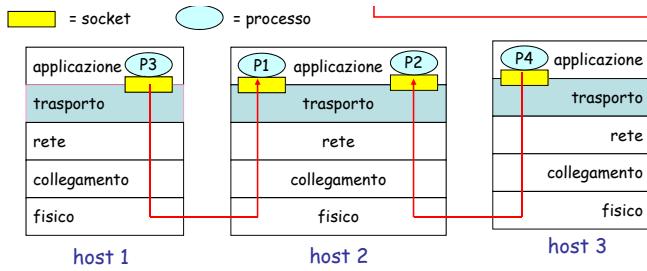
Il passaggio da consegna host-to-host a consegna process-to-process viene detto multiplexing e demultiplexing a livello di trasporto.



Un processo, come parte di una applicazione di rete, può presentare una o più **socket**: porte attraverso le quali i dati fluiscono dal processo alla rete e viceversa

Il livello di trasporto nell'host ricevente non trasferisce i dati direttamente al processo ma ad una socket intermedia

Ogni socket è identificata univocamente dal servizio, dall'indirizzo IP e dal numero di porta



Demultiplexing

Esamina i campi del segmento e li indirizza alla socket appropriata.

Multiplexing

Raccoglie i dati, li incapsula con un preambolo e invia il segmento al livello di rete.



Struttura del segmento TCP/UDP

Trasporto senza connessione: UDP

User Datagram Protocol [RFC 786, id 7] E' un protocollo di trasporto "senza fronzoli" con un servizio di consegna "best effort", i segmenti UDP possono essere perduti e consegnati fuori sequenza all'applicazione.

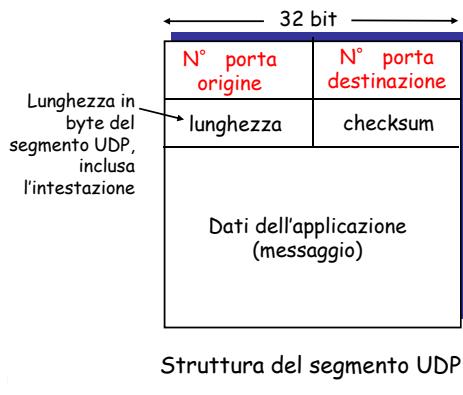
Rappresenta il minimo che un protocollo di trasporto può offrire. Infatti offre soltanto il Multiplexing, Demultiplexing e una piccola forma di error detection.

Esso non fa altro che prendere i messaggi dall'applicazione, ci aggiunge il suo preambolo e passa il segmento al livello sottostante. UDP non ha handshaking es DNS.

Si usa UDP perché:

- e' veloce, alcune applicazioni possono tollerare una piccola perdita di dati. Non
- nessuna connessione, Non ha ritardi nello stabilire la connessione e quindi invia subito.
- semplice, non ha nessuno stato di connessione nel mittente e nel destinatario e quindi un server puo' supportare piu' client.
- intestazioni di pacchetto più corte (8 byte).
- UDP viene spesso utilizzato per il trasporto di dati di gestione, dato che molto spesso questi dati fluiscono nel momento in cui la rete e' in stato di stress.

Protocollo/Servizio	Porta	Descrizione
Echo	7	Risponde con gli stessi dati arrivati
Elimina	9	Elimina i datagram
Utenti	11	Fornisce il numero di utenti collegati
Daytime	13	Fornisce la data e l'ora
Quote	17	Fornisce la massima del giorno
Chargen	19	Risponde con una stringa di caratteri
DNS	53	Server DNS
BOOTP	67	Server BOOTP
BOOTP	68	Client BOOTP
TFTP	69	Trivial File Transfer Protocol
RPC	111	Server Remote Procedure Call
NTP	123	Network Time Protocol
SNMP	161	Server Simple Mail Network Protocol
SNMP	162	Porta di servizio SNMP



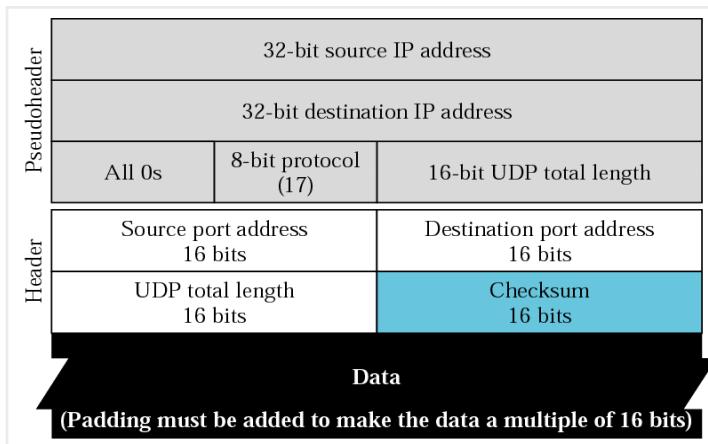
- **Source port [16 bit]** - Identifica il numero di porta sull'host del mittente del datagramma;
- **Destination port [16 bit]** - Identifica il numero di porta sull'host del destinatario del datagramma;
- **Length [16 bit]** - contiene la lunghezza totale in bytes del datagramma UDP (header+dati);
- **Checksum [16 bit]** - contiene il codice di controllo del datagramma (header+dati+pseudo-header, quest'ultimo comprendente gli indirizzi IP di sorgente e destinazione). L'algoritmo di calcolo è definito nell' RFC del protocollo;
- **Data** - contiene i dati del datagramma

Error control (Checksum UDP)

Ha come obiettivo quello di rilevare gli "errori" (bit alterati) nel segmento trasmesso.

Il mittente, tratta il contenuto del segmento come **una sequenza di interi da 16 bit**, il checksum è la somma (complemento a 1) dei contenuti del segmento e dopo aver calcolato tutto ciò lo pone nel campo del preambolo UDP.

Il destinatario, calcola la checksum del segmento ricevuto e controlla se il checksum coincide con quello del preambolo.



Nota

– Quando si sommano i numeri, un riporto dal bit più significativo deve essere sommato al risultato

Esempio: sommare due interi da 16 bit

a capo	$ \begin{array}{r} 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 0 \\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1 \end{array} $
	$ \begin{array}{r} \textcircled{1} \ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1 \end{array} $
	somma
	checksum

Perché UDP calcola la checksum

Anzitutto UDP non se ne fa nulla della informazione di Checksum, alcune implementazioni scartano il segmento errato, altre si limitano a segnalare la cosa al livello applicativo

Anche se i livelli sottostanti controllano gli errori, si potrebbero verificare errori all'interno dei router ed il principio punto-punto secondo il quale determinate funzionalità devono essere implementate su base punto-punto

Approfondimento

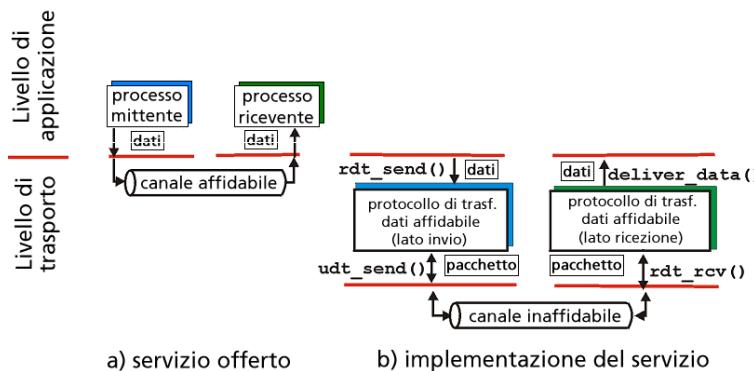
A differenza del TCP l'UDP è un protocollo di tipo connectionless, inoltre non gestisce il riordinamento dei pacchetti né la ritrasmissione di quelli persi, ed è perciò generalmente considerato di minore affidabilità. È in compenso molto rapido ed efficiente per le applicazioni "leggere" o time-sensitive. Ad esempio, è usato spesso per la trasmissione di informazioni audio o video. Dato che le applicazioni in tempo reale spesso richiedono un ritmo minimo di spedizione, non vogliono ritardare eccessivamente la trasmissione dei pacchetti e possono tollerare qualche perdita di dati, il modello di servizio TCP può non essere particolarmente adatto alle loro caratteristiche. L'UDP fornisce soltanto i servizi basilari del livello di trasporto, ovvero:

- multiplazione delle connessioni, ottenuta attraverso il meccanismo delle porte
 - verifica degli errori mediante una checksum, inserita in un campo dell'intestazione del pacchetto.
- mentre TCP garantisce anche il trasferimento affidabile dei dati, il controllo di flusso e il controllo di congestione.

L'UDP è un protocollo stateless, ovvero non tiene nota dello stato della connessione, dunque ha rispetto al TCP informazioni in meno da memorizzare. Un server dedicato ad una particolare applicazione che scelga UDP come protocollo di trasporto può supportare molti più client attivi.

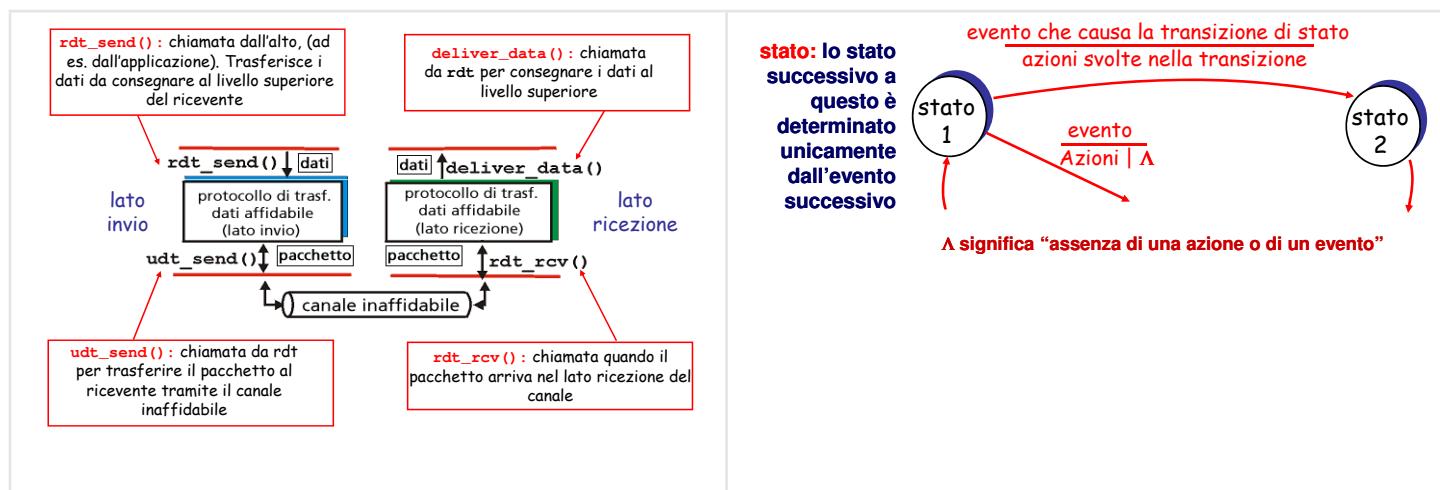
Principi del trasferimento Affidabile

il compito dei **protocolli di trasferimento dati affidabile** è l'implementazione di questa astrazione del servizio.



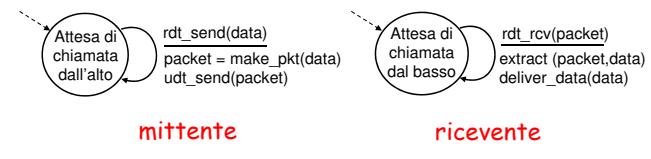
Le caratteristiche del canale inaffidabile determinano la complessità del protocollo di trasferimento dati affidabile (reliable data transfer o rdt, udt_send). Dove **rdt** sta' per **reliable data transfer** e **udt** sta' per **unreliable data transfer**.

Trasferimento dati affidabile



Considereremo soltanto i trasferimenti dati unidirezionali, ma le informazioni di controllo fluiranno in entrambe le direzioni.

Rdt 1.0: trasferimento affidabile su canale affidabile



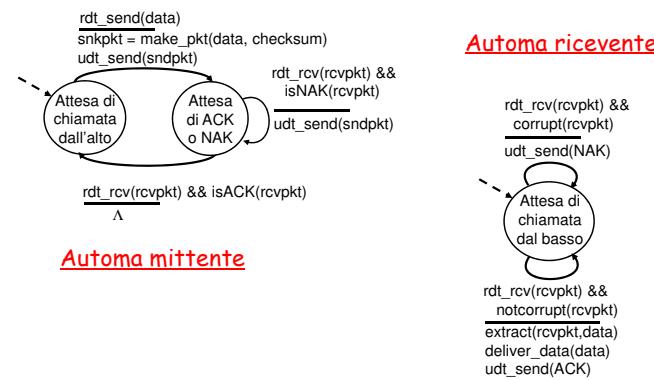
La freccia tratteggiata individua lo stato iniziale
In questo canale nulla può andare storto e mittente e ricevente operano alla stessa velocità per cui non c'è esigenza di controllo di flusso

Canale sottostante perfettamente affidabile, Nessun errore nei bit ed Nessuna perdita di pacchetti.

Automa distinto per il mittente e per il ricevente:

- il mittente invia i dati nel canale sottostante
 - il ricevente legge i dati dal canale sottostante

Rdt 2.0: canale con errori nei bit



Il canale sottostante potrebbe confondere i bit nei pacchetti (l'ordine è ancora garantito) andando così a realizzare un metodo per rilevare gli errori nei bit (checksum).

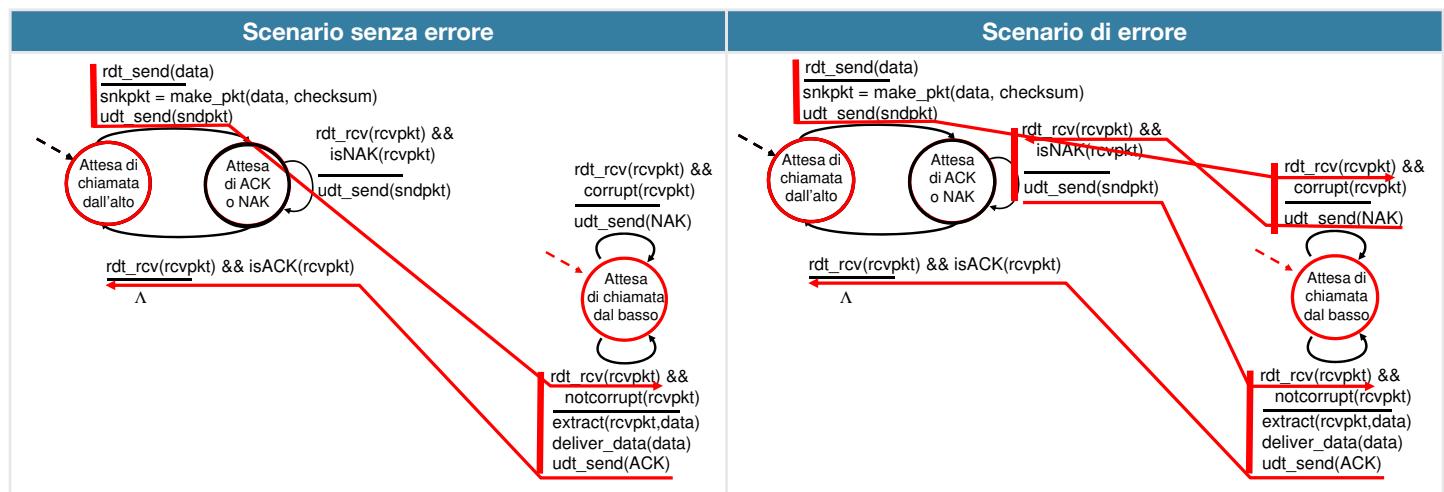
Domanda: come correggere gli errori ?

- **Notifica positiva (ACK):** il ricevente comunica espressamente al mittente che il pacchetto ricevuto è corretto
 - **Notifica negativa (NAK):** il ricevente comunica espressamente al mittente che il pacchetto contiene errori
 - Il mittente **rtrasmette** il pacchetto se riceve un NAK

Questi protocolli necessitano di nuovi meccanismi in rdt2.0 (oltre a quelli di rdt1.0):

- Rilevamento di errore
 - Le tecniche di rilevamento richiedono la trasmissione di bits extra
 - Feedback del destinatario: messaggi di controllo (ACK, NAK) ricevente => mittente
 - Ritrasmissioni
 - Ogni pacchetto ricevuto con errori sarà ritrasmesso
 - Analogia della dettatura al telefono

Protocolli ARQ (Automatic Repeat reQuest): nel networking i protocolli basati su ritrasmissioni si chiamano protocolli ARQ.



Da notare:

Quando il mittente e' in attesa di un ACK/NAK non puo accettare dati dal livello superiore:

- Non puo aver luogo l'evento rdt_send() senza la ricezione di un ACK che permette al mittente di cambiare stato
- Il mittente non invierà nuovi dati finchè non è certo che il ricevente abbia ricevuto correttamente il pacchetto corrente
- Rdt 2.0 è un protocollo di tipo **stop-and-wait**
- L'automa lato rx ha un solo stato

Difetto di Rdt 2.0

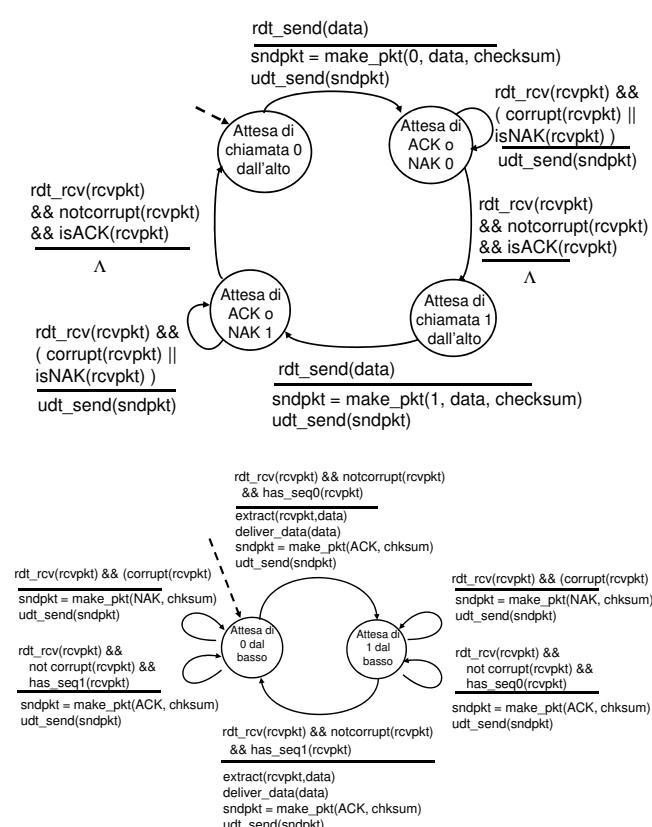
Che cosa accade se i pacchetti ack/nack sono danneggiati?

Il mittente non sa se il destinatario abbia ricevuto correttamente o meno l'ultimo pacchetto trasmesso e per risolvere la rispedizione non basta perché si creerebbero dei duplicati.

Gestione dei duplicati:

Il mittente ritrasmette il pacchetto corrente se ACK/NAK e' alterato, inoltre il mittente aggiunge un *numero di sequenza* a ogni pacchetto e il ricevente dovrà scartare i pacchetti duplicati. Da notare che in questo modello senza perdita basta un solo bit per i numeri di sequenza.

Rdt 2.1



Il mittente:

Gestisce gli ACK/NAK alterati andando ad aggiungere il numero di sequenza al pacchetto

- Saranno sufficienti due numeri di sequenza (0,1). Perché nel caso di una ritrasmissione il numero di sequenza del pacchetto ha lo stesso numero di sequenza del pacchetto appena ricevuto, in caso di invio di un pacchetto non' trasmesso il numero di sequenza sara' diverso.

- Deve controllare se gli ACK/NAK sono danneggiati, ma lavorando con un canale senza perdita di pacchetti, si ha che essi dovendo indicare il numero di sequenza del pacchetto di cui rappresentano la modifica ed inoltre il mittente che un pacchetto ricevuto di tipo ACK/NAK e' stato generato come risposta al pacchetto dati trasmesso più di recente.

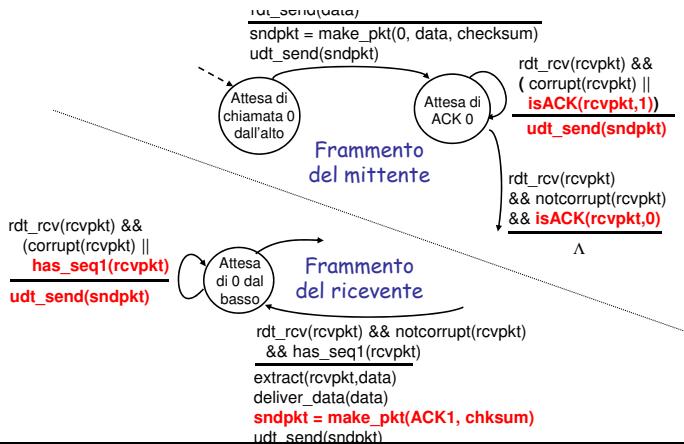
- Il doppio di stati: lo stato deve "ricordarsi" se il pacchetto "corrente" ha numero di sequenza 0 o 1

Il ricevente:

Gestisce gli ACK/NAK alterati andando a controllare se il pacchetto ricevuto e' duplicato, lo stato indica se il numero di sequenza previsto e' 0 o 1.

Da notare che il ricevente non puo' sapere se il mittente ha ricevuto il suo ultimo ACK/NAK e se l'ha ricevuto correttamente.

Rdt 2.2: un protocollo senza NAK

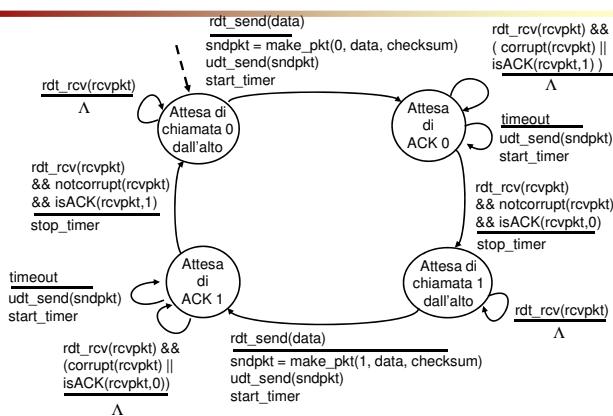


Ha le stesse funzionalità di Rdt 2.1, ma utilizza soltanto gli ACK.

Il destinatario invia un ACK per l'ultimo pacchetto ricevuto correttamente, andando ad includere esplicitamente il numero di sequenza del pacchetto.

La stessa azione del NAK e' ottenuta dalla ricezione del mittente di un ACK duplicato, il quale provoca la ritrasmissione del pacchetto corrente.

Rdt 3.0: canali con errori e perdite



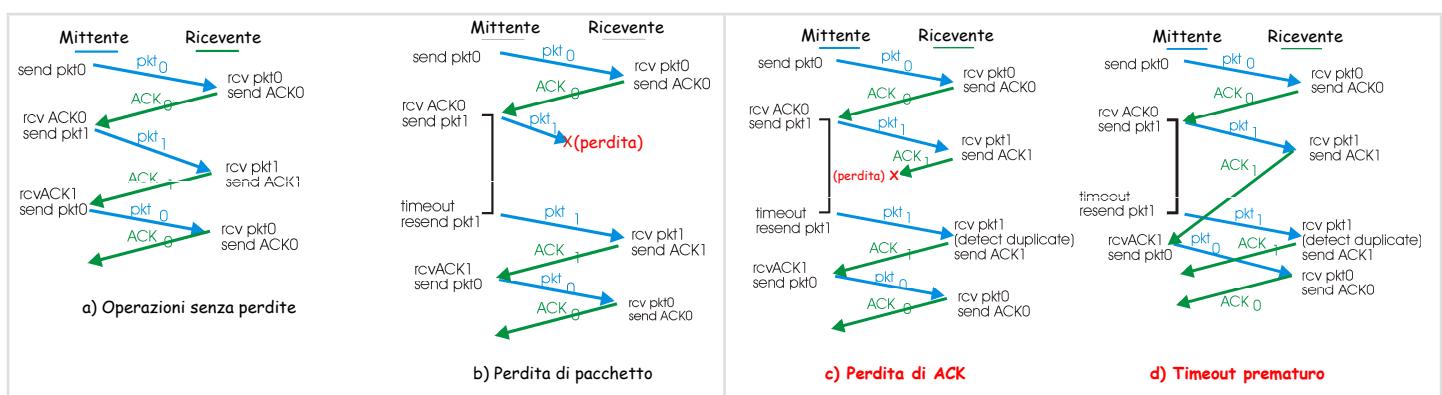
Nuova ipotesi: il canale sottostante può anche perdere i pacchetti dati o ACK. il checksum, numero di sequenza, ACK e ritrasmissioni aiuteranno, ma non saranno sufficienti.

Approccio: il mittente può attendere un ACK per un tempo "ragionevole" e ritrasmettere se non ha ricevuto un ACK in questo periodo.

Prime problematiche:

- se il pacchetto dati o l'ack e' soltanto in ritardo
- la ritrasmissione sarà duplicata , ma l'uso dei numeri di sequenza gestisce già questo caso.

- il destinatario deve specificare il numero di sequenza del pacchetto da riscontrare.
- occorre un contatore (timer) il cui valore non e' di facile stima.



Come possiamo notare B e C sono uguali per il mittente e C e D uguali per il destinatario.

Rdt 3.0 e' anche detto protocollo ad alternanza di bit (alternative bit).

Key features di un protocollo affidabile

Checksum: Per rilevare gli errori; Numeri di sequenza: Per evitare duplicazioni; Contatori/timer: Per ritrasmissione automatica; Ack/Nack: Feedbacks.

Prestazioni di Rdt 3.0: funziona, ma le prestazioni non sono apprezzabili

Esempio: collegamento da 1 Gbps, due hosts uno sulla costa occidentale degli USA e uno su quella orientale (ritardo di propagazione 15 ms), pacchetti di dimensione L= 1000 Bytes:

- Un confronto con un protocollo stop-and-wait, se il mittente comincia a inviare pacchetti a t=0, allora l'ultimo bit entra sul canale sul lato d'invio al tempo $t = L/R$ bps = 8 us.
- Il pacchetto viaggia attraverso il continente il 15 ms
- l'ultimo bit giunge al destinatario a $t=RTT/2 + L/R = 15,008$ ms
- Supponendo ACK di piccole dimensioni e immediati da parte del destinatario, il mittente riceve l'ACK a $t=RTT + L/R = 30,008$ ms
- adesso il mittente puo' spedire nuovamente, ma in 30,008 ms avra trasmesso solo per 0,008 ms Umitt

$$T_{\text{trasm}} = \frac{L \text{ (lunghezza del pacchetto in bit)}}{R \text{ (tasso trasmissivo, bps)}} = \frac{8000 \text{ bit/pacc}}{10^9 \text{ bit/sec}} = 8 \text{ microsec}$$

$$U_{\text{mitt}} = \frac{L/R}{RTT + L/R} = \frac{0,008}{30,008} = 0,00027$$

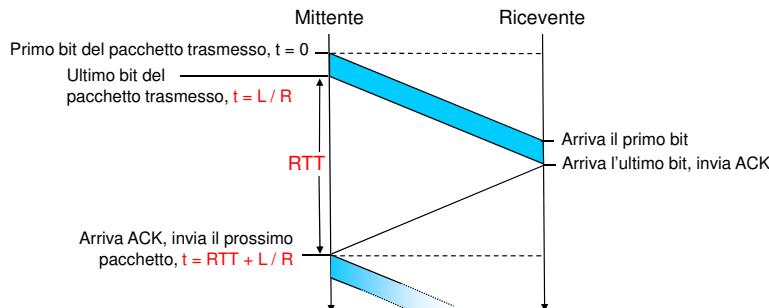
Umitt : utilizzo è la frazione di tempo in cui il mittente è occupato nell'invio di bit

Un pacchetto da 1KB ogni 30 msec => throughput di 33 kB/sec ovvero 267 Kbps in un collegamento da 1 Gbps (1000000 Kbps)

La soluzione e' utilizzare Rdt 3.0 con pipeline.

Il protocollo di rete limita l'uso delle risorse fisiche!

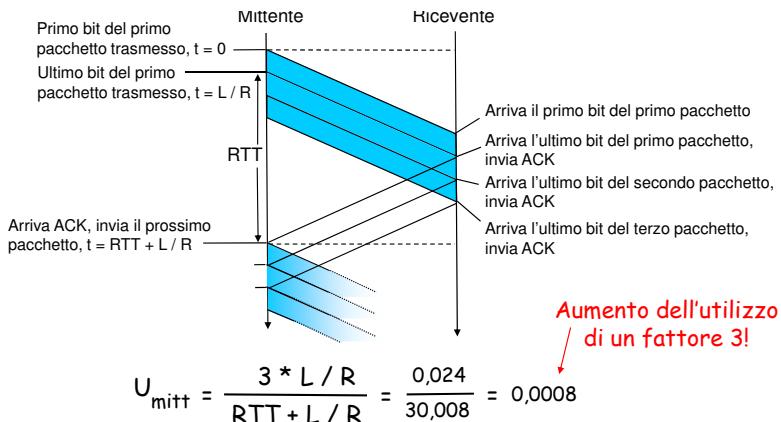
Rdt 3.0: con funzionamento stop-and-wait



$$U_{\text{mitt}} = \frac{L / R}{RTT + L / R} = \frac{0,008}{30,008} = 0,00027 \text{ microsec}$$

Protocolli con pipeline

Pipelining: il mittente ammette più pacchetti in transito, ancora da notificare; ciò implica che l'intervallo dei numeri di sequenza deve essere incrementato e l'implementazione del buffering dei pacchetti presso il mittente e/o ricevente.

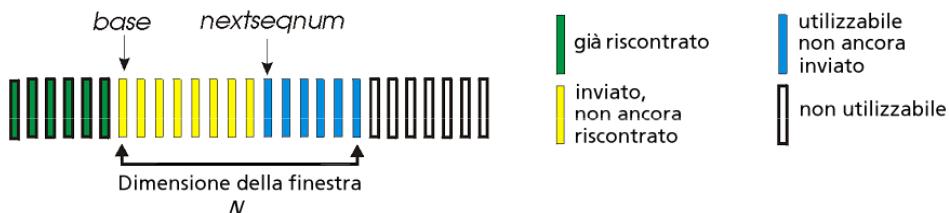


Tale protocollo ha delle conseguenze:

- aumento dell'intervallo dei numeri di sequenza, data che ogni pacchetto dovrà avere un numero di sequenza differente, comprese le ritrasmissioni
- i lati d'invio e di ricezione possono dover bufferizzare più di un pacchetto.
- La quantità di numeri di sequenza necessari e i requisiti di buffering dipendono dal modo in cui il protocollo di trasferimento dati risponde ai pacchetti smarriti, alterati o troppo in ritardo: **Go-Back-N e ripetizione selettiva**.

Go-Back-N

In questo tipo di protocollo, il mittente può trasmettere più pacchetti senza dover attendere la notifica (sfruttano al massimo la banda), ma non può avere più di un dato numero massimo consentito N di pacchetti (se disponibili) in attesa di notifica. Tale protocollo prevede la presenza di un buffer di ricezione e d'invio e la dimensione della finestra sono limitate dalla congestione del flusso. Esso fa uso dei numeri di sequenza, riscontri cumulativi, checksum e operazioni di timeout/ritrasmissione.



Il **numero di sequenza** di un pacchetto viene trasportato in un campo a dimensione fissa dell'intestazione del pacchetto. Tale campo è generalmente di lunghezza pari a **k bit**, fornendo così un'intervallo di numeri da 0 a $(2^k - 1)$. Facendo pensare ad i numeri di sequenza come un'insieme circolare di 2^k elementi. La finestra contiene fino a N pacchetti consecutivi non riscontrati.

Base = numero di sequenza del pacchetto più vecchio che non sia stato ancora riscontrato;

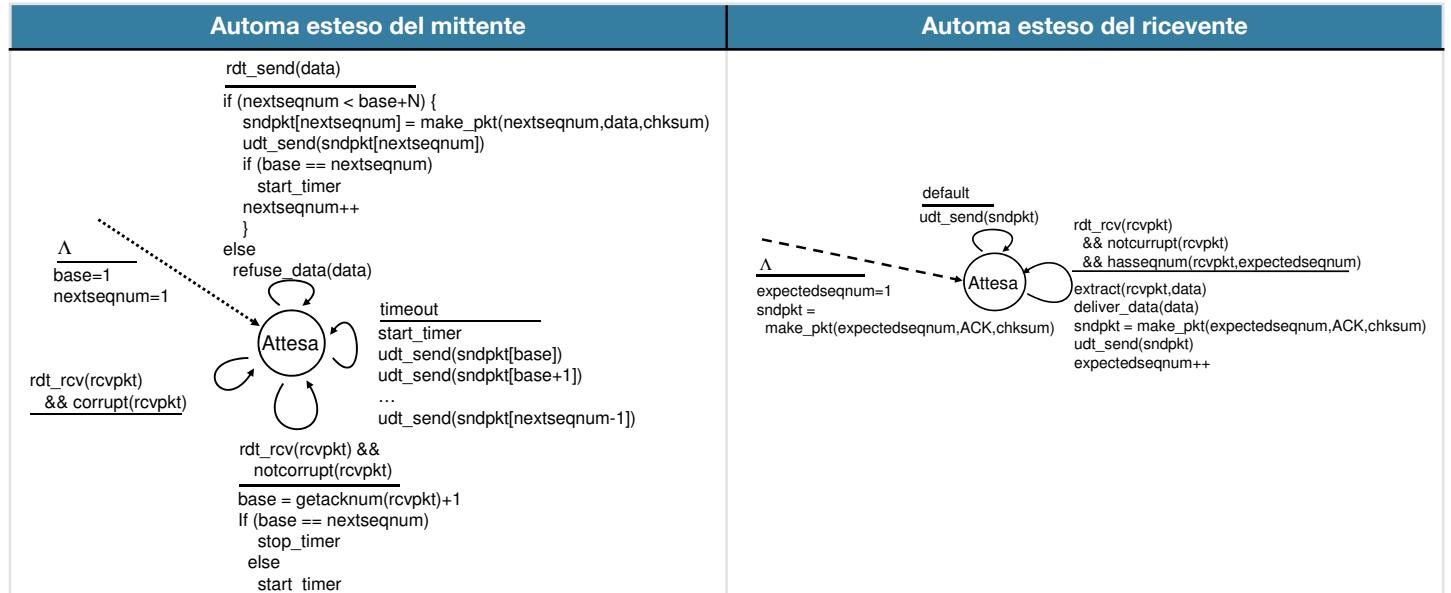
Nextseqnum = il numero di sequenza del prossimo pacchetto da inviare;

Possiamo individuare 4 intervalli:

[0, base -1]	pacchetti già trasmessi e riscontrati
[base, nextseqnum]	pacchetti inviati ma non ancora riscontrati
[nextseqnum, base + N +1]	slot liberi per altri pacchetti provenienti dal livello superiore
[base + N, inf]	slot non utilizzabili fino alla ricezione dei risconti

L'intervallo [base, base + N +1] prende il nome di **Window** (protocollo a scorrimenti di finestra/sliding windows). Per quanto riguarda TCP, esso utilizza un campo da 32 bit, ed utilizza come conteggio dei numeri di sequenza, il conteggio dei byte nello stream.

Automi estesi per il lato d'invio e di ricezione di un protocollo **go-back-n** :



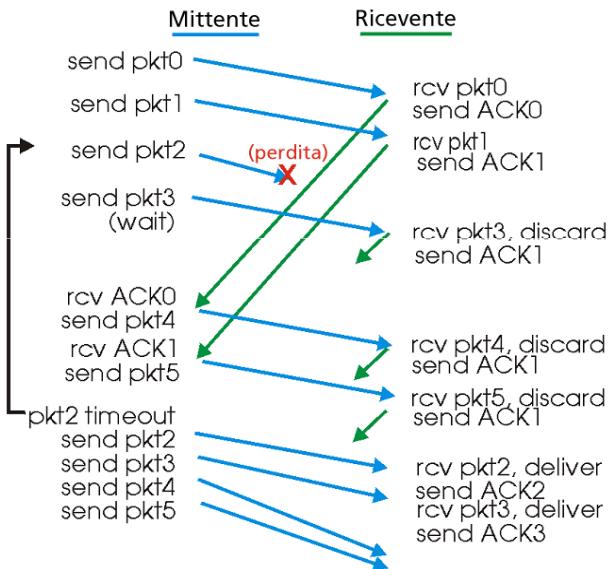
Tale protocollo prevede una programmazione basata su eventi

Il mittente GBN deve rispondere a tre tipi di evento:

- *Invocazione dall'alto (rdt_send)*. Controlla che la finestra non sia piena ed in quel caso crea e spedisce il pacchetto, andando a porlo in un buffer o implementando una politica di flag/semafori per bloccare la chiamata a coda piena.
- *Ricezione di un ACK*. Acknowledgment cumulativo.
- *Timeout*. Il timer è unico e si riferisce al pacchetto trasmesso meno di recente e non ancora riscontrato. Alla ricezione del suddetto ACK, il timer viene fatto ripartire. Viene fermato in caso di coda vuota. Un evento di Timeout(n) ha come causa la ritrasmissione del pacchetto n e tutti i pacchetti con i numeri di sequenza più grandi nella finestra.

Il ricevente GBN:

- *ACK-soltanto*: invia sempre un ACK per il pacchetto ricevuto correttamente con il numero di sequenza più alto, ciò potrebbe generare ACK duplicati e il ricevente deve solo memorizzare il valore di *expectedseqnum*.
- *Pacchetto fuori sequenza*: viene scartato -> senza buffering del ricevente
 - rimanda un ACK per il pacchetto con il numero di sequenza più alto in sequenza.



GBN fa uso di notifiche collettive ovvero alla ricezione di un ACK per il pacchetto k , tutti i pacchetti con numero di sequenza inferiore sono anch'essi contrassegnati, dato che GBN inoltra al livello superiore un pacchetto alla volta. In caso di ACK fuori sequenza, esso verrà scartato e verrà rispedito il pacchetto meno recente non ancora riscontrato (cioè viene effettuato in quanto il protocollo deve fornire al livello superiore i dati in modo ordinato).

Vantaggi: la semplicità è il fatto che il destinatario non deve memorizzare i pk fuori ordine. Lo scarto di un pacchetto fuori ordine è "obbligatorio" per questo tipo di protocollo ($n+1$, ack n , trasm n e $n+1$).

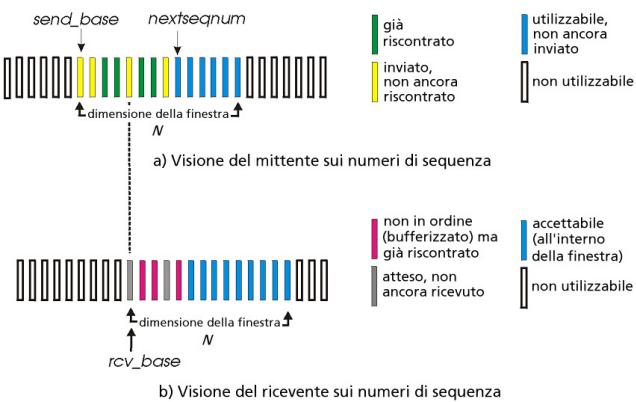
Svantaggi: scartando un pacchetto si incorre nel rischio di successive ritrasmissioni perse o alterate.

Ripetizione Selettiva

Tale protocollo cerca di risolvere i problemi di prestazione di GBN in determinate condizioni/scenari, in particolare quando al dimensione della finestra e il prodotto larghezza di banda * ritardo sono entrambi grandi si ha come effetto quello di un numero elevato di pacchetti nelle pipeline. Ed un solo errore può provocare la ritrasmissione di tutti i pacchetti (non necessarie).

Il Protocollo di ripetizione selettiva, non fa altro che forzare il mittente a spedire quei pacchetti su cui esistono sospetti di errore (smarrimento/alterazione). Anche in questo caso, useremo una dimensione di finestra pari ad N per il numero di pacchetti senza riscontro nella pipeline, il quale stavolta è fissato dal protocollo, non molto grande per evitare problemi di ritrasmissione e/o saturare le pipeline.

Il destinatario SR invierà un riscontro per i pacchetti correttamente ricevuti sia in ordine sia fuori ordine. Questi vengono bufferizzati finché non sono stati ricevuti tutti i pacchetti mancanti, momento in cui un blocco di pacchetti può essere trasportato in ordine al livello superiore,



Eventi e azioni di un mittente SR

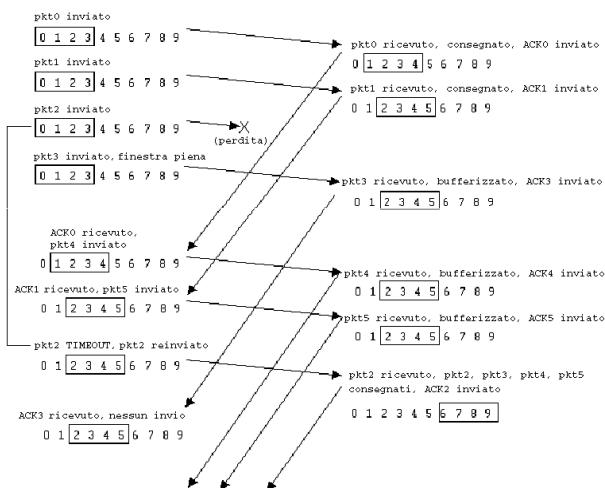
- Dati ricevuti dall'alto**, quando si ricevono dati dall'alto, il mittente SR controlla il successivo numero di sequenza disponibile per il pacchetto. Se è all'interno della finestra del mittente, i dati vengono impacchettati e inviati; altrimenti sono salvati nei buffer o restituiti al livello superiore per una successiva ritrasmissione, come in GBN.
- Timeout**, Vengono usati ancora i contatori per cautelarsi contro la perdita di pacchetti. Ora però ogni pacchetto deve avere un proprio timer logico, dato che al timeout sarà ritrasmesso solo un pacchetto. Si può utilizzare un solo contatore hardware per simulare le operazioni di più timer logici.

- 3. ACK ricevuto.** Se si riceve un ACK, il mittente SR marca tale pacchetto come ricevuto, ammesso che sia nella finestra. Se il numero di sequenza del pacchetto e' uguale a $send_base$, la base della finestra si muove verso il pacchetto non riscontrato con il più piccolo numero di sequenza. Se la finestra si sposta e ci sono pacchetti non trasmessi con un numero di sequenza che ora cade all'interno della finestra, questi vengono trasmessi.

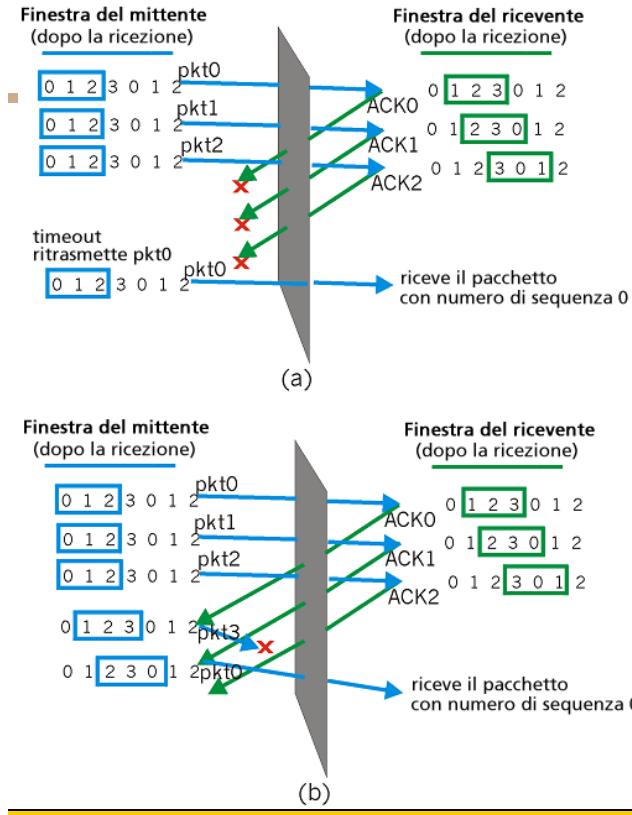
3.1. $ACK(n) \& n \in [send_base, send_base + N - 1] \Rightarrow$ buono, marca n come ricevuto , se $n = send_base \Rightarrow$ sposto base fino al primo non riscontrato.

Eventi e azioni di un ricevente SR

- Il pacchetto con numero di sequenza nell'intervallo $[rcv_base, rcv_base + N-1]$ viene ricevuto correttamente,** in questo caso il pacchetto ricevuto ricade all'interno della finestra del ricevente e al mittente viene restituito un pacchetto di ACK selettivo. Se il pacchetto non era già stato ricevuto, viene bufferizzato. Se presenta un numero di sequenza uguale alla base della finestra di ricezione (rcv_base) vengono consegnati al livello superiore.
- Viene ricevuto il pacchetto con numero di sequenza nell'intervallo $[rcv_base - N, rcv_base - 1]$** in questo caso, si deve generare un ACK, anche se si tratta di un pacchetto che il ricevente ha già riscontrato.
- Altrimenti** si ignora il pacchetto



Non sempre il mittente e destinatario hanno la stessa visuale su cosa e' stato ricevuto correttamente. Ovvero le finestre dei due non coincidono, ciò ha conseguenze importanti quando abbiamo a che fare con la realtà di un intervallo finito di numeri di sequenza.



Esempio:

- Numeri di sequenza: 0, 1, 2, 3
 - Dimensione della finestra = 3
 - Il ricevente non vede alcuna differenza fra i due scenari!
 - Passa erroneamente i dati duplicati come nuovi in (a)

D: Qual è la relazione fra lo spazio dei numeri di sequenza e la dimensione della finestra?

Il caso pessimo e' che il mittente abbia inviato pacchetti fino a riempire la propria finestra. Per gestire il tutto senza problemi i numeri di sequenza devo esser pari al doppio.

#Numseq >= 2N

Note conclusive: I pacchetti tra mittente e destinatario viaggiano nella rete si può verificare il fenomeno del riordinamento dei pacchetti. Ciò si manifesta con la ricezione di vecchie copie di un pacchetto con numero di seq o riscontro x, il quale e' non e' presente in nessuna delle due finestre, in questo caso dobbiamo portare particolare attenzione in quanto il riuso dei numeri di sequenza crea delle problematiche riguardo la duplicazione dei pacchetti. Ciò viene risolto cercando di non riutilizzare un numseq prima di un determinato lasso di tempo, presumo il ttl (time to

live).

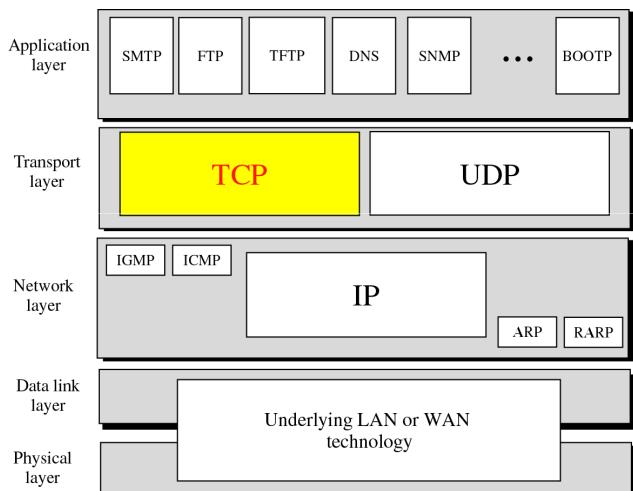
UDP=> S&W or Pipeline

Pipeline => Go-Back-N or SR, prevalentemente utilizzati dal link.

Meccanismo	Uso e commenti
Checksum	Utilizzato per rilevare errori sui bit in un pacchetto trasmesso.
Timer	Serve a far scadere un pacchetto e ritrasmetterlo, forse perché il pacchetto (o il suo ACK) si è smarrito all'interno del canale. I timeout si possono verificare per via dei ritardi anziché degli smarrimenti (timeout prematuro), o quando il pacchetto è stato ricevuto dal destinatario ma è andato perduto il relativo ACK dal destinatario al mittente. Per questi motivi il destinatario può ricevere copie duplicate di un pacchetto.
Numero di sequenza	Usato per numerare sequenzialmente i pacchetti di dati che fluiscono tra mittente e destinatario. Le discontinuità nei numeri di sequenza di pacchetti ricevuti consentono al destinatario di rilevare i pacchetti persi. I pacchetti con numero di sequenza ripetuto consentono al destinatario di rilevare copie dei pacchetti duplicati.
Acknowledgment (ACK)	Utilizzato dal destinatario per comunicare al mittente che un pacchetto o un insieme di pacchetti sono stati ricevuti correttamente. I riscontri trasporteranno generalmente i numeri di sequenza del pacchetto o dei pacchetti da confermare. A seconda del protocollo, i riscontri possono essere individuali o cumulativi.
Acknowledgment negativo (NAK)	Usato dal destinatario per comunicare al mittente che un pacchetto non è stato ricevuto correttamente. I riscontri negativi trasporteranno generalmente il numero di sequenza del pacchetto che non è stato ricevuto correttamente.
Finestra e pipelining	Il mittente può essere forzato a inviare soltanto pacchetti con numeri di sequenza che ricadono in un determinato intervallo. Consentendo a più pacchetti di essere trasmessi ma non ancora riscontrati, si può migliorare l'utilizzo del mittente rispetto alla modalità operativa stop-and-wait. Vedremo tra breve che la dimensione della finestra può essere impostata sulla base della capacità del destinatario di ricevere e bufferizzare messaggi, o su quella del livello di congestione della rete, o su entrambe.

Trasporto orientato alla connessione: TCP

Definito nelle RFC 793, 1122, 1323, 2018 e 2581.



- Comunicazione da processo-a-processo
- Una connessione TCP è punto- punto:
un mittente, un destinatario
il multicast non è possibile con TCP
- flusso di byte affidabile, in sequenza:
nessun "confine ai messaggi"
- pipeline:
il controllo di flusso e di congestione TCP definiscono la dimensione della finestra
- buffer d'invio e di ricezione
Una connessione TCP consiste di buffers, variabili e di 2 sockets verso i due end- systems
- Una connessione TCP offre un servizio full duplex:
flusso di dati bidirezionale nella stessa connessione

la massima quantità di dati che vengono imbustati in un segmento è detta MSS (maximum segment size)

- orientato alla connessione:

I handshaking (scambio di messaggi di controllo) inizializza lo stato del mittente e del destinatario prima di scambiare i dati

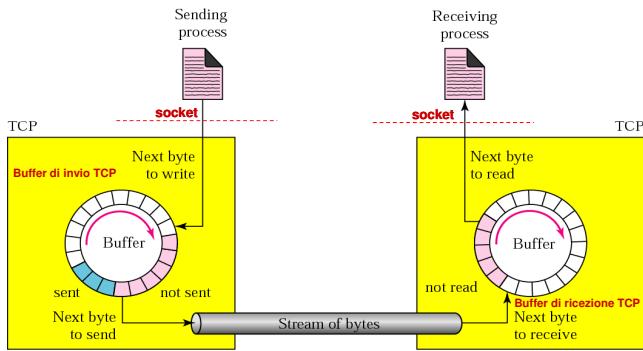
- Processo tx ≠ da processo Rx:

Due buffer: uno per lato

- non offre: temporizzazione, ampiezza di banda minima

Attenzione: Una connessione TCP non è un circuito end-to-end TDM o FDM, come in una rete a commutazione di circuito e neanche un circuito virtuale, in quanto lo stato della connessione risiede completamente nei due sistemi terminali.

Connessione TCP



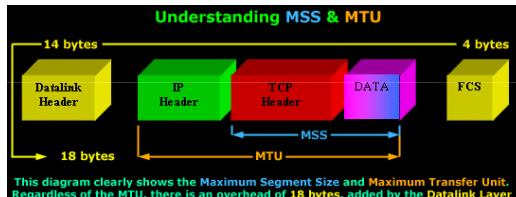
Caratteristiche principali del protocollo:

- Orientato alla Connessione, dato che prima di effettuare lo scambio dei dati, vi è la trasmissione di alcuni pacchetti preliminari (2 non trasportano carico utile l'ultimo invece può farlo) per determinare i parametri dello scambio dati (3 way -handshake), e momento in cui i due host inizializzeranno molte variabili di stato TCP.
- Lo stato della connessione risiede sempre nei due sistemi terminali.
- Servizio full-duplex, il flusso dei dati tra due host a livello applicazione può verificarsi contemporaneamente nelle due direzioni.

- TCP dirige i dati nel buffer d'invio (riservato in fase di connessione) della connessione, RFC definisce che TCP dovrebbe spedire tali dati in segmenti quanto è più conveniente.

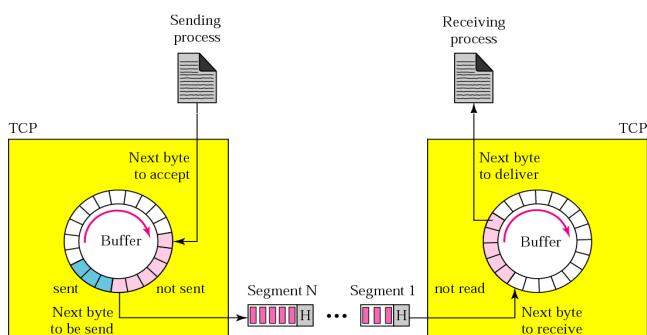
La massima quantità di dati che può esser posizionata in un segmento viene limitata da due dimensioni:

- **MSS [maximum segment size]**, dimensione massima del segmento. Ovvero la massima quantità di dati a livello applicazione nel segmento (non la max dim del segmento TCP).
 - MSS è relazionata a MTU che definisce il frame più grande a livello di datalink
 - Valori comuni di MTU sono: 1460, 536, 512
- **MTU [maximum transmission unit]**, unità trasmisiva massima. Ovvero la lunghezza del frame più grande a livello di collegamento che può esser inviato dall'host mittente locale.



Quindi avremo: **MTU { Datagram Link { IP (header, segmento TCP = MMS) } }**, dove il segmento TCP è determinato dallo stesso TCP andando ad accoppiare ogni porzione di dati con l'intestazione TCP.

Struttura dei segmenti TCP



Quando TCP deve inviare un file si hanno almeno due casistiche di base:

- *file di grandi dimensioni*. In questo caso il protocollo frammenta il file in porzioni di dimensione MMS.
- *file di piccole dimensioni*. In questo caso il protocollo non effettuerà nessuna frammentazione.

L'intestazione TCP occupa 20 byte, contro gli 8 di UDP.

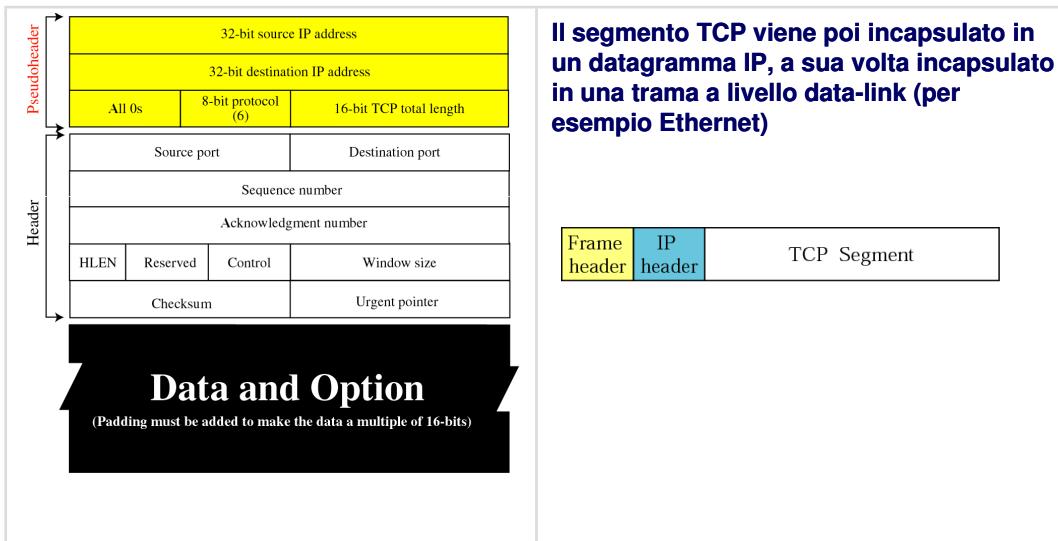
TCP Header

Bit offset	Bits 0–3	4–7	8–15								16–31				
0			Source port										Destination port		
32			Sequence number												
64			Acknowledgment number												
96	Data offset	Reserved	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size				
128			Checksum										Urgent pointer		
160			Options (optional)												
160/192+			Data												

- **Source port [16 bit]** - Identifica il numero di porta sull'host mittente associato alla connessione TCP.
- **Destination port [16 bit]** - Identifica il numero di porta sull'host destinatario associato alla connessione TCP.
- **Sequence number [32 bit]** - Numero di sequenza, indica lo scostamento (espresso in byte) dell'inizio del segmento TCP all'interno del flusso completo, a partire dall' Initial Sequence Number (ISN), negoziato all'apertura della connessione.
- **Acknowledgment number [32 bit]** - Numero di riscontro, ha significato solo se il flag ACK è settato a 1, e conferma la ricezione di una parte del flusso di dati nella direzione opposta, indicando il valore del prossimo Sequence number che l'host mittente del segmento TCP si aspetta di ricevere.
- **Data offset [4 bit]** - Indica la lunghezza (in word da 32 bit) dell'header del segmento TCP; tale lunghezza può variare da 5 word (20 byte) a 15 word (60 byte) a seconda della presenza e della lunghezza del campo facoltativo Options.
- **Reserved [4 bit]** - Bit non utilizzati e predisposti per sviluppi futuri del protocollo; dovrebbero essere settati a zero.
- **Flags [8 bit]** - Bit utilizzati per il controllo del protocollo:
 - CWR (Congestion Window Reduced) - se settato a 1 indica che l'host sorgente ha ricevuto un segmento TCP con il flag ECE settato a 1 (aggiunto all'header in [RFC 3168](#)).
 - ECE (ECN-Echo) - se settato a 1 indica che l'host supporta ECN (Explicit Congestion Notification) durante il 3-way handshake (aggiunto all'header in [RFC 3168](#)).
 - URG - se settato a 1 indica che nel flusso sono presenti dati urgenti alla posizione (offset) indicata dal campo Urgent pointer. Urgent Pointer punta alla fine dei dati urgenti;
 - ACK - se settato a 1 indica che il campo Acknowledgment number è valido;
 - PSH - se settato a 1 indica che i dati in arrivo non devono essere bufferizzati ma passati subito ai livelli superiori dell'applicazione;
 - RST - se settato a 1 indica che la connessione non è valida; viene utilizzato in caso di grave errore; a volte utilizzato insieme al flag ACK per la chiusura di una connessione.
 - SYN - se settato a 1 indica che l'host mittente del segmento vuole aprire una connessione TCP con l'host destinatario e specifica nel campo Sequence number il valore dell' Initial Sequence Number (ISN); ha lo scopo di sincronizzare i numeri di sequenza dei due host. L'host che ha inviato il SYN deve attendere dall'host remoto un pacchetto SYN/ACK.
 - FIN - se settato a 1 indica che l'host mittente del segmento vuole chiudere la connessione TCP aperta con l'host destinatario. Il mittente attende la conferma dal ricevente (con un FIN-ACK). A questo punto la connessione è ritenuta chiusa per metà: l'host che ha inviato FIN non potrà più inviare dati, mentre l'altro host ha il canale di comunicazione ancora disponibile. Quando anche l'altro host invierà il pacchetto con FIN impostato la connessione, dopo il relativo FIN-ACK, sarà considerata completamente chiusa.
- **Advertise Window [16 bit]** - Indica la dimensione della finestra di ricezione dell'host mittente, cioè il numero di byte che il mittente è in grado di accettare a partire da quello specificato dall'acknowledgment number.
- **Checksum [16 bit]** - Campo di controllo utilizzato per la verifica della validità del segmento. È ottenuto facendo il complemento a 1 della somma complemento a uno a 16 bit dell'intero header TCP (con il campo checksum messo a zero), dell'intero payload, con l'aggiunta di

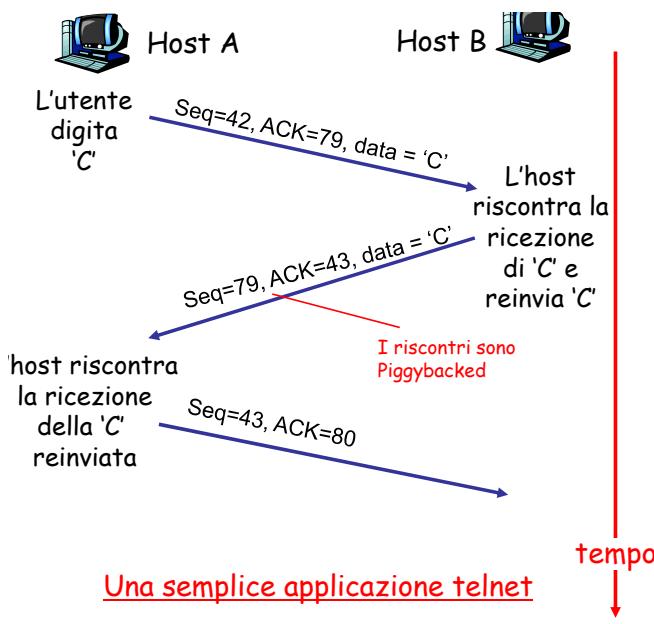
uno pseudo header composto da: indirizzo IP sorgente(32bit),indirizzo IP destinazione(32bit), un byte di zeri, un byte che indica il protocollo e due byte che indicano la lunghezza del pacchetto TCP (header + dati).

- **Urgent pointer [16 bit]** - Puntatore a dato urgente, ha significato solo se il flag **URG** è settato a 1 ed indica lo scostamento in byte a partire dal Sequence number del byte di dati urgenti all'interno del flusso.
- **Options** - Opzioni (facoltative) per usi del protocollo avanzati.
- **Data** - rappresenta il carico utile o payload da trasmettere.



Numeri di sequenza e numeri di riscontro

TCP vede i dati come un flusso di byte non strutturati ma ordinati. Pertanto il **Sequence Number** per un segmento e' il numero nel flusso di byte del primo byte del segmento. I lati della connessione TCP scelgono a caso un numero di sequenza iniziale, così da diminuire la possibilità che un segmento ancora presente nella rete venga interpretato erroneamente come un segmento della connessione attuale.



l'Ack number invece e' il numero di sequenza del byte successivo che vogliamo ricevere dall'host mittente. E dato che TCP effettua il riscontro dei byte fino al primo byte mancante nel flusso, si dice che tale protocollo offre **riscontri cumulativi** (cumulative acknowledgment).

Un file di 500.000 byte, con un MMS di 1.000 byte, verrà suddiviso in 500 segmenti per questo flusso dati. Al primo segmento verrà assegnato un sequence number pari a 0, il secondo pari a 1.000 ecc.

NB: le RFC non specificano quale comportamento deve tenere TCP in caso di segmenti non ordinati.

Telnet e' un protocollo di livello applicazione utilizzato per il login remoto che utilizza tcp. E' un ottimo esempio per studiare i numeri di sequenza e di riscontro.

In particolar modo notiamo come molto spesso i riscontri siano portati a cavalluccio (**piggybacked**) sul segmento dati dal server al client o viceversa.

Ritrasmissione in TCP

Quando un segmento rimane non riscontrato (**unacknowledged**) per un certo periodo di tempo, il TCP assume che sia andato perso e lo ritrasmette.

TCP cerca di calcolare il “round trip time” (RTT) richiesto da un segmento e dal suo ACK, dalla conoscenza di RTT, TCP può valutare quanto tempo aspettare prima di andare in timeout e di ritrasmettere.

Approfondimento

Per ogni segmento ricevuto in sequenza inoltre TCP lato ricevente invia un Acknowledgment Number o numero di riscontro dell'avvenuta ricezione. Il numero di riscontro presente in un segmento riguarda il flusso di dati nella direzione opposta. In particolare, il numero di riscontro inviato da A a B è pari al numero di sequenza atteso da A e, quindi, riguarda il flusso di dati da B ad A.

In particolare il protocollo TCP adotta la politica di Conferma cumulativa, ovvero l'arrivo di numero di riscontro indica al TCP trasmittente che il ricevente ha ricevuto e correttamente inoltrato al proprio processo applicativo il segmento avente numero di sequenza pari al numero di riscontro indicato (-1) ed anche tutti i segmenti ad esso precedenti. Per tale motivo TCP lato trasmittente mantiene temporaneamente in un buffer una copia di tutti i dati inviati, ma non ancora riscontrati: quando questi riceve un numero di riscontro per un certo segmento, deduce che tutti i segmenti precedenti a quel numero sono stati ricevuti correttamente liberando il proprio buffer dai dati. La dimensione massima dei pacchetti riscontrabili in maniera cumulativa è specificata dalle dimensioni della cosiddetta [finestra scorrevole](#).

Per evitare tempi di attesa troppo lunghi o troppo corti per ciascun segmento inviato TCP lato trasmittente avvia un [timer](#), detto timer di ritrasmissione RTO (Retransmission Time Out): se questi non riceve un ACK di riscontro per il segmento inviato prima che il timer scada, TCP assume che tutti i segmenti trasmessi a partire da questo siano andati persi e quindi procede alla ritrasmissione.

Si noti che, in TCP, il meccanismo dei numeri di riscontro non permette al ricevitore di comunicare al trasmettitore che un segmento è stato perso, ma alcuni dei successivi sono stati ricevuti (meccanismo ad Acknowledgment Number negativi), quindi è possibile che per un solo pacchetto perso ne debbano essere ritrasmessi molti. Questo comportamento non ottimale è compensato dalla semplicità del protocollo. Questa tecnica è detta [Go-Back-N](#) (vai indietro di N segmenti); l'alternativa, ovvero progettare il protocollo in modo tale che solo i pacchetti effettivamente persi vengano ritrasmessi, è detta [Selective Repeat](#) (ripetizione selettiva); l'utilizzo però di alcuni campi opzionali appositi permette l'utilizzo della ripetizione selettiva.

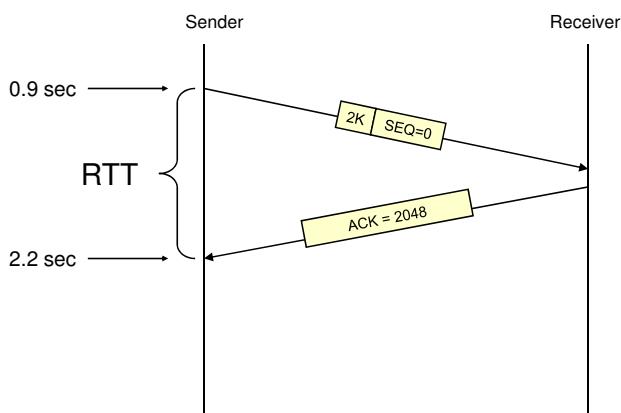
I numeri di riscontro e i relativi timer di ritrasmissione permettono quindi di realizzare la consegna affidabile, ovvero di garantire che tutti i dati inviati siano comunque consegnati nel caso in cui qualche pacchetto possa essere perso nel transito attraverso la rete (controllo di errore in termini di riscontro di trasmissione).

Stima del tempo di andata e ritorno

TCP, come il protocollo rdt si basa sull'utilizzo dei time-out. La questione e' se il time-out deve esser > o < del RTT e in che misura!! Partendo dal fatto che RTT non e' prevedibile in quanto IP come sappiamo e' best-effort, che il timer e' associato ad un solo segmento, ovvero al piu' vecchio non ancora riscontrato.

Stima del tempo

- **SampleRTT**, la quantità di tempo che intercorre tra l'invio del segmento (ovvero il suo passaggio a IP) e quello di ricezione dell'acknowledgment del segmento. Generalmente viene calcolato quando spediamo il primo segmento e poi ricalcolato/stimato per uno solo sei segmenti trasmessi e non ancora riscontrati. In particolare TCP non ricalcola mai il SampleRTT per i pacchetti ritrasmessi perché'??
- **EstimatedRTT**, calcolo della media mobile esponenziale ponderata dei vari SampleRTT dato che esso varia in modo significativo a causa della congestione dei router e del carico sui sistemi terminali.



$$RTT = 2.2 \text{ sec} - 0.9 \text{ sec.} = 1.3 \text{ sec}$$

tenderà a crescere.

Intervallo di tempo di ritrasmissione

L'intervallo logicamente dovrà esser:

$$\text{Intervallo} = \text{EstimatedRTT} + \text{value}$$

Dove **value** dovrà esser un valore scelto con cura perché' un valore elevato porterebbe ad una ritrasmissione troppo lenta del segmento perduto e nello stesso tempo non un valore estremamente piccolo dato che esso avrebbe come risultato delle ritrasmissioni non necessarie.

Il concetto fondamentale per la sua determinazione e' il concetto di **fluttuazione** nei valori di SampleRTT ovvero la **deviazione** tra SampleRTT e EstimatedRTT => **DevRTT**.

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4 * \text{DevRTT}$$

Trasferimento dati affidabile

TCP crea un **servizio di trasporto affidabile** al di sopra del servizio inaffidabile e best-effort di IP, assicurando che il flusso di byte non sia alterato, non abbia buchi, non presenti duplicazioni e rispetti la sequenza originaria.

Le procedure suggerite per la gestione dei timer TCP [RFC 2988] utilizzano **un solo timer di ritrasmissione**, anche in presenza di più' segmenti trasmessi ma non ancora riscontrati.

Possiamo quindi schematizzare il comportamento come segue:

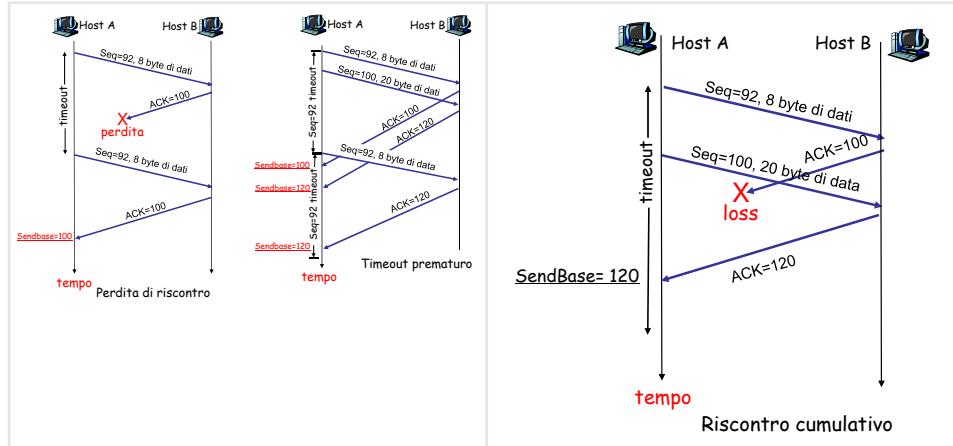
```
NextSeqNum= InitialSeqNumber
SendBase = InitialSeqNumber
loop( true) {
    switch (evento)
```

evento: ricezione dati dall'applicazione superiore

- creano segmenti TCP con numero di sequenza NextSeqNum
- if (timer attualmente non funzionante)
 - avvia il timer
- passa il segmento a IP
- NextSeqNum = NextSeqNum + lunghezza(dati) "solo dati non il preambolo"

evento: timeout del timer

- ritrasmetti il segmento non ancora riscontrato con il piu' piccolo numero di sequenza
- raddoppia il valore del timer
- avvia il timer



Ritrasmissione Rapida

Il periodo di timeout spesso è relativamente lungo: lungo ritardo prima di ritrasmettere il pacchetto perduto.

Rileva i segmenti perduti tramite gli ACK duplicati, ciò implica che il mittente spesso invia molti segmenti e se un segmento viene smarrito, è probabile che ci saranno molti ACK duplicati.

Se il mittente riceve 3 ACK per lo stesso dato, suppone che il segmento che segue il dato riscontrato è andato perduto => ritrasmissione rapida: rispedisce il segmento prima che scada il timer.

evento: ACK ricevuto, con valore del campo ACK pari a y

- ```

• fi (y > SendBase) {
 SendBase = y
 if (esistono attualmente segmenti non ancora riscontrati; SendBase != NextSeqNum)
 avvia il timer
} else { /*ACK duplicato per un segmento già riscontrato*/
 • incrementa il numero di ACK duplicati ricevuti per y
 • if (numero ack di y = 3)
 rispedisci il pacchetto con numero di sequenza y
 avvia il timer
}

```

Caratteristiche:

- **Riscontri Cumulativi**, in questo caso vediamo in pratica cosa succede, ovvero il riscontro di tutti i segmenti con num < y (ACK).
- **Raddoppio timer**, effettuato quando si verifica un evento di timeout, notiamo che la scadenza del timer viene causata da una congestione della rete, in tali periodi se le sorgenti continuano a spedire i pacchetti senza raddoppiare il timer, la congestione può peggiorare. Nei casi successivi il valore in TimerInterval viene calcolato in modo classico, in modo da ripetere il normale svolgimento.
- **Fast Retransmission**, ovvero il fenomeno in cui il periodo di timeout può rivelarsi eccessivamente lungo. Il mittente tramite il meccanismo degli ack duplicati spesso riesce a rilevare la perdita dei pacchetti anche prima dello scadere del timer.

| Evento nel destinatario                                                                                                                  | Azione del ricevente TCP                                                                                      |
|------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|
| Arrivo ordinato di un segmento con numero di sequenza atteso. Tutti i dati fino al numero di sequenza atteso sono già stati riscontrati. | ACK ritardato. Attende fino a 500 ms l'arrivo del prossimo segmento. Se il segmento non arriva, invia un ACK. |
| Arrivo ordinato di un segmento con numero di sequenza atteso. Un altro segmento è in attesa di trasmissione dell'ACK.                    | Invia immediatamente un singolo ACK cumulativo, riscontrando entrambi i segmenti ordinati.                    |
| Arrivo non ordinato di un segmento con numero di sequenza superiore a quello atteso. Viene rilevato un buco.                             | Invia immediatamente un ACK duplicato, indicando il numero di sequenza del prossimo byte atteso.              |
| Arrivo di un segmento che colma parzialmente o completamente il buco.                                                                    | Invia immediatamente un ACK, ammesso che il segmento cominci all'estremità inferiore del buco.                |

ricevuto.

Gli ack duplicati vengono spediti dal destinatario quando riceve dei pacchetti con un numero di seq maggiore di quello atteso, rilevando così un buco. Per indicare al mittente tale destinazione, non fa' altro che rispedire un ack dell'ultimo byte di dati che ancora nn ha ricevuto andando così a duplicare l'ack.

Il mittente procede con una **ritrasmissione rapida**, rispedendo il pacchetto prima che il timer scada.

Una modifica preposta da TCP del cosiddetto **riscontro selettivo**, e' la possibilità di riscontrare , in modo selettivo i segmenti non in ordine anziché riscontrarli cumulativamente l'ultimo segmento

Un altro fenomeno delle comunicazioni TCP e' il **Delayed ACK**, ovvero quando il destinatario attende un tempo finito (relativamente breve) prima di mandare un'ack per il segmento ricevuto, ciò' viene effettuato con la speranza di ricevere i segmenti successivi in rapida successione ed inviare un ACK cumulativo per i pacchetti ricevuti (in ordine).

**GO-Back-N o Ripetizione Selettiva?** Per rispondere a tale domanda dobbiamo analizzare i seguenti aspetti:

- Se analizziamo le variabili utilizzate, sendbase e nextseqnum, tcp assomiglia molto a GBN.
- Ma molte implementazioni TCP differiscono da GBN in quanto bafferizzano i segmenti ricevuti correttamente ma non in ordine.
- Se utilizzassimo il cosiddetto riscontro selettivo per i segmenti arrivati non in ordine TCP assomiglierebbe a SR.

Per questi motivi TCP viene identificato come un'ibrido tra GBN e SR.

## Controllo di Flusso

TCP offre un **servizio di controllo sul flusso** alle proprie applicazioni per scongiurare l'eventualità di un overflow del buffer di ricezione. Il controllo di flusso e' un **controllo sulle velocità**, dato che paragona la frequenza d'invio del mittente con quella di lettura dell'applicazione ricevente.

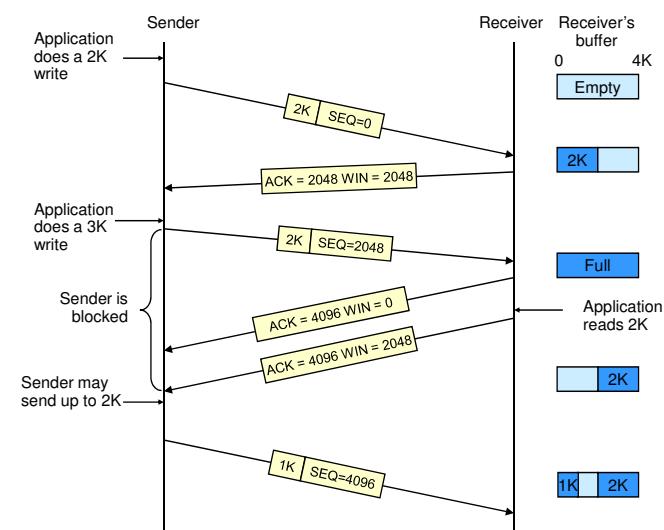
## Approfondimento

In **telecomunicazioni** nell'ambito delle **rete di telecomunicazioni** il controllo di flusso, oltre al controllo della **congestione**, è un tipo di controllo di **trasmissione** effettuato dagli agenti di una comunicazione (il mittente e il destinatario) sui **pacchetti** inviati e ricevuti attraverso alcuni **protocolli di comunicazione** come ad esempio **TCP**.

L'obiettivo di tale controllo è evitare che il mittente invii una quantità eccessiva di dati che potrebbero, in alcune situazioni, mandare in **overflow** il **buffer** di ricezione del destinatario generando una perdita di pacchetti e la necessità di ritrasmissione con perdita in efficienza (**Goodput**) a causa delle ritrasmissioni dei pacchetti persi, risultando dunque particolarmente utile per il mantenimento delle prestazioni della connessione.

In altre parole il controllo di flusso è a livello logico e fisico agli estremi della connessione dipendendo esclusivamente dalle capacità riceventi del ricevitore, mentre il controllo di congestione, pur coinvolgendo anch'esso il trasmettitore e il ricevitore agli estremi, dipende dal traffico sui nodi interni di commutazione della rete di trasporto sul quale transitano anche i flussi dati di altre connessioni tra altri utenti, che sommati tra loro potrebbero appunto generare effetti di congestione interni.

Presupposti: TCP scarta i segmenti non in ordine!



Tale meccanismo di controllo viene offerto andando a mantenere dal lato del mittente una variabile chiamata **finestra di ricezione - receive window** la quale fornisce al mittente un'indicazione dello spazio libero disponibile nel buffer del destinatario.

Dato che TCP è full-duplex, i due mittenti mantengono finestre di ricezione distinte.

Supponendo uno scambio di dati host A → host B, saranno attive le seguenti variabili:

Host B : **LastByteRead**, numero dell'ultimo byte nel flusso di dati letto dall'applicazione; **LastByteRcvd**, numero dell'ultimo byte nel flusso di dati letto dalla rete ed inserito nel buffer.

Esse saranno in relazione..

$$\text{LastByteRcvd} - \text{LastByteRead} \leq \text{RcvBuffer} \rightarrow \text{RcvWindow} = \text{RcvBuffer} - [\text{LastByteRcvd} - \text{LastByteRead}]$$

come si puo' notare la RcvWindow e' dinamica. Ed il suo valore viene comunicato ad A tramite l'header TCP.

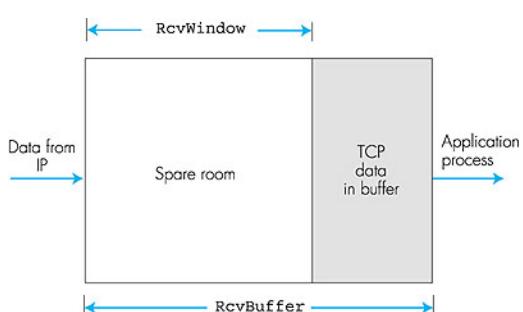


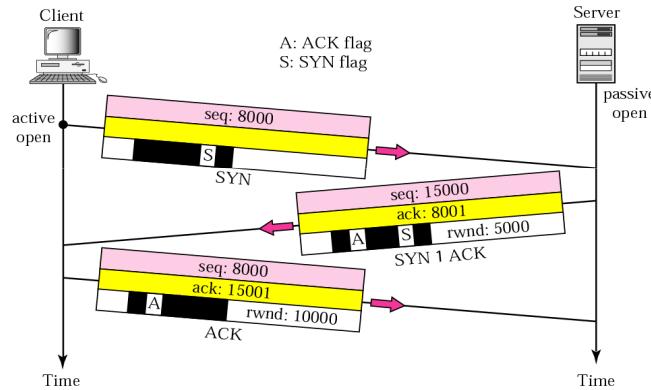
Figure 3.35: The receive window (RcvWindow) and the receive buffer (RcvBuffer)

Host A: **LastByteSent** e **LastByteAcked**, la loro differenza rappresenta la quantità di dati non ancora riscontrati ma spediti. Mantenendo tale valore  $\leq$  di RcvWindow si garantisce che l'host A non manda in overflow il buffer di B.

$$\text{LastByteSent} - \text{LastByteAcked} \leq \text{RcvWindow}$$

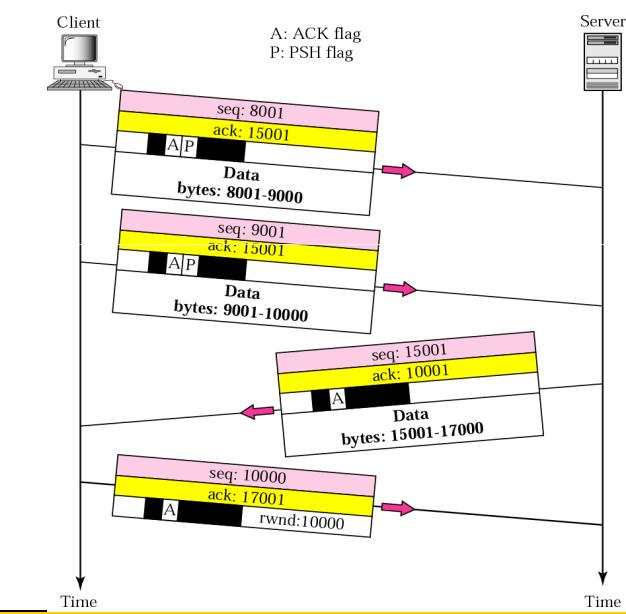
**Un problema** che deve esser considerato e' l'eventualità di una finestra di ricezione pari a 0 e la mancanza di ulteriori dati da inviare!! in questo caso la convenzione utilizzata da TCP e' che l'host A continui ad inviare segmenti con un byte di dati quando la finestra di ricezione e' pari a 0.

## Gestione delle connessioni TCP



### Gli step del 3-way-handshake sono i seguenti:

1.TCP-client invia un segmento al TCP-server, tale segmento non contiene dati a livello applicativo, ma il bit di SYN dell'header posto a 1. Inoltre l'header di tale segmento conterrà il numero di sequenza iniziale (clinet\_isn). Comunemente chiamato **segmento SYN**



2.TCP-server alla ricezione del datagramma, estraie il pacchetto e alloca i buffer e le variabili necessarie per la connessione TCP (ogni connessione ha il suo package (buffer, variabili..)). Dopo aver allocato il tutto il TCP-server risponde con un segmento caratterizzato da:

2.1.non ha dati a livello applicazioni, SYN = 1, ACK = client\_isn+1, Sequence Number = server\_isn .

2.2.Comunemente chiamato **segmento SYNACK**

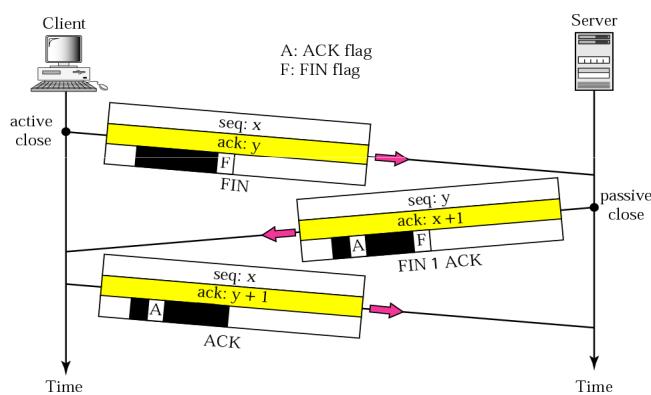
3.Alla ricezione del synack il client allocare il buffer e le variabili e successivamente risponde con un segmento con le seguenti caratteristiche:

3.1.può' trasportare carico utile

3.2.SYN = 0 , ACK = server\_isn + 1

3.3.Comunemente chiamato **segmento ACK**

### Gli step per la terminazione sono i seguenti:



Entrambe le parti della connessione possono decidere di chiudere la connessione, in questo caso presupponiamo che sia il client a chiuderla.

1.II TCP-client invia un segmento :

1.1.può' portare carico utile

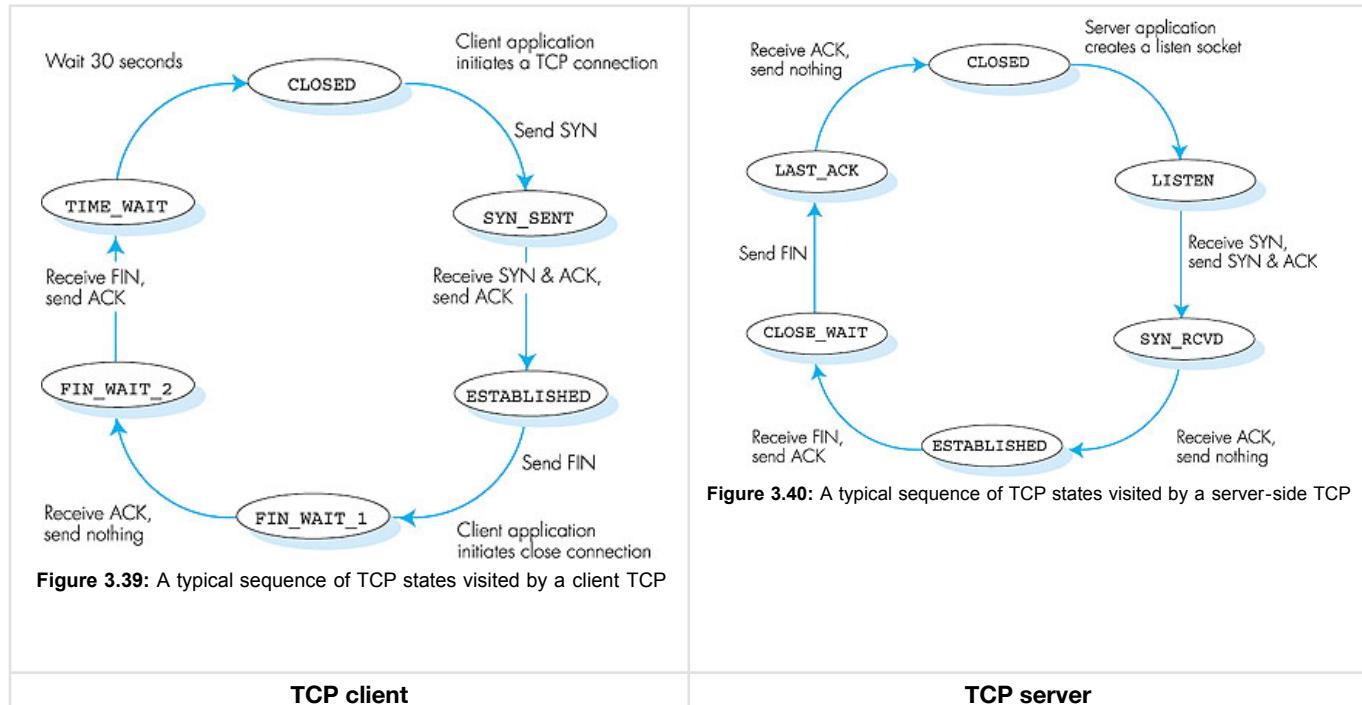
1.2.FIN = 1

2.II TCP server lo riceve invia un riscontro ACK, dopo di che invia il proprio segmento di shoutdown, con il bit FIN = 1

3.II TCP-client non farà' altro che rispondere con un riscontro ACK.

A questo punto tutte le risorse sugli host sono state deallocate.

Per quanto riguarda invece gli **stati di una connessione TCP** possiamo vedere dall'immagine i vari stati.



**NB: TCP client** l'etichette dell'arco ESTABLISHED e FIN\_WAIT\_1 vanno invertite dato che prima l'applicazione inizia la chiusura della connessione e poi viene inviato il FIN.

Nmap....host bersaglio!!

## Controllo di congestione TCP

Il TCP implementa il controllo della congestione, congiuntamente al controllo di flusso, solo agli estremi della comunicazione, e non richiede nessun supporto da parte dei router intermedi per realizzare questa funzione. Questo è coerente con il modello progettuale di IP, che prevede di aggiungere intelligenza ai nodi terminali lasciando i router il più possibile semplici.

TCP impone a ciascun mittente un limite alla frequenza di invio sulla propria connessione in funzione della congestione di rete percepita

### 1. x

Il meccanismo di controllo della congestione fa tenere traccia agli estremi della connessione di una variabile aggiuntiva: **La finestra di congestione - CongWin**.

$$\text{LastByteSent} - \text{LastByteAcked} \leq \min \{ \text{CongWin}, \text{RcvWindow} \}$$

Per la nostra trattazione ignoreremo RcvWindow, ciò ci porta a considerare la frequenza d'invio del mittente come CongWin / RTT bps.

### 2. Come il mittente TCP percepisce la presenza di congestione sul cammino verso la destinazione?

**Evento di perdita** = timeout o ricezione di tre ACK duplicati ; **Evento di Ok** = ricezione dei riscontri dei segmenti inviati.

TCP viene definito come **auto-temporizzante - self-clocking** perché' in base alla frequenza di arrivo dei riscontri aumenterà o diminuirà l'ampiezza della CongWin.

L'algoritmo utilizzato da TCP per regolare la frequenza e' **L'algoritmo di controllo della congestione TCP - TCP congestion control algorithm**.

- **Incremento additivo e decremento moltiplicativo (AIMD)**, la domanda a cui risponde tale componente e' quella relativa a quando dovrebbe esser la riduzione o l'incremento della finestra di congestione nei vari casi.

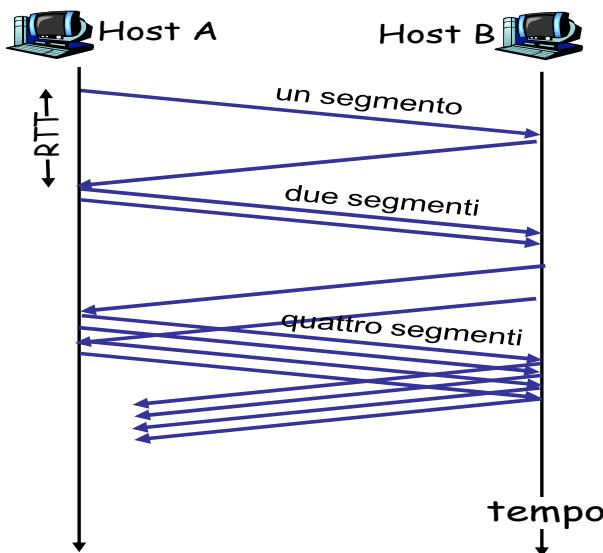
**Decremento Moltiplicativo** = In presenza di uno smarrimento TCP utilizza un approccio detto "Decremento moltiplicativo" che dimezza la CongWin, portandolo ad una drastica diminuzione fino al valore minimo di 1 MSS. Xke??

**Incremento Additivo** = nel caso in cui non si riscontri la congestione, probabilmente sarà a disposizione una larghezza di banda non utilizzata. Tale incremento viene effettuato ogni volta che riceviamo un nuovo riscontro ed e' pari ad 1MSS, un approccio comune e' l'incremento da parte del mittente TCP di :

$$\text{CongWin} + (\text{CongWin} / \text{MSS} \times E) = \text{CongWin} + \text{MSS} \Rightarrow \text{MSS} / \text{CongWin byte ad ogni riscontro}$$

In questo caso l'incremento di CongWin ha un'andamento lineare, tale fase viene identificata con il nome **Congestion Avoidance**.

#### • Partenza Lenta



Il valore di CongWin inizialmente viene impostato su 1 MSS e quindi in fase iniziale avrà una frequenza di trasmissione pari a MSS/RTT bps, ed un suo incremento lineare avrebbe poco senso in quanto sicuramente la banda a disposizione sarà molto più grande. Perciò TCP applica durante tale fase detta **Partenza Lenta - Slow Start - SS** un approccio basato sull'incremento della frequenza del mittente in modo esponenziale fino al momento in cui si verifica un evento di smarrimento. Tale incremento viene ottenuto andando ad aumentare la finestra di 1 MSS per ogni ricezione di un nuovo riscontro. In pratica ad ogni riscontro seguirà un'invio di un numero doppio di segmenti rispetto al precedente invio.

#### • Reazione agli eventi di timeout

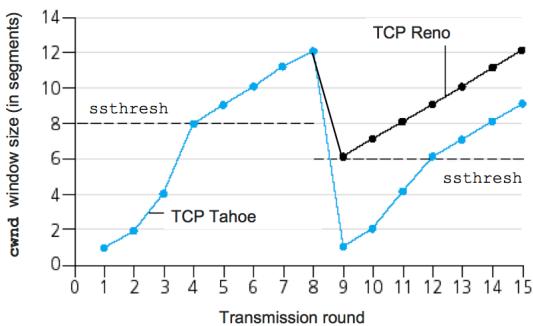
Una caratteristica fondamentale del controllo di congestione e' il fatto che tratta in modo differente l'eventi di timeout e dei 3 ack duplicati.

Nel caso dei 3 ACK duplicati, la finestra di congestione viene dimezzata e poi incrementata linearmente.

Nel caso di un timeout, il mittente entra in fase di slow-start ( CongWin = 1 MSS e poi incrementa in modo esponenziale fino a Congwin/2 e da lì in poi torniamo al normale incremento lineare)

Per gestire tale tematiche TCP utilizza la variabile **Threshold** = soglia della finestra in cui terminerà lo slow-start e inizierà la congestion avoidance. Tale variabile viene inizializzata ad un valore molto grande (65 Kbyte, tale da esser ignorata) e successivamente il suo valore verrà determinato dinamicamente.

### 3. Perche' il mittente TCP si comporta in modo diverso nei due casi di evento di perdita?



TCP assume un comportamento **conservativo** dopo un evento di timeout, ed un comportamento meno conservativo nel caso dei 3 ACK duplicati. Le versioni di TCP Tahoe non faceva distinzione tra i due eventi di perdita, mentre la versione TCP Reno elimina la fase di slow-start dopo la ricezione di 3 ACK duplicati partendo dal presupposto che se ricevo gli ack vuol dire che almeno qualche segmento è arrivato al mittente. Ciò dimostra chiaramente che la rete pur essendo congestionata è sempre in grado di consegnare i segmenti anche se alcuni andranno perduti.

Tale tecnica prende il nome di **Ripristino Veloce - Fast Recovery**.

L'algoritmo TCP Vegas...leggere.

#### 4. Throughput medio di TCP

Andando ad analizzare il throughput medio (frequenza media di trasmissione) di una connessione TCP prolungata ed ignorando le fasi di slow-start si ha:

Se la dim. della finestra vale  $w$  byte e il tempo di andata e di ritorno corrente RTT, la frequenza trasmissiva è approssimabile in  $w/RTT$ . TCP come sappiamo va alla ricerca della banda aggiuntiva aumentando  $w$  di 1MSS ogni RTT fino al verificarsi di un evento di smarrimento identificando con  $W$  il valore di  $w$  in questo caso si ha che  $W/(2 \text{ RTT}) \leq \text{frequenza trasmissiva} \leq W/\text{RTT}$ .

Dato che il throughput cresce in maniera lineare tra i due intervalli si ha che il **Throughput medio di una connessione TCP = ( 0,75 \* W ) / RTT**.

#### Riassunto: il controllo della congestione TCP

- Quando CongWin è sotto la soglia (Threshold), il mittente è nella fase di **partenza lenta**; la finestra cresce in modo esponenziale.
- Quando CongWin è sopra la soglia, il mittente è nella fase di **congestion avoidance**; la finestra cresce in modo lineare.
- Quando si verificano **tre ACK duplicati**, il valore di Threshold viene impostato a CongWin/2 e CongWin viene impostata al valore di Threshold.
- Quando si verifica un **timeout**, il valore di Threshold viene impostato a CongWin/2 e CongWin è impostata a 1 MSS.

| Stato                     | Evento                                                    | Azione del mittente TCP                                                                           | Commenti                                                                                    |
|---------------------------|-----------------------------------------------------------|---------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| Slow Start (SS)           | Ricezione di ACK per dati precedentemente non riscontrati | CongWin = CongWin + MSS,<br>If (CongWin > Threshold)<br>imposta lo stato a "Congestion Avoidance" | CongWin raddoppia a ogni RTT                                                                |
| Congestion Avoidance (CA) | Ricezione di ACK per dati precedentemente non riscontrati | CongWin = CongWin + MSS * (MSS/CongWin)                                                           | Incremento additivo:<br>CongWin aumenta di 1 MSS a ogni RTT                                 |
| SS o CA                   | Rilevato un evento di perdita da tre ACK duplicati        | Threshold = CongWin/2,<br>CongWin = Threshold,<br>imposta lo stato a "Congestion Avoidance"       | Ripristino rapido con il decremento moltiplicativo.<br>CongWin non sarà mai minore di 1 MSS |
| SS o CA                   | Timeout                                                   | Threshold = CongWin/2,<br>CongWin = 1 MSS,<br>imposta lo stato a "Slow Start"                     | Entra nello stato "Slow Start"                                                              |
| SS o CA                   | ACK duplicato                                             | Incrementa il conteggio degli ACK duplicati per il segmento in corso di riscontro                 | CongWin e Threshold non variano                                                             |

Vedi allegato slide.

## Domande e Problemi

1. ....
2. ....
3. ....
4. Perché' uno sviluppatore potrebbe scegliere UDP anziché' TCP?

Le applicazioni che hanno la necessità di un trasferimento affidabile dei loro dati si affidano ovviamente a TCP. Le applicazioni più elastiche riguardo alla perdita dei dati e dipendenti dal tempo si affidano invece a UDP. Inoltre si utilizza UDP per comunicazioni in broadcast (invio a tutti i terminali in una rete locale) e multicast (invio a tutti i terminali iscritti ad un servizio).

|                                                |                                   |                              |
|------------------------------------------------|-----------------------------------|------------------------------|
| Streaming Audio/Video                          | <a href="#">RTSP/RTP</a>          | TCP (comandi) + UDP (flusso) |
| Server di file remoto                          | <a href="#">NFS</a>               | tipicamente UDP              |
| Telefonia su internet ( <a href="#">VoIP</a> ) | <a href="#">SIP, H.323, altri</a> | tipicamente UDP              |
| Gestione della rete                            | <a href="#">SNMP</a>              | tipicamente UDP              |
| Protocollo di routing                          | <a href="#">RIP</a>               | tipicamente UDP              |
| Risoluzione dei nomi                           | <a href="#">DNS</a>               | tipicamente UDP              |

5. Perche' per lo scambio di video e voce e' scelto TCP invece di UDP?

Uno dei motivi di tale scelta e' il fatto che UDP potrebbe esser deprecato in alcune reti ed i segmenti andrebbero persi. Inoltre un'altro motivo potrebbe essere il fatto che TCP crea una connessione tra i due host e grazie a TCP si possono instaurare delle connessioni sicure.  
[Leggi PDF](#).

6. E' possibile che un'applicazione che gira su UDP ottenga un trasferimento affidabile dei dati?

Sì, l'applicazione dovrebbe occuparsi personalmente del controllo della perdita dei pacchetti e determinare i meccanismi più adatti al caso. Con ritrasmissioni dei pacchetti persi e al limite prevedere anche un'ordinamento. Ciò comunque sarebbe molto complicato in quanto l'header UDP e' scarso di informazioni, ne dovremmo aggiungere altre nel campo dati o prevederne un altro.

7. Come fa un processo ad identificare più segmenti ricevuti dalla sua socket UDP provenienti da più host?

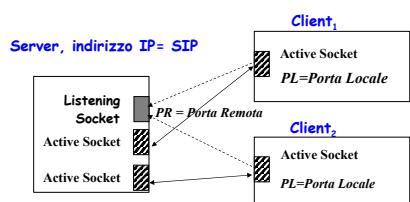
Il processo determina il mittente del pacchetto UDP grazie al preambolo/header IP il quale contiene l'indirizzo IP del mittente.

|                          |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |              |    |    |    |    |    |    |    |    |    |    |
|--------------------------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--------------|----|----|----|----|----|----|----|----|----|----|
| 00                       | 01       | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21           | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Source IPv4 address      |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |              |    |    |    |    |    |    |    |    |    |    |
| Destination IPv4 address |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |              |    |    |    |    |    |    |    |    |    |    |
| 0                        | Protocol |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Total length |    |    |    |    |    |    |    |    |    |    |

## UDP header:

|                    |    |    |    |    |    |    |    |                         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|--------------------|----|----|----|----|----|----|----|-------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00                 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08                      | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| <u>Source Port</u> |    |    |    |    |    |    |    | <u>Destination Port</u> |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| <u>Length</u>      |    |    |    |    |    |    |    | <u>Checksum</u>         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Data :::           |    |    |    |    |    |    |    |                         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

8. Interazione tra server web e piu' host, gli host si collegano tutti sulla porta 80. Tutte le richieste vengono inviate attraverso la stessa socket del server web? se diverse, entrambe hanno al porta 80?



Il principio di funzionamento di un server web implica la presenza di una listening socket ovvero la nostra porta 80, una volta che il server ha accettato una connessione, viene creata una socket dedicata per la connessione con l'host detta Active Socket, la quale verrà usata per l'interazione con l'host e tutti i messaggi da esso spediti vengono diretti automaticamente sul nuovo socket creato.

9. Perché' nel protocollo rdt abbiamo dovuto utilizzare i numeri di sequenza??

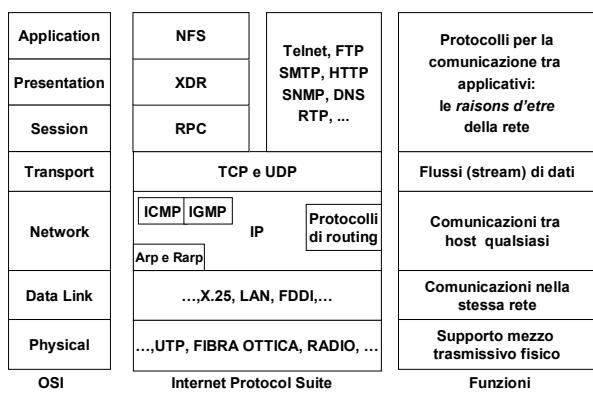
Abbiamo dovuto utilizzare tali numeri per poter andar a determinare un'errore nei pacchetti di ACK e NACK, infatti senza numeri di sequenza non potevamo capire a quale pacchetto si riferivano e quindi validarli.

## Domande e Problemi

- 1.

# Capitolo 4 Livello Network

## Introduzione



Il livello network è incaricato di muovere i pacchetti dalla sorgente fino alla destinazione finale, attraversando tanti sistemi intermedi (router o gateway) della subnet di comunicazione quanti è necessario. In particolare esso fornisce un servizio **Host to Host** (*tra macchine*). Anche se ciò' non e' proprio vero perché' il suo header ha anche un ID per il (processo) protocollo a livello superiore che lo sta usando.

Le incombenze principali di questo livello sono:

- conoscere la tipologia della rete;
- de (multiplexing) e instradamento (routing), scegliere ogni volta il cammino migliore, ciò' viene effettuato dal livello network tramite algoritmi d'intradamento.
- gestire il flusso di dati e le congestioni (flow control e congestion control)

- gestire le problematiche derivanti dalla presenza di più' reti diverse (internetworking).
- forwarding o inoltro, quando un router riceve un pacchetto lo deve trasferire sull'appropriato collegamento d'uscita.

## Inoltro e instradamento

Per il trasferimento dei pacchetti da un host ad un altro possiamo identificare due importati funzioni:

- **Inoltro (forwarding)**, con tale termine indichiamo il fenomeno che ogni router effettua alla ricezione di un pacchetto, ovvero il suo trasferimento sull'appropriato collegamento d'uscita.
- **Intradamento (routing)**, il livello di network deve identificare il percorso che i pacchetti devono seguire e lo fa' tramite l'*algoritmo d'intradamento*.

L'*algoritmo d'inoltro* può' esser di due tipi:

- **Centralizzato**, nel momento in cui si ha un sito centrale che scarica informazioni d'intradamento a ciascuno dei router.
- **Decentralizzato**, quando una parte dell'algoritmo d'intradamento e' distribuito in esecuzione su ogni router.

In entrambi i casi il comportamento dei router e' il medesimo, infatti i messaggi d'intradamento ricevuti vengono utilizzati per la configurazione della **Tabella d'inoltro**.

**Packet Switch (commutatore di pacchetto)** e' un generico dispositivo che si occupa del trasferimento dall'interfaccia in ingresso a quella in uscita in base al valore del campo intestazione nel pacchetto. (Router a livello di rete e commutatori a livello di collegamento).

**Connection Setup (Intradamento della connessione)** alcune architetture a livello network richiedono che i router lungo il cammino tra l'host sorgente e destinazione, effettuino l'handshake tra loro per impostare lo stato prima che i pacchetti iniziano a fluire.

## Modelli di servizi di rete

Definisce le caratteristiche del trasporto end-to-end di dati tra sistema terminale d'invio e quello di ricezione.

Alcuni servizi che il livello network potrebbe offrire sono:

- **Consegna garantita**, in questo caso viene assicurato che prima o poi il pacchetto giunga a destinazione.

- **Consegna garantita con ritardo limitato**, in questo caso viene assicurato la consegna dei pacchetti e il rispetto di un limite di ritardo.

Inoltre ad un **flusso di pacchetti** possono esser offerti i seguenti servizi:

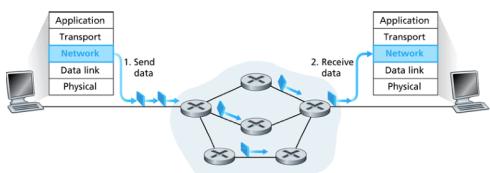
- **Consegna in ordine**, in questo caso viene assicurato che i pacchetti arrivino nell'ordine in cui sono stati inviati.
- **Minima ampiezza di banda garantita**, emula il comportamento di un collegamento trasmissivo con un bitrate specificato tra i due host. Finche' l'host trasmette bit ad una frequenza inferiore del bitrate i pacchetti giungono a destinazione entro un certo ritardo.
- **Jitter limitato**, garantisce che il lasso di tempo tra la trasmissione di due pacchetti consecutivi sia uguale a quello di ricezione.
- **Servizi di sicurezza**, usa una chiave segreta di sessione nota solo alla sorgente e alla destinazione.

Il livello network di Internet mette a disposizione solo un servizio best-effort, ossia con il massimo impegno possibile. Ma esistono due architetture che vanno oltre il best-effort:

- **ATM a bitrate costante**: CBR constant bit rate, banda fissa tra l'host destinazione. Ritardo limitato con poche perdite.
- **ATM a bitrate disponibile**: può smarrire celle (pacchetti) ma l'ordine non può esser alterato e le connessioni garantiscono un minimo tasso trasmissivo (MRC).

| Architettura di Rete | Modello di servizio | Garanzia sulla banda     | Garanzia di consegna | Ordinamento   | Temporizzazione | Indicazione di congestione  |
|----------------------|---------------------|--------------------------|----------------------|---------------|-----------------|-----------------------------|
| Internet             | Best-effort         | Nessuna                  | Nessuna              | Non garantito | Non mantenuta   | No                          |
| ATM                  | CBR                 | Tasso costante garantito | Si                   | Rispettato    | Mantenuta       | Non si verifica congestione |
| ATM                  | ABR                 | Minimo garantito         | Nessuna              | Rispettato    | Non mantenuta   | Si                          |

## Reti a datagramma



Anche il livello network può offrire servizi interessanti o non alla connessione con handshake tra host:

- sono servizi tra host
- un architettura fornisce o un servizio o un altro, non entrambi
- se con connessione anche nei router è implementato.

Le reti di calcolatori che mettono a disposizione un servizio con connessione sono chiamate **reti a circuito virtuale** mentre quelle che offrono un servizio senza connessione sono dette **reti a datagramma**.

| <p>Diagram illustrating packet forwarding through a network of routers. A packet with address 0111 enters a router. The router consults its local forward table to determine the output port (e.g., port 1). The packet then continues through the network.</p>                                                                                                                                                                                                                                                                                                  | <p>Quando vogliamo inviare un pacchetto tramite una rete a datagramma prima contrassegna con l'indirizzo del destinatario e lo immette nella rete. I pacchetti passano attraverso i router che guardano l'indirizzo e decidono in quale collegamento mandarli grazie alla tabella d'inoltro.</p>                                                   |          |             |             |          |          |       |   |          |          |          |   |          |          |       |   |            |  |  |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|-------------|-------------|----------|----------|-------|---|----------|----------|----------|---|----------|----------|-------|---|------------|--|--|---|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #0070C0; color: white;"> <th colspan="3">Corrispondenza di un prefisso</th> <th>Interfaccia</th> </tr> </thead> <tbody> <tr> <td>11001000</td> <td>00010111</td> <td>00010</td> <td>0</td> </tr> <tr> <td>11001000</td> <td>00010111</td> <td>00011000</td> <td>1</td> </tr> <tr> <td>11001000</td> <td>00010111</td> <td>00011</td> <td>2</td> </tr> <tr> <td colspan="3" style="text-align: center;">Altrimenti</td><td>3</td> </tr> </tbody> </table> | Corrispondenza di un prefisso                                                                                                                                                                                                                                                                                                                      |          |             | Interfaccia | 11001000 | 00010111 | 00010 | 0 | 11001000 | 00010111 | 00011000 | 1 | 11001000 | 00010111 | 00011 | 2 | Altrimenti |  |  | 3 | <p>Nella tabella d'inoltro non serve che ci sia una riga per ogni indirizzo, ma possiamo indicare un range di indirizzi.</p> <p>Il router confronta un <b>Prefisso</b> dell'indirizzo destinazione del pacchetto con una riga della tabella. Quando si verificano corrispondenze multiple usiamo le <b>regole di corrispondenza a prefisso più lungo</b>. Inoltre in questi tipi di rete i router non conservano informazioni sullo stato della connessione, ma quelle sullo stato d'inoltro.</p> |
| Corrispondenza di un prefisso                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                                    |          | Interfaccia |             |          |          |       |   |          |          |          |   |          |          |       |   |            |  |  |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 11001000                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 00010111                                                                                                                                                                                                                                                                                                                                           | 00010    | 0           |             |          |          |       |   |          |          |          |   |          |          |       |   |            |  |  |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 11001000                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 00010111                                                                                                                                                                                                                                                                                                                                           | 00011000 | 1           |             |          |          |       |   |          |          |          |   |          |          |       |   |            |  |  |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 11001000                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 00010111                                                                                                                                                                                                                                                                                                                                           | 00011    | 2           |             |          |          |       |   |          |          |          |   |          |          |       |   |            |  |  |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Altrimenti                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                                                                                                                                                                                                                                                                                                                                                    |          | 3           |             |          |          |       |   |          |          |          |   |          |          |       |   |            |  |  |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <p><b>Internet (datagrammi)</b></p> <ul style="list-style-type: none"> <li>- Necessità di scambiare dati tra differenti calcolatori</li> <li>- Servizi elasticci, non vi sono eccessivi requisiti di tempo</li> <li>- L'interconnessione è semplice (computer)</li> <li>- È adattabile, effettua controlli e recupera errori</li> <li>- Rete interna non complessa, la complessità sta agli estremi</li> <li>- Svariati tipi di link</li> <li>- Caratteristiche differenti</li> <li>- Difficile uniformarne il servizio</li> </ul>                               | <p><b>ATM (CV)</b></p> <p>Deriva dal mondo della telefonia</p> <p>Conversazione telefonica:</p> <ul style="list-style-type: none"> <li>- Requisiti stringenti in termini di tempo e affidabilità</li> <li>- Necessità di servizi garantiti</li> <li>- Sistemi terminali "stupidi" Telefoni</li> </ul> <p>La complessità sta nella rete interna</p> |          |             |             |          |          |       |   |          |          |          |   |          |          |       |   |            |  |  |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

## IP Service Model: Perchè Best-Effort?

IP vuol dire non dover mai dire "mi spiace" ...

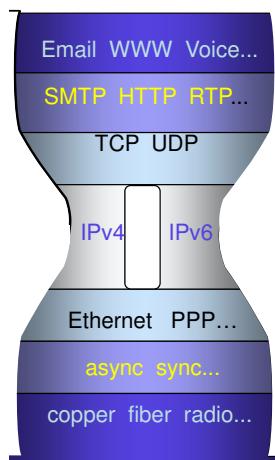
- Non necessita di riservare banda e memoria
- Non necessita di fare error detection e/o correction
- Non ha bisogno di ricordarsi che i pacchetti sono tra loro legati

Più facile sopravvivere a guasti

- Si tollerano perdite temporanee durante la fase di failover

... ma, le applicazioni vogliono efficienza, un trasferimento accurato, dati in ordine, in tempi ragionevoli

- Assenza di error detection o correction
  - Protocolli di più alto livello possono fare l'error checking
- Packets successivi possono fare percorsi diversi: Non è un problema basta che i packets arrivino a destinazione
- Packets possono essere consegnati out-of-order: Il Receiver può ordinare i packets (se necessario)
- Packets possono andare perduti o subire ritardi arbitrari: Il Sender può rispedire i packets (se desiderato)
- Non c'è un controllo di congestione (escluso il "drop"): Il Sender può rallentare a seguito di perdita o di ritardo



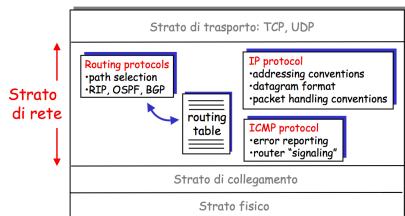
L'introduzione di una transizione in IP : da V4 a V6

Raddoppia il numero delle interfacce di servizio

Richiede cambiamenti sopra e sotto il livello IP

Crea sottili (neanche tanto sottili) problemi di interoperabilità

## Protocollo Internet (IP)



Prima di iniziare la trattazione andiamo a considerare quali sono i componenti di Internet a livello network.

|      | Bits 0-3 | 4-7                    | 8-15                                   | 16-18 | 19-31           |
|------|----------|------------------------|----------------------------------------|-------|-----------------|
| 0    | Version  | Internet Header length | Type of Service (now DiffServ and ECN) |       | Total Length    |
| 32   |          | Identification         |                                        | Flags | Fragment Offset |
| 64   |          | Time to Live           | Protocol                               |       | Header Checksum |
| 96   |          |                        | Source Address                         |       |                 |
| 128  |          |                        | Destination Address                    |       |                 |
| 160  |          |                        | Options (optional)                     |       |                 |
| 160  |          |                        | Data                                   |       |                 |
| 192+ |          |                        |                                        |       |                 |

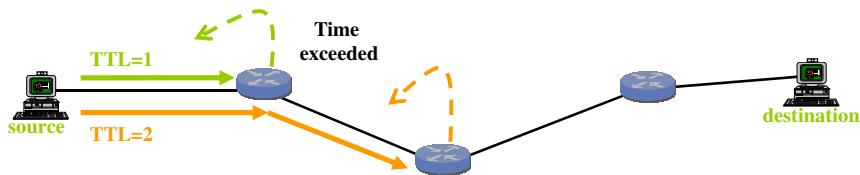
## Formato dei datagrammi

Un pacchetto a livello network viene detto *datagramma*.

- **Versione [4 bit]** - Indica la versione del datagramma IP: per IPv4, ha valore 4 (da qui il nome IPv4).
- **Internet Header Length (IHL) [4 bit]** - Indica la lunghezza (in parole da 32 bit) dell'header del datagramma IP ovvero l'offset del campo dati; tale lunghezza può variare da 5 parole (20 byte) a 15 parole (60 byte) a seconda della presenza e della lunghezza del campo facoltativo Options;
- **Type of Service (TOS) [8 bit]** - Nelle specifiche iniziali del protocollo ([RFC 791](#)), questi bit servivano all'host mittente per specificare il modo e in particolare la precedenza con cui l'host ricevente doveva trattare il datagramma. Nella pratica questo uso del campo TOS non ha preso largamente piede. Dopo molte sperimentazioni e ricerche, recentemente questi 8 bit sono stati ridefiniti ed hanno la funzione di [Differentiated services \(DiffServ\)](#) nell'[IETE](#) e [Explicit Congestion Notification \(ECN\)](#) codepoints (vedi [RFC 3168](#)), necessari per le nuove

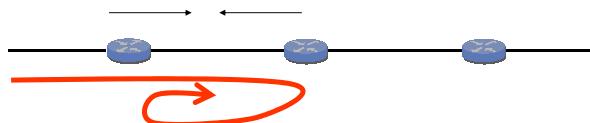
tecnologie basate sullo streaming dei dati in tempo reale, come per il esempio il [Voice over IP \(VoIP\)](#) usato per lo scambio interattivo dei dati vocali. Lo specifico livello di servizio e' determinato dall'amministratore del router.

- **Total Length [16 bit]** - Indica la dimensione (in byte) dell'intero datagramma, comprendendo header e dati; tale lunghezza può variare da un minimo di 20 byte (header minimo e campo dati vuoto) ad un massimo di 65535 byte. In ogni momento, ad ogni host è richiesto di essere in grado di gestire datagrammi aventi una dimensione minima di 576 byte mentre sono autorizzati, se necessario, a frammentare datagrammi di dimensione maggiore.
- **Identification [16 bit]** - Utilizzato, come da specifiche iniziali, per identificare in modo univoco i vari frammenti in cui può essere stato "spezzato" un datagramma IP. Alcune sperimentazioni successive hanno però suggerito di utilizzare questo campo per altri scopi, come aggiungere la funzionalità di tracciamento dei pacchetti.
- **Flags [3 bit]** - Bit utilizzati per il controllo del protocollo e della frammentazione dei datagrammi:
  - Reserved - sempre settato a 0.
  - **DF** (Don't Fragment) - se settato a 1 indica che il datagramma non deve essere frammentato; se tale datagramma non può essere inoltrato da un host senza essere frammentato, viene semplicemente scartato. Questo può risultare utile per "ispezionare" la capacità di gestione dei vari host del percorso di [routing](#).
  - **MF** (More Fragments) - se settato a 0 indica che il datagramma è l'ultimo frammento o il solo frammento del datagramma originario, pertanto tutti gli altri suoi frammenti hanno il bit MF settato a 1. Naturalmente, questo bit sarà sempre 0 anche in tutti i datagrammi che non sono stati frammentati.
- **Fragment Offset [13 bit]** - Indica l'offset (misurato in blocchi di 8 byte) di un particolare frammento relativamente all'inizio del datagramma IP originale: il primo frammento ha offset 0. L'offset massimo risulta pertanto pari a  $(2^{13} - 1) \times 8 = 65528$  byte che, includendo l'header, potrebbe eccedere la dimensione massima di 65535 byte di un datagramma IP.
- **Time To Live (TTL) [8 bit]** - Indica il tempo di vita ([time to live](#)) del datagramma, necessario per evitarne la persistenza indefinita sulla rete nel caso in cui non si riesca a recapitarlo al destinatario. Storicamente il TTL misurava i "secondi di vita" del datagramma, mentre ora esso misura il numero di "salti" da nodo a nodo della rete: ogni router che riceve il datagramma prima di inoltrarlo ne decremente il TTL (modificando di conseguenza anche il campo Header Checksum), quando questo arriva a zero il datagramma non viene più inoltrato ma scartato. Tipicamente, quando un datagramma viene scartato per esaurimento del TTL, viene automaticamente inviato un messaggio [ICMP](#) al mittente del datagramma, specificando il codice di Richiesta scaduta; la ricezione di questo messaggio ICMP è alla base del meccanismo di [traceroute](#).



#### • Potenziale Problema

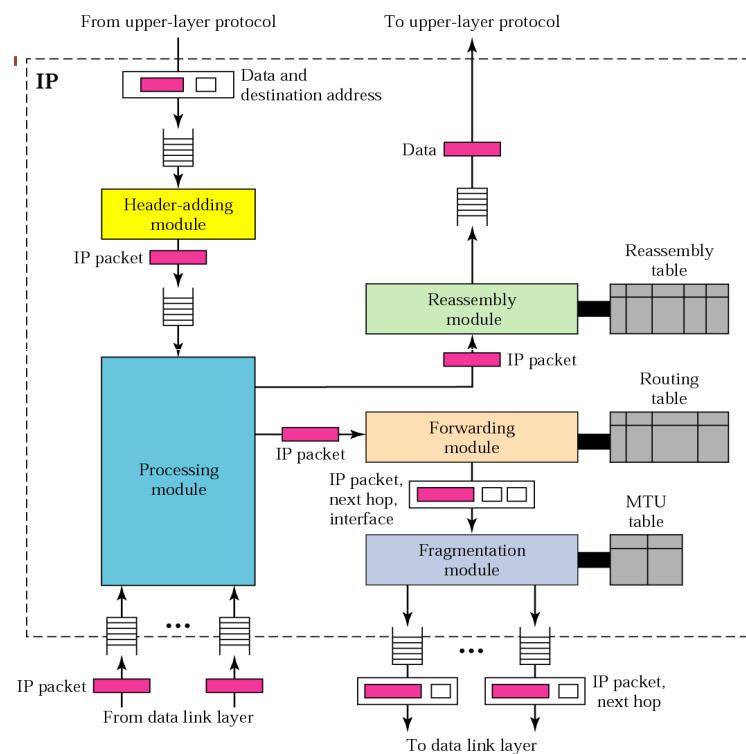
- Un forwarding loop puo' causare packets che girano per sempre
- Confusione se il packet arriva con ritardo eccessivo



- **Protocol [8 bit]** - Indica il codice associato al [protocollo](#) utilizzato nel campo dati del datagramma IP: per esempio al protocollo [TCP](#) è associato il codice 6, ad [UDP](#) il codice 17, mentre ad [IPv6](#) è associato il codice 41.

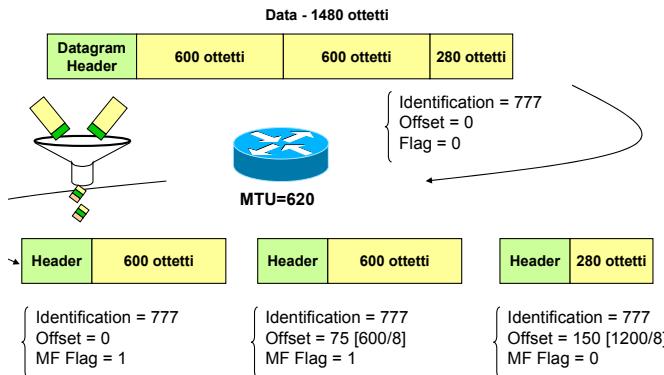
- **Header Checksum [16 bit]** - È un campo usato per il controllo degli errori dell'header. Ad ogni hop, il checksum viene ricalcolato (secondo la definizione data in [RFC 791](#)) e confrontato con il valore di questo campo: se non c'è corrispondenza il pacchetto viene scartato. È da notare che non viene effettuato alcun controllo sulla presenza di errori nel campo Data deputandolo ai livelli superiori.
- **Source address [32 bit]** - Indica l'indirizzo IP associato all'host del mittente del datagramma. Da notare che questo indirizzo potrebbe non essere quello del "vero" mittente nel caso di traduzioni mediante [NAT](#). Infatti, qualora un host intermedio effettui questa traduzione, sostituisce l'indirizzo del mittente con uno proprio, procurandosi poi di ripristinare l'indirizzo originario su tutti i messaggi di risposta che gli arrivano destinati al mittente originario.
  - Perchè qualcuno dovrebbe fare ciò ?
  - Lanzio di un attacco denial-of-service
    - Spedire un eccessivo numero di packets alla destinazione
    - ...sovrafficare il nodo,o i links che portano al nodo
  - Evitare di essere intercettati (by "spoofing")
    - Infatti, la vittima vi identificherebbe dal source address
    - ...ecco perchè mettere il source address di qualcun altro nei packets
  - Oppure, scatenare un attacco contro lo 'spoofed host'
    - Lo spoofed host è erroneamente accusato
    - Lo spoofed host riceve il traffico di ritorno dal receiver
- **Destination address [32 bit]** - Indica l'indirizzo IP associato all'host del destinatario del datagramma e segue le medesime regole del campo Source address.
- **Options** - Opzioni (facoltative e non molto usate) per usi più specifici del protocollo.

Si ricorda che il valore del campo IHL deve essere sufficientemente grande da includere anche tutte le opzioni e, nel caso queste siano più corte di una word, il padding necessario a completare i 32 bit. Inoltre, nel caso in cui la lista di opzioni non coincida con la fine dell'header, occorre aggiungere in coda ad essa un marcatore EOL (End of Options List). C'è da notare infine che, potendo causare problemi di sicurezza, l'uso delle opzioni LSSR e SSRR (Loose e Strict Source and Record Route) è scoraggiato e molti router bloccano i datagrammi che contengono queste opzioni.



## Frammentazione dei datagrammi IP

Non tutti i protocolli a livello di collegamento possono trasportare pacchetti dello stessa dimensione a livello di rete. Es: Ethernet = 1500 byte , Collegamento di grandi distanze = 576 byte.



La massima quota che un frame può inviare viene chiamata **MTU (maximum transmission unit)**. Dato che i datagrammi IP sono incapsulati in frame, la MTU pone un limite alla lunghezza dei datagrammi. Il problema non e' la MTU, ma la possibilità che sul percorso il datagramma può incontrare protocolli diversi con diverse MTU. Per ovviare a questo problema, si frammentano i dati del datagramma in datagrammi più piccoli chiamati frammenti. I quali verranno ricompattati prima di esser passati al livello di trasporto; tale scopo secondo i progettisti di ipV4 aspetta ai sistemi terminali.

Per permettere questo ricompattamento, nell'header di IP abbiamo i campi *ID, flag, Offset*.

Quando un'host crea un datagramma, lo contrassegna con un numero identificativo e con gli indirizzi di origine e destinazione. Quando poi un router lo frammenta, lascia invariati questi tre campi, ma modifica il flag e l'offset. Per identificare l'ultimo segmento mettiamo il bit flag a 0. Se qualche pacchetto va' perduto viene scartato anche il datagramma.

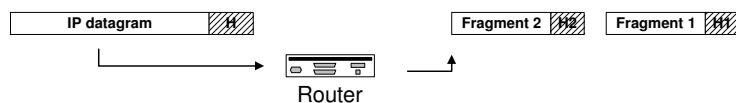
- *IPv4*: il router deve spezzare i segmenti
- *IPv6*: ICMP comunica al mittente che il datagramma e' troppo grande, il mittente deve perciò dividerlo e rimandarlo.

La frammentazione rende i router e i sistemi terminali più complessi e può esser usata per attacchi DoS, mandando segmenti con offset mai nullo o segmenti con offset che non si allineano mai perfettamente. Inoltre

vi sono altre problematiche legate alla frammentazione :

- Maggiore overhead di trasmissione ( la perdita di un frammento invalida l'intero datagramma, e un maggior numero di byte per gli headers)
- Impegna risorse nell'host ricevente (timer e buffer).

La frammentazione viene fatta o dal sender o dai router intermedi, lo stesso datagramma può essere frammentato più volte e il riassemblaggio dei datagrammi originali solo al destination host.



## Campi coinvolti nella Frammentazione

| version            | header length | DS       | ECN             | total length (in bytes) |  |  |
|--------------------|---------------|----------|-----------------|-------------------------|--|--|
| Identification     |               | 0        | D M             | Fragment offset         |  |  |
| time-to-live (TTL) |               | protocol | header checksum |                         |  |  |
|                    |               |          |                 |                         |  |  |

**Indication:** Quando un datagramma è frammentato, l'*identification* è lo stesso per tutti i frammenti

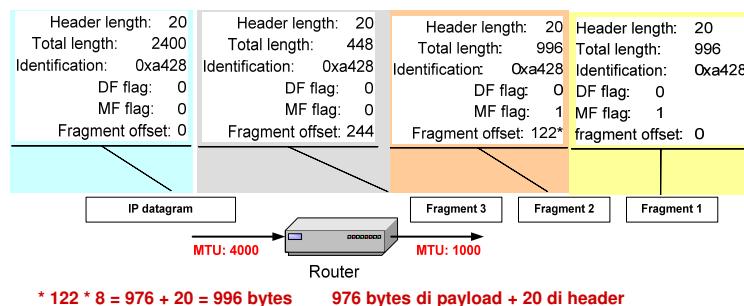
### Flags:

- **DF dit set**: Datagramma non puo' essere frammentato e deve esser

scartato se MTU e' piccola.

- **MF bit set:** Datagramma parte di un frammento e altri frammenti sono in arrivo.

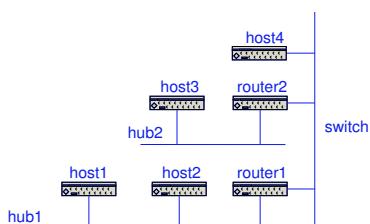
### Un datagramma da 2400 bytes deve essere frammentato perché la MTU ha un limite di 1000 bytes



## Indirizzamento IPv4

Generalmente un Host ha un solo collegamento verso la rete; quando l'IP deve inviare dati lo fa attraverso questo collegamento. Il confine tra Host e collegamento fisico viene detto *Interfaccia*. Un router ha almeno 2 collegamenti, e 2 differenti IP.

*In particolare si ha che l'IP e' assegnato all'interfaccia anziché all'host o router.*



Gli indirizzi IP sono lunghi 32 bit, perciò ce ne sono  $2^{32} = 4$  mld. In genere sono scritti in notazione decimale puntata. 8 bit . 8bit . 8 bit . 8bit.

Ogni interfaccia di Host o Router ha un indirizzo IP globalmente univoco e non scelti in modo arbitrario, ma una parte dell'indirizzo di un interfaccia e' determinato dalla sottorete a cui e' collegata.

**Sottorete**, e' una rete i cui punti terminali sono collegati ad un interfaccia di un host o di un router.

La strategia di assegnazione degli indirizzi Internet e' detta **Interdomain Routing CIDR** il quale generalizza la nozione d'indirizzamento di sottorete. Generalmente l'indirizzo IP viene diviso in 2 parti e mantiene la forma decimale puntata  $a.b.c.d / x$  dove x indica il numero di bit nella prima parte dell'indirizzo (*Prefisso di rete*).

Esempio con  $x = 23$  [ subnet part (23 bit) | host part (32 - 23 bit) ]

Appartiene alla sottorete??... da completare

I router esterni alla sottorete considerano per l'inoltro solamente gli  $X$  bit del prefisso così da ottenere una riduzione delle tabelle di routing perché una sola riga  $a.b.c.d / x$  e' sufficiente per far arrivare i dati alla sottorete. I rimanenti 32 - X bit vengono utilizzati per distinguere gli host all'interno della sottorete, i quali verranno utilizzati dai router interni della sottorete per inviare i dati al destinatario.

Prima del CIDR l'indirizzamento era Classfull Addressing, con sotto reti:

- di classe A: 8
- di classe B : 16 bit , 65534 host **Troppi**
- di classe C : 24 bit , ospitano sono 254 host ( 2 per usi speciali) **Pochi**

**IP Broadcast** 255.255.255.255 Il messaggio viene consegnato a tutti gli host della sottorete. ( i router potrebbero mandarlo alle sotto reti confinanti)

## Come ottenere un blocco di indirizzi IP

Un amministratore di rete deve controllare il suo ISP. Ad esempio all'ISP sono stati dati il blocco d'indirizzi 200.23.26.0 / 20, potrebbe dividere il blocco in 8 sotto reti uguali con indirizzi contigui ed assegnare ogni blocco ad una diversa organizzazione.

Blocco ISP

Organizzazione 1...

Contattare l'ISP non e' l'unico modo per ottenere un blocco d'indirizzi, inoltre l'ISP a sua volta deve richiedere un blocco d'indirizzi all'autorita' globale che ha il compito di gestire lo spazio d'indirizzamento IP e i blocchi: ICANN la quale si occupa inoltre della gestione dei server radice DNS e di assegnare e risolvere dispute sui nomi di dominio.

Vedi allegato 2.

## Come ottenere l'indirizzo IP di un host: DHCP (protocollo Dinamico di configurazione degli Host)

L'organizzazione può assegnare individualmente gli IP del blocco dell'interfaccia di host e router nella propria struttura. L'amministratore di rete in genere configura manualmente gli indirizzi ip nel router, mentre l'assegnazione di quelli degli host viene delegata al DHCP (Dynamic Host Configuration Protocol), esso permette ad ogni host di ricevere un IP e altre informazioni:

- la sua maschera di sottorete
- l'indirizzo del suo router di primo hop (gateway)
- l'indirizzo DNS locale

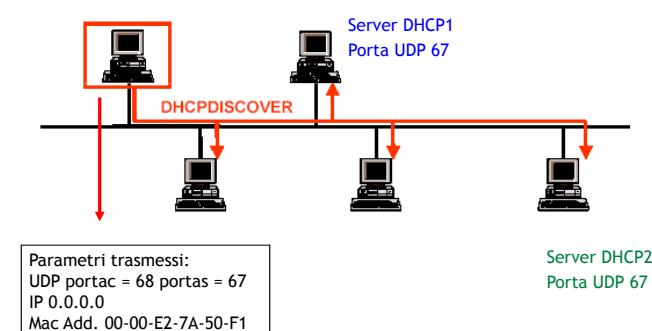
DHCP può esser configurato in modo da assegnare un indirizzo IP persistente ogni volta che l'host si connette oppure in modo da assegnare un indirizzo temporaneo che sarà sempre diverso. Il DHCP viene detto *Plug and Play* per la sua capacita' di automatizzare le connessioni.

Quando gli host entrano ed escono dalla rete, DHCP aggiorna la sua tabella degli IP disponibili. Inoltre e' un protocollo Client Server, client nel momento in cui un host decide di ricevere info sulla configurazione della rete.

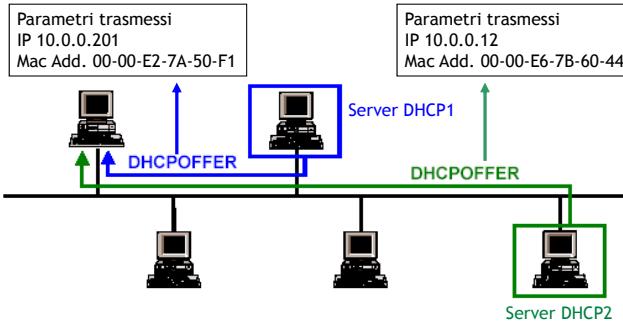
### Ambiti di Utilizzo:

- Non sappiamo a priori quanti client ci siano in rete.
- Si hanno portatili sulla rete
- Si vogliono fornire dinamicamente anche altre info( DNS,Gateway,ipr..)
- Si puo' dedicare una macchina a tale servizio

### Ip Lease Discover & Offert



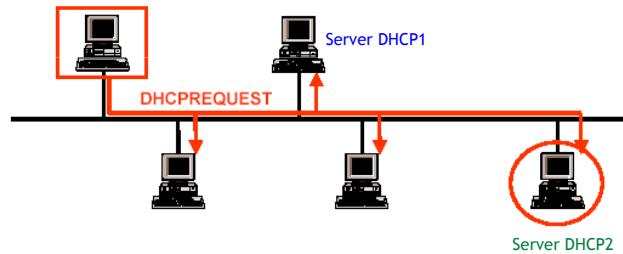
Il client invia DHCPDISCOVER, un messaggio broadcast utilizzato dal client per richiedere i parametri di configurazione del server DHCP



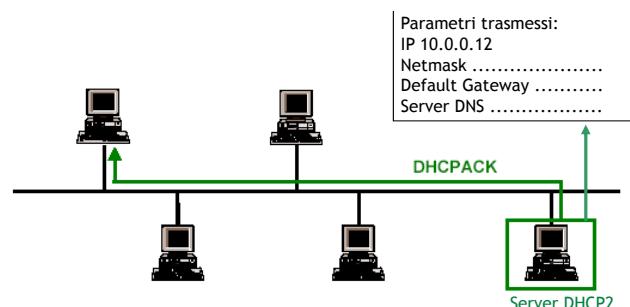
Ogni server risponde con un **DHCPOFFER**, un messaggio utilizzato dal server DHCP per offrire un indirizzo IP ai client che li richiedono.

Note:

- Ogni server invia il suo Mac Address poiché è l'unico identificativo valido in questa fase.
- Utilizzando il Mac Address ricevuto da ogni server, il client può rispondere al server
- Ogni server DHCP ha un suo range di Address assegnabili.

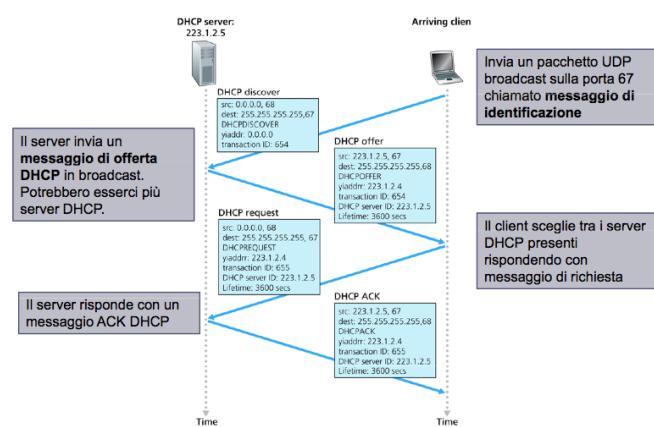


Il client invia **DHCPREQUEST**, un messaggio utilizzato dal client per accettare o rinnovare l'assegnazione di un indirizzo IP, specificando il DHCP server.



**DHCPACK**, messaggio utilizzato dal server DHCP per riconoscere l'accettazione da parte di un client di un indirizzo IP offerto specificando i parametri di configurazione.

Nel caso più semplice ogni sottorete ha un suo DHCP, altrimenti c'è un router (agente di appoggio di DHCP) che conosce l'indirizzo di un server DHCP per quella rete.



Possono arrivare più di un'offerta, quindi ci vogliono altre 2 interazioni, quella in cui dice la scelta e la conferma del DHCP. L'IP non scelto viene riservato per un'altra macchina.

Dato che un client potrebbe voler utilizzare il proprio indirizzo IP oltre alla durata della connessione, il DHCP fornisce un meccanismo di connessione ai client di rimuovere le connessioni di un indirizzo IP.

Step per l'acquisizione di un indirizzo IP tramite un server DHCP:

- **Individuazione del DHCP server**, in questa fase il client DHCP invia un *messaggio d'identificazione DHCP*, tale messaggio viene inviato tramite un pacchetto UDP attraverso la porta 67. Tale datagramma viene inviato in broadcast con ip destinazione "0.0.0.0";

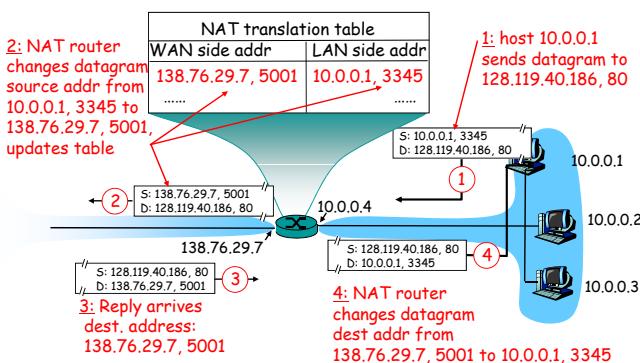
- **Offerte del server DHCP**, uno dei server DHCP della rete risponde con un *messaggio di offerta DHCP* che viene inviato in broadcast a tutti i nodi della sottorete con IP broadcast 255.255.255.255. Il client si potrebbe trovare a scegliere quale IP e da quale server, quindi ogni messaggio di offerta è caratterizzato da un ID di transazione, l'indirizzo IP offerto e la durata dell'offerta;

- **Richiesta DHCP**, una volta che il client ha scelto il sever, gli risponderà con un messaggio di richiesta DHCP che riporta i parametri di configurazione.
- **Conferma DHCP**, il server gli risponde con un messaggio di ACK DHCP.

## Traduzione degli indirizzi di rete: Nat (Network Address Translation)

Una rete locale ha un numero limitato di indirizzi IP pubblici assegnati dal provider. Tutte le macchine possono comunicare simultaneamente su Internet sfruttando un indirizzo IP pubblico grazie al metodo del Network Address Translation (NAT).

Il funzionamento del NAT risiede in un dispositivo di routing, che traduce gli indirizzi IP sorgenti dei pacchetti interni; l'IP interno viene cioè scartato ed al suo posto viene inserito l'indirizzo IP pubblico dell'interfaccia esterna del router. In questo modo i pacchetti possono essere inoltrati esternamente e gestiti dai sistemi di destinazione; il server remoto avrà così la possibilità di recapitare a destinazione le risposte.



I router abilitati al NAT all'esterno non appaiono come router, ma come un dispositivo con un unico IP. Questo router nasconde i dettagli della rete domestica al resto del mondo. Il router Nat ottiene un IP dal DHCP dell'ISP e mette in esecuzione un server DHCP che fornisce gli IP ai dispositivi della rete interna.

**Indirizzi IP pubblici**, vengono, in genere assegnati in modo dinamico, cioè cambiano ad ogni connessione. In alcuni casi è necessario fornirsi di un indirizzo IP statico.

**Indirizzi IP privati**, Questi numeri possono essere assegnati in piena autonomia alle macchine che fanno parte di una rete privata e permetteranno a tutte le macchine di utilizzare localmente i protocolli TCP/IP, ma i calcolatori a cui sono assegnati questi indirizzi saranno "nascosti" ad Internet e non potranno essere visti se non dalle macchine che si trovano sulla stessa rete privata.

| Numero di indirizzi | dall'indirizzo | all'indirizzo   |
|---------------------|----------------|-----------------|
| 16 milioni          | 10.0.0.0       | 10.255.255.255  |
| 65 mila             | 192.168.0.0    | 192.168.255.255 |
| 4096                | 172.16.0.0     | 172.31.0.255    |

La struttura principale è la **Tabella di traduzione NAT** la quale associa ad ogni host che vuol comunicare con l'esterno una riga nella quale sono contenuti l'indirizzo IP della rete interna dell'host e il numero di porta su cui esso ha inviato il messaggio.

Ad esempio se C deve inviare una richiesta ad un server web con IP 128.119.40.186, invia un messaggio con IP mittente 10.0.0.3 e porta abbinata 3345. Il router NAT quando riceve il messaggio gli assegna una porta tra quelle ancora non nella tabella di traduzione (es.5001) e sostituisce nel messaggio l'IP del Mittente con il suo IP e la porta con la 5002. Poi crea un'associazione nella tabella NAT.

| Tabella di traduzione NAT |                     |
|---------------------------|---------------------|
| Lato WAN                  | Lato LAN            |
| 137.76.29.7:80 (TCP)      | 10.0.0.1:80 (TCP)   |
| 137.76.29.7:2687 (TCP)    | 10.0.0.1:1165 (TCP) |
| 137.76.29.7:5001 (UDP)    | 10.0.0.3:3345 (UDP) |
| ...                       | ...                 |

Quando un pacchetto arriva dall'esterno il funzionamento è analogo, con l'uso di NAT "**cade**" il discorso del collegamento end-to-end. Le porte vengono usate per individuare l'host anziché l'applicazione. Esso viene considerato come una pezza al problema della mancanza d'indirizzi IP, quando invece la soluzione standard sarebbe IPv6. Ma nonostante ciò i router NAT sono ormai un'importante componente di Internet.

Tali router introducono delle problematiche nei protocolli P2P e nella condivisione di file e di Voice over IP, alcune risolte utilizzando l'espeditore conosciuto con il nome *Inversione di connessione* (*connection reversal*), ovvero un host che utilizza il nat, riesce a comunicare con un altro tramite un terzo che si interpone tra di loro.

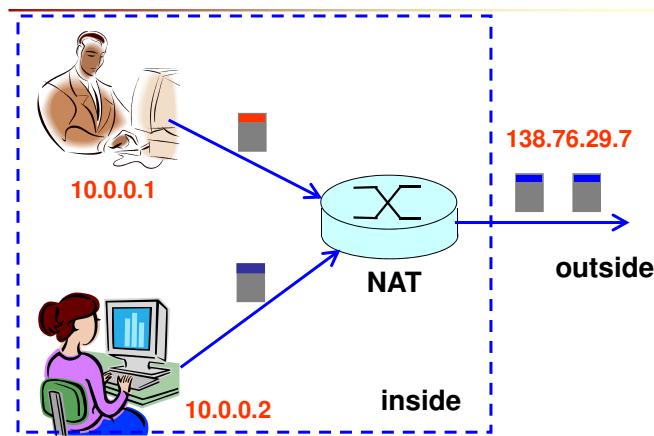
**Nat statico**, consiste in una mappatura biunivoca (1:1) tra indirizzi IP non registrati e indirizzi IP registrati. Risulta particolarmente utile quando è necessario rendere accessibile un dispositivo alla rete esterna.

**Nat dinamico**, mappa un indirizzo IP registrato preso in un gruppo di indirizzi IP registrati e disponibili.

## Tabelle di Natting

Quando un pacchetto di dati, proveniente dalla rete privata, viene inviato verso l'esterno e' costretto ad attraversare in NAT dove il pacchetto viene modificato assumendo l'indirizzo del NAT stesso e contemporaneamente viene aggiornata una tabella di attraversamento.

Quando giunge il messaggio di risposta, relativo al pacchetto inviato, viene consultata la tabella di attraversamento e, in base a questa, il NAT individua il calcolatore della rete privata che ha fatto l'interrogazione.



Cio' ci porta ad utilizzare un **Singolo Source Address**.

### Casi limite:

- Supponiamo che i due hosts contattino la stessa destinazione
- E.g., entrambi aprono una socket con local port 3345 alla destinazione 128.119.40.186 sulla porta 80
- il router-NAT assegna ai packets lo stesso source address
- Quindi tutti i packets hanno source address 138.76.29.7
- Problemi
- Può la destinazione discriminare tra i senders ?
- Può ritornare il traffico all'host corretto ?

## Port-Translating NAT:

- *Mappaggio dei pacchetti in uscita*
  - Rimpiazzare source address con *NAT address*
  - Rimpiazzare source port number con *new port number*
  - Remote host risponde usando (*NAT address*, *new port #*)
- *Mantenere una translation table*
  - (source address, port #) => (*NAT address*, *new port #*)
- *Mappaggio dei pacchetti in ingresso*
  - Consultare la translation table
  - Sostituire destination address e port number
  - L'Host locale riceve il pacchetto correttamente

## Vantaggi

- *Sicurezza*: Crea automaticamente un firewall tra la rete interna e quella esterna
- *Accesso a Internet*: è sufficiente un solo indirizzo IP non mascherato per "far vedere la rete" a decine di macchine. Non fa alcuna differenza in questo caso che l'indirizzo IP sia fisso o assegnato dinamicamente dal provider, in quanto la tabella di instradamento viene azzerata e ricostruita ad ogni nuova connessione alla rete

## Svantaggi

- Nessun *tracking* delle informazioni relative ai protocolli utilizzati se non il numero di porta.
- Non si ha un reale controllo su chi può far partire le connessioni dall'interno della rete verso l'esterno, se non definendo nelle access list quali host possono eseguire queste operazioni di connessione

## Considerazioni



- Il NAT è talvolta confuso con il Proxy Server; il NAT è trasparente sia per il computer sorgente che per quello di destinazione, un Proxy Server invece non è trasparente, il computer sorgente sa che sta inoltrando una richiesta al Proxy Server e deve essere configurato opportunamente.
- Inoltre i Proxy Server lavorano normalmente a partire dal livello 4 del modello di riferimento OSI mentre il NAT lavora a livello 3.
- Il Proxy Server lavorando a livelli OSI più alti rispetto al NAT risulta generalmente più lento

## UPnP

E' un protocollo che consente a un host di individuare e configurare un NAT vicino, con UPnp un'applicazione in esecuzione su un host può richiedere una corrispondenza NAT tra (ip int, port int) e (ip ext , porta ext) per un qualsiasi numero di porta. UPnP consente all'applicazione di conoscere IP ext e porta et in modo che le possano comunicare al mondo esterno.

Inoltre UPnP favorisce un'efficiente e robusta soluzione per l'attraversamento del NAT nei protocolli P2P, permettendo ad un'host esterno di iniziare sessioni di comunicazione con host con NAT usando sia TCP che UDP,

foto esempio torrent

## ICMP

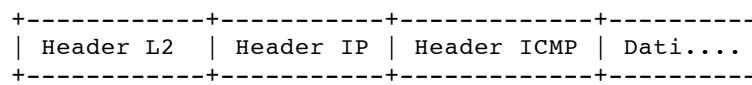
*Viene usato da host e router per scambiarsi informazioni a livello di rete. Il suo uso più tipico e' la notifica di errori e l'uso per i ping.*

Es "Destinazione non raggiungibile" ha origine in ICMP. Identifica il fenomeno che avviene quando un router da qualche parte non e' stato in grado di trovare un percorso verso l'host specificato ed ha composto e d inviato al nostro host un messaggio ICMP.

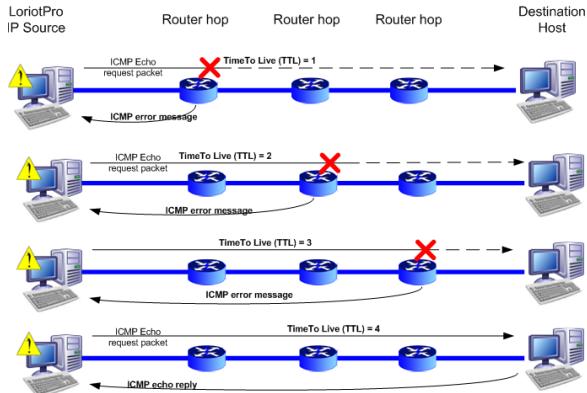
Spesso ICMP e' considerato parte di IP ma in realtà usa IP per inviare i suoi messaggi */C/M/P -> I/P -> D/L*. I messaggi ICMP hanno un campo tipo e un campo codice e contengono i primi 8 byte e l'intestazione del datagramma che ha provocato la generazione del messaggio, in modo che il mittente sappia quale datagramma ha generato l'errore.

| Bits | 0-7            | 8-15 | 16-23    | 24-31 |
|------|----------------|------|----------|-------|
| 0    | Type           | Code | Checksum |       |
| 32   | Rest of Header |      |          |       |

Un messaggio ICMP viene incapsulato in IP:



## TraceRoute



Quando l'n-esimo pacchetto arriva all'n-esimo router e' scaduto (TTL). Questo lo scarta e invia all'origine un messaggio ICMP 11 0. TraceRoute si ferma quando gli giunge un ICMP 3 3 (porta non raggiungibile) Così l'host sa' quanti e quali sono i router che formano la sequenza dalla destinazione e il RDT.

## IPv6

i motivi per cui fu definito un nuovo protocollo IP sono essenzialmente 3:

- Esaurimento degli indirizzi IPv4 a 32 bit, IPv6 a 128 bit.
- Problemi legati alla frammentazione
- L'introduzione del concetto di flusso

Oltre a unicast e multicast, anycast permette di consegnare un datagramma a un qualsiasi host all'interno di un gruppo [ GETHTTP tra server mirror ]

**Intestazione a 40 byte a linea di flusso**, Una serie di campi IPv4 sono stati eliminati o resi opzionali e l'intestazione e' a lunghezza fissa che consente una più rapida elaborazione dei datagrammi IP.

**Etichettatura e proprietà del flusso**, IPv6 ha una definizione elusiva di flusso, "Consente l'etichettatura di pacchetti che appartengono a flussi particolari" come ad esempio audio, video e traffico di altra proprietà. Inoltre IPv6 prevede un campo "classe di traffico" che può essere utilizzato per attribuire proprietà ai datagrammi di un flusso o provenienti da certe applicazioni.

### Il pacchetto IPv6

| +                             | Bits 0-3 | 4-11           | 12-15                         | 16-23       | 24-31     |
|-------------------------------|----------|----------------|-------------------------------|-------------|-----------|
| <b>0-31</b>                   | Version  | Traffic Class  |                               | Flow Label  |           |
| <b>32-63</b>                  |          | Payload Length |                               | Next Header | Hop Limit |
| <b>64</b><br>-<br><b>191</b>  |          |                | Source Address (128 bit)      |             |           |
| <b>192</b><br>-<br><b>319</b> |          |                | Destination Address (128 bit) |             |           |

Il pacchetto IPv6 si compone di due parti principali: l'header e il payload.

L'header è costituito dai primi 40 bytes del pacchetto e contiene 8 campi, 5 in meno rispetto all'IPv4. I campi sono inseriti col byte più significativo messo per primo e all'interno dei singoli byte il bit più significativo è il primo (quello di indice 0).

- **Version [4 bit]** - Indica la versione del datagramma IP: per IPv6, ha valore 6 (da qui il nome IPv6).
- **Traffic Class [8 bit]** - Si traduce come "classe di traffico", permette di gestire le code by priority assegnando ad ogni pacchetto una classe di priorità rispetto ad altri pacchetti provenienti dalla stessa sorgente. Viene usata nel controllo della congestione.
- **Flow Label [20 bit]** - Usata dal mittente per etichettare una sequenza di pacchetti come se fossero nello stesso flusso. Supporta la gestione del **QoS** (Quality of Service), consentendo ad esempio di specificare quali etichette abbiano via libera rispetto ad altre. Al momento, questo campo è ancora in fase sperimentale.

- **Payload Length [16 bit]** - È la dimensione del payload, ovvero il numero di byte di tutto ciò che viene dopo l'header. Da notare che eventuali estensioni dell'header (utili ad esempio per l'instradamento o per la frammentazione) sono considerate payload, e quindi conteggiate nella lunghezza del carico. Se il suo valore è 65.535 byte, significa che ho un pacchetto di dimensione massima, anche detto Jumbogram.
- **Next Header [8 bit]** - Indica quale tipo di intestazione segue l'header di base IPv6. Molto simile al campo protocol dell'header IPv4, del quale usa gli stessi valori.
- **Hop Limit [8 bit]** - È il limite di salti consentito, praticamente il [Time to live](#). Il suo valore viene decrementato di 1 ogni volta che il pacchetto passa da un router: quando arriva a zero viene scartato.
- **Source Address [128 bit]** - Indica l'indirizzo IP del mittente del pacchetto.
- **Destination Address [128 bit]** - Indica l'indirizzo IP del destinatario del pacchetto.

La parte successiva contiene il carico utile (payload in inglese) lungo come minimo 1280 byte o 1500 byte se la rete supporta un MTU variabile. Il carico utile può raggiungere i 65.535 byte in modalità standard o può essere di dimensioni maggiori in modalità "jumbo payload". Esistono due versioni di IPv6 lievemente diverse tra loro: la versione iniziale (ora obsoleta, descritta nel RFC 1883) differisce da quella attuale (descritta nel RFC 2460) per un campo. Si tratta della classe di traffico la cui dimensione è stata portata da 4 a 8 bit. Tutte le altre differenze sono minime.

Sono stati eliminati i campi:

- **Frammentazione/ riassemblaggio**, IPv6 non consente la frammentazione né il riassemblaggio presso i router intermedi. **Possono esser effettuati solo presso l'origine o la destinazione**. Se un router riceve un datagramma troppo grande lo elimina o invia con ICMP "Packet too big". Il mittente poi lo rispedisce della dimensione adattata. Ciò porta ad un miglioramento dell'instradamento IP.
- **Elimina il checksum nell'intestazione**, dato che i protocolli di trasporto hanno già un proprio checksum e questo campo era ridondante. In più il checksum doveva esser ricalcolato su ogni router.
- **Elimina il campo opzioni** rendendo la lunghezza dell'header fissa.

## Passaggio da IPv4 a IPv6

Esistono sostanzialmente two way:

1. **Approccio a doppia pila (dual stack)**, che prevede l'implementazione di un nodo IPv6/IPv4 capace di inviare e ricevere sia datagrammi IPv4 che IPv6. La necessità di determinare la tipologia d'indirizzo IP viene soddisfatta dal DNS che provvederà a restituire un indirizzo della tipologia esatta. Con tale tecnica però siamo "vincolati" alla perdita d'informazioni da IPv4 a IPv6 dato che l'header di IPv4 non contiene tutte le informazioni necessarie (flusso) per determinare l'esatto corrispondente IPv6.
2. **Tunneling**, 6 -> 4 -> 4 -> 6 , Prendiamo il datagramma in IPv6 e lo mettiamo nella parte dati di un datagramma IPv4. I router intermedi instradano il datagramma IPv4 e sul lato di ricezione durante la fase di demultiplexing verrà estratto il datagramma IPv6 contenuto dal datagramma IPv4 giunto a destinazione.

## Algoritmi di instradamento

Generalmente un host è connesso direttamente al router chiamato **router di primo hop** verso cui trasferisce tutti i pacchetti. Il router di primo hop del mittente è detto router origine, la destinazione invece è il router di primo hop della destinazione.

Dato un insieme di router interconnessi, lo scopo di un algoritmo di routing è trovare un "buon percorso" tra origine e destinazione.  
**Ottimale: costo inferiore.**

**Usiamo i grafi non orientati, un grafo è un insieme di N nodi e di E archi: G=(N,E).** Nodo = router che prendono decisioni sull'inoltro, Archi = collegamento fisico, C(i,s) = costo dell'arco che va da i a s , se (i,s) non appartiene a E allora c(i,s) = inf.

L'obiettivo di un algoritmo di routing è l'individuazione dei percorsi meno costosi tra origine e destinazione. Tali algoritmi possono essere:

- **Globali**, l'algoritmo ha una conoscenza globale e completa della rete. Riceve in ingresso tutti i collegamenti e tutti i costi, avendo così informazioni complete sulla connettività e costi. **Algoritmi a stato del collegamento (Link State)**
- **Decentralizzati**, il cammino a costo minimo viene calcolato in modo **distributivo e iterativo**. Nessun nodo ha informazioni complete su tutta la rete. I nodi conoscono solo i costi dei collegamenti adiacenti. Attraverso un processo iterativo e lo scambio di informazioni con i vicini un nodo calcola gradualmente il percorso a costo minimo, **Algoritmo a vettore distanza (Distance Vector)**.
- **Statici**, cammini cambiano raramente, spesso come risultato di un intervento umano.
- **Dinamici**, determinano gli stradamenti al cariare del volume di traffico o della tipologia di rete. Periodicamente o conseguenza di qualche evento. Essi rispondono meglio ai cambiamenti della rete, ma sono anche più soggetti a problemi come l'instradamento circolare e l'oscillazione nei percorsi.
- **Sensibilità al carico**, se un collegamento è affollato si ha un costo alto, Link State [Dijkstra]. Gli algoritmi d'instradamento Internet, sono **insensibili** al carico (RIP, OSPF, BGP).

### **Algoritmo d'instradamento a stato del collegamento "Link State"**

#### **Calcolo del cammino di costo minimo da un nodo S a tutti i nodi della rete**

**Ipotesi:** La tipologia di rete e tutti i costi dei collegamenti sono noti, ossia disponibili in input all'algoritmo LS. Conoscenza globale viene ottenuta grazie all'algoritmo di **LS broadcast**, ovvero tutti i nodi inviano dei pacchetti a tutti gli altri nodi, i quali contengono identità e costi dei collegamenti connessi ai nodi che l'invia.

LS utilizza l'algoritmo di Dijkstra, il quale calcola il cammino di costo minimo da un nodo a tutti gli altri nodi della rete, è iterativo e dopo la k-esima iterazione termina. E si hanno k cammini di costo minimo per k nodi destinazione.

|                                                                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $D[x]$ = valore del costo minimo da $S$ a $X$<br>$p[X]$ = predecessore di $X$ nel cammino da $S$ a $X$<br>$known$ = sottoinsieme di nodi che contiene tutti i nodi il cui cammino di costo minimo è noto. | //Inizializzazione<br>$known = \{S\}$<br>for each $X \in N \setminus \{S\}$ : //inizializzo i costi verso tutti i nodi<br>$D[X] = C[S,X];$<br>if ( $D[X] \neq \infty$ ) $P[X] = S;$<br>//ciclo<br>while ( $known \neq N$ ) do {<br>choose $W \in (N \setminus known)$ : not exist $Z \in N$ : $D[Z] < D[W]$<br>$known = known \cup \{W\}$<br>//controllo che non ci siano cammini di costo minore per i nodi adiacenti a partire dal nodo scelto<br>for each $X \in (N \setminus known)$ : if ( $D[X] > D[W] + C[W,X]$<br>$\{ D[X] = D[W] + C[W,X]$<br>$P[X] = W \}$<br>}         } |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Esempio ....

Cammino a costo minimo da u:

| passo | N'     | D(v),p(v) | D(w),p(w) | D(x),p(x) | D(y),p(y) | D(z),p(z) |
|-------|--------|-----------|-----------|-----------|-----------|-----------|
| 0     | u      | 2,u       | 5,u       | 1,u       | $\infty$  | $\infty$  |
| 1     | ux     | 2,u       | 4,x       | 2,x       | $\infty$  |           |
| 2     | uxy    | 2,u       | 3,y       | 4,y       |           |           |
| 3     | uxyy   | 3,y       |           | 4,y       |           |           |
| 4     | uxyyw  |           |           | 4,y       |           |           |
| 5     | uxyywz |           |           |           |           |           |

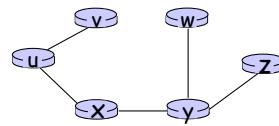


Tabella d'inoltro in u:

| destinazione | collegamento |
|--------------|--------------|
| v            | (u,v)        |
| x            | (u,x)        |
| y            | (u,x)        |
| w            | (u,x)        |
| z            | (u,x)        |

LS ha complessità  $O(n^2)$   
 $n * n(+1)/2$

// **Costo computazionale e' O ( n^2 )**, tale costo puo esser migliorato se utilizziamo un heap diventando  $\log(n)$ .

La tabella d'inoltro puo esser costruita grazie alla conoscenza dei predecessori:

```
nexthop(X) = { R= X;
 while (P[R] != S
 R = P[R]
 return R; }
```

Con tale calcolo provvedo a determinare il router di primo hop rispetto al nodo che sto' analizzando, ovvero non devo farmi ingannare dai predecessori precedentemente determinati.

Una situazione patologica per tale algoritmo si ha quando i costi dei collegamenti sono uguali al carico trasportato sul collegamento, il che riflette il ritardo che si verificherebbe. In questi casi, i costi dei collegamenti non sono simmetrici se il carico trasportato non e' il medesimo. Ciò può portare a delle oscillazioni, risolte assicurandosi che non tutti i router lancino algoritmi LS nello stesso istante. anche se i ricercatori hanno determinato che i router in Internet si possono auto-sincronizzare.

### Algoritmo d'instradamento con vettore distanza "Distance Vector"

Tale algoritmo e' iterativo, asincrono e distribuito. Ogni nodo riceve informazioni direttamente da uno o più dei suoi vicini e dopo aver effettuato i calcoli li restituisce. Non serve che tutti i nodi operino al passo con gli altri.

- **Distribuito:** nel senso che ciascun nodo riceve parte dell'informazione da uno o piu' dei suoi vicini direttamente connessi a lui.
- **Iterativo:** nel senso che questo processo si ripete fino a quando non avviene un ulteriore scambio con i vicini o il cambiamento di un costo, l'algoritmo e' auto-terminate.
- **Asincrono:** non richiede che tutti i nodi operino al passo con gli altri.

Sia  $Dx(y)$  il costo minimo da X a Y, allora per la regola di Bellman-Ford si ha che

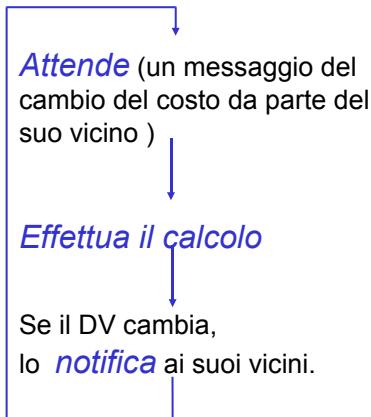
$$Dx(y) = \min \{ C(X,V) + Dv(y) \}$$

dove  $\min$  riguarda tutti i vicini di X. Questa regola ci fornisce le righe della tabella d'inoltro nel nodo X. Ogni nodo mantiene alcune informazioni:

- Per ciascun vicino V, il costo  $C(X,V)$  da X a V;
- Il vettore distanza del nodo X, che e'  $Dx[ Dx(y) : Y \in N ]$  che contiene la stima presso X del costo verso tutte le destinazioni Y in N;
- Il vettore distanza dei suoi vicini, ossia  $Dv[ Dv(y) : Y \in N ]$

In questo algoritmo di quando in quando un nodo invia una copia del proprio vettore distanza a ciascun vicino, quando questo lo riceve con la formula di Bellman-Ford aggiorna il suo vettore distanza  $Dx(y) = \min\{C(X,Y) + Dv(y)\}$  per ogni  $Y \in N$

### Ciascun nodo:



**Iterativo, asincorno**, quando ogni iterazione locale e' causata da un cambio del costo di uno dei collegamenti locali o dalla ricezione da parte di qualche vicino di un vettore delle distanze aggiornato.

**Distribuito**, ogni nodo aggiorna i suoi vicini solo quando il suo DV cambia. ( tra vicini si avvisano se necessario)

Nel caso di notizie buone in poche iterazioni l'algoritmo e' in grado di aggiornarsi velocemente, al contrario "**bad news goes slowly**" esempio se il collegamento B-D schianta

L'algoritmo DV e' **decentralizzato** e non usa informazioni globali sulla rete, ma solo quelle ottenute dal nodo, ovvero il costo dei collegamenti verso i vicini direttamente connessi e quelle ricevute da questi vicini.

### Algoritmo

for each  $y \in N \setminus \{S\}$ :  $Ds[y] = C(S,y); //$  se non adiacente metto inf.

$Ds[S] = 0;$

for each "vicino"  $V$   $Dv[y] = \text{inf}$  per ogni  $y \in N \setminus \{S\}$

for each "vicino"  $V$  invia  $Ds = [Ds(y) : y \in N]$

loop

on event "cambia costo di un collegamento con  $S$  o riceviamo un vettore da un vicino"

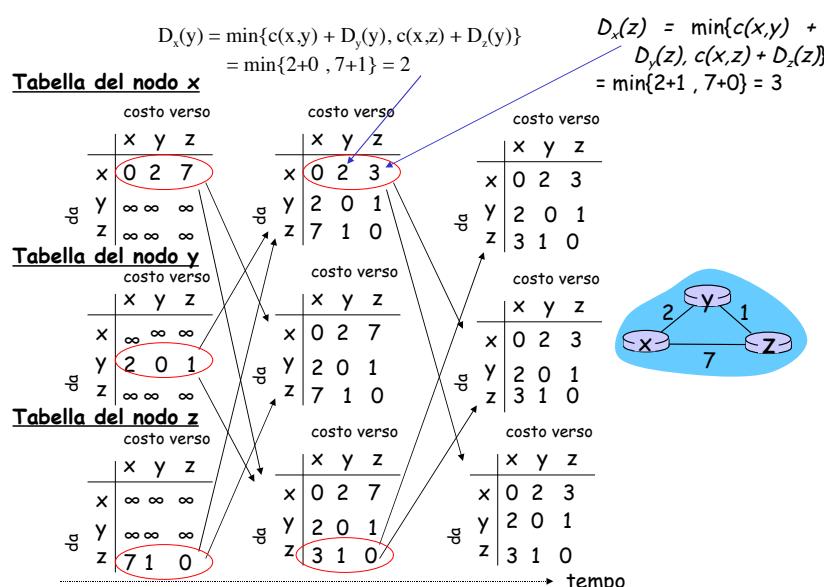
do {

for each  $y \in N \setminus \{S\}$ :

$Ds[y] = \min\{C(S,y) + Dv(y)\};$

if  $Ds[y]$  e' cambiato invia  $Ds$  a tutti i vicini

Esempio..



## Modifica dei costi e guasti dei collegamenti

Le buone notizie si propagano velocemente, raggiungendo in poco tempo lo stato di quiescenza. Mentre le cattive notizie oltre a propagarsi lentamente, possono dar origine al cosiddetto **instradamento ciclico**, che ha come risultato il problema del count-to-infinity.

Per risolvere tale problema dobbiamo applicare la tecnica dell'inversione avvelenata.

### Aggiunga dell'inversione avvelenata (*poisoned reverse*)

Se ad esempio C instrada verso B i pacchetti da far giungere a D, allora C avverrà B che la sua distanza verso D è infinita. In questo modo B non instraderà a C i pacchetti per D.

### Confronto tra gli algoritmi LS e 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.

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.

Velocità di convergenza:

- **LS**: l'algoritmo  $O(n^2)$  richiede  $O(nE)$  messaggi.
  - ci possono essere oscillazioni di velocità.
- **DV**: può convergere lentamente.
  - può presentare cicli d'instradamento.
  - può presentare il problema del conteggio all'infinito.

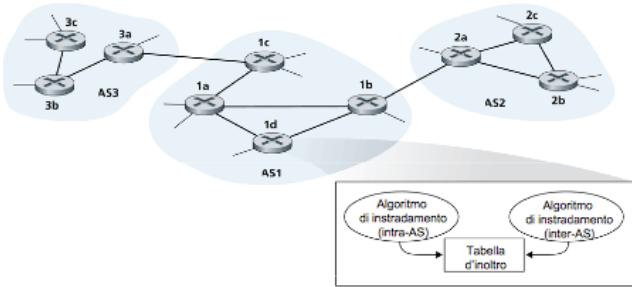
**Altre informazioni su questi tipi di problemi, come si creano in realtà'.**

### Instradamento Gerarchico

Fino ad adesso abbiamo considerato tutti i router identici che eseguivano lo stesso algoritmo, ma questa visione è semplicistica per almeno 2 motivi:

- **Scalabilità**, al crescere del numero di router memorizzare e comunicare le informazioni d'instradamento diventa proibitivo, ad esempio LS non lascerebbe più spazio per i messaggi, DV invece non convergerebbe mai.
- **Autonomia amministrativa**, ciascuno dovrebbe esser in grado di amministrare la rete nel modo che desidera, pur mantenendo la possibilità di collegarsi a reti esterne.

Questo problema può esser risolto organizzando i router in **sistemi autonomi AS**, i router di un AS eseguono lo stesso algoritmo d'instradamento ovvero il **protocollo d'instradamento interno al AS** [ intra-autonomous system routing protocol ] e per collegarsi due AS utilizzano un router speciale chiamato **Gateway**.



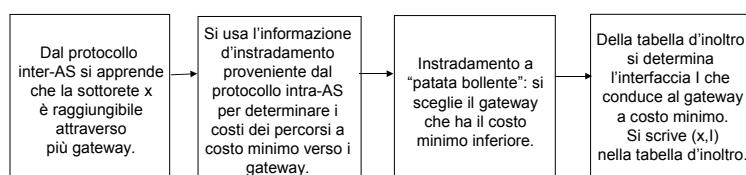
I router interni se hanno un solo Gateway sanno quale è il cammino minimo verso di esso e se devono inviare un pacchetto ad un altro AS lo faranno tramite lui.

Il **Gateway** dell'AS destinatario si occuperà poi di instradare il pacchetto nell'AS di sua competenza.

Se invece ce ne sono 2?? In questo caso AS1 dovrebbe conoscere le destinazioni raggiungibili tramite AS2 e AS3 e quindi informare i router all'interno del estremo in modo che sappiano dove mandare, ovvero configurare la propria tab d'inoltro. Per risolvere queste problematiche si fa ricorso al un unico **protocollo d'instradamento tra AS** [inter-autonomous system routing protocol]. Tutti gli AS devono usare lo stesso protocollo [BGP 4 internet].

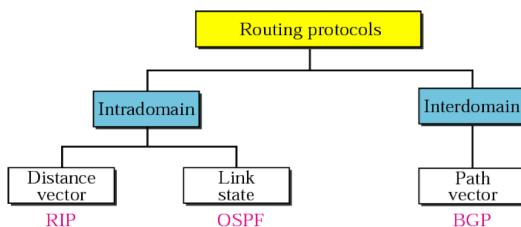
Se un router generico interno all'AS riceve 2 pubblicità per una solita destinazione, lui sceglie quello con il gateway più vicino tendenzialmente tramite l'**instradamento a patata bollente** [hot potato routing].

Quando un AS viene a conoscenza di una destinazione di un AS confinante, può inoltrare questa informazione ad altri AS confinanti secondo certe regole (politiche), i suoi vicini che sono stati informati poi se devono rispedire alla nuova destinazione lo faranno tramite lui.



## Instradamento in Internet

Ora ci concentreremo sui protocolli d'instradamento Internet, il cui compito è quello di determinare il percorso che un datagramma seguirà dall'origine alla destinazione

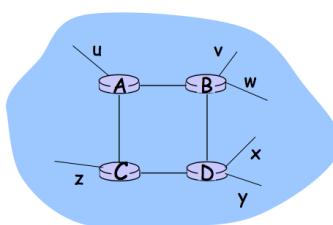


## Instradamento interno ai sistemi Internet (intra AS)

I protocolli d'instradamento intra-AS sono noti come **protocolli gateway interni (IGP)**, i più usati sono RIP e OSPF.

### RIP [ udp 520 ]

è un protocollo **DV + poisoned reverse**, con le seguenti caratteristiche:



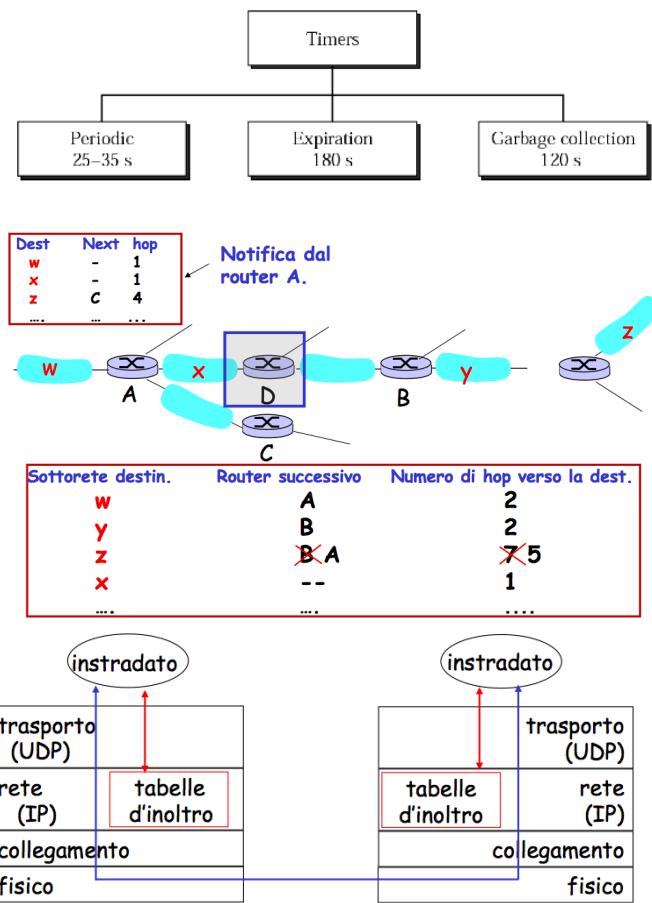
Dal router A alle varie sottoreti:

| destinazione | hop |
|--------------|-----|
| u            | 1   |
| v            | 2   |
| w            | 2   |
| x            | 3   |
| y            | 3   |
| z            | 2   |

- Conteggio degli host come metrica dei costi: #hops, ovvero ogni collegamento ha costo 1
- Costi calcolati tra router origine e sottorete, destinazione inclusa. Il termine *hop* rappresenta il numero di sottoreti attraversate lungo il percorso minimo dal router origine alla sottorete destinazione.
- Costo massimo 15 (15 hops).
- Si scambiano i vettori ogni 30 secondi con un messaggio di risposta

RIP, il quale contiene un elenco con fino a 25 sottoreti destinazione e la distanza del mittente verso tali sottoreti **Annunci RIP**

**Advertisement**



- se un router non riceve adv da V per 180 secondi allora considero rotto il collegamento, viene modificata la tabella d'instradamento e l'informazione manda a tutti i vicini.

- **Periodic timer**: controlla gli “advertising” di regolari messaggi di update (25-35 sec)

- **Expiration timer:** governa la validità di una route (180 sec), Se riceve un update (circa ogni 30 sec) il timer si resetta ; Se non si riceve un update in questo periodo la metrica è posta a 16.

- **Garbage timer:** 120 sec , Una route può essere annunciata con metrica 16 per 120 sec prima di venire eliminata. Permette ai vicini di avere conoscenza che una route è invalida

Ciascun router mantiene una **Tabella d'instradamento** RIP, che include il vettore di distanze e la tabella d'inoltro.

Un router può richiedere informazioni sul costo dei vicini usando i messaggi di richiesta *RIP*, tali messaggi usano UDP sulla porta 520.

Un processo chiamato **routed** esegue RIP, ossia mantiene le informazioni d'instradamento e scambia messaggi con i processi **routed** nei router vicini.

Poiché RIP viene implementato come un processo a livello di applicazione (routed), può inviare e ricevere messaggi su una socket standard e utilizzare un protocollo di trasporto standard (**UDP, porta 520**).

In DV quando ricevo un cammino col costo maggiore posso trovare un cammino alternativo meno costoso, questo con RIP invece non e' possibile **perché non mantengo le vecchie informazioni sui vicini**, quindi mi tengo il costo maggiore, ma dopo 30 secondi ricevendo un adv e me ne accorgo.

OSPF [id 89]

Il protocollo **Open Shortest Path First** o **OSPF** è uno dei protocolli di instradamento più diffusi, che gestisce le tabelle di instradamento di una rete Internet con il metodo dello LS, generalmente utilizzati negli ISP di livello superiore.

Questo standard è open (aperto) nel senso che è un protocollo non proprietario.

Il protocollo utilizza un metodo di instradamento che non si differenzia sostanzialmente da quello LS, ma aggiunge delle altre proprietà:

- È un protocollo a stato del collegamento che:
    - Utilizza il **flooding** di informazioni di stato del collegamento, ovvero adv disseminati (inondazioni)
    - Un router costituisce una propria tipologia dell'AS
    - Utilizza l'algoritmo di **Dijkstra** per la determinazione del percorso a costo minimo.
    - I **costi dei collegamenti** sono fissati dall'amministratore, di default sono posti tutti ad 1 (metrica degli hop), ma permette anche di modificarli andando ad applicare politiche d'ingegnerizzazione del traffico.
  - Con OSPF, ogni volta che si verifica un cambiamento nello stato di un collegamento, il router manda informazioni d'instradamento via broadcast a tutti gli altri router nel sistema autonomo. Comunque ogni 30 minuti vengono mandate informazioni sullo stato per aggiungere robustezza.

- **Invia messaggi OSPF all'intero sistema autonomo, utilizzando il flooding.**

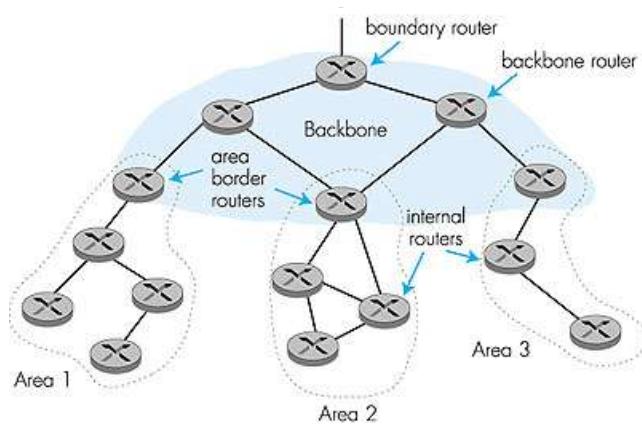
- I messaggi OSPF vengono trasportati direttamente da IP con id 89 (e non da TCP o UDP) con un protocollo di livello superiore implementato da OSPF che permetta il trasferimento affidabile dei messaggi e il broadcast dello stato dei collegamenti, andando ad effettuare anche un controllo su di essi verificando che siano attivi.

- Autenticazione dei messaggi
- Cammini multipli
- Fornisce supporto al multicast
- Bilanciamento del carico
- Aggiunta di un ulteriore grado di gerarchia nei domini

altre proprietà sono:

I vantaggi di OSPF rispetto a RIP sono :

- Sicurezza: gli scambi tra router sono autenticati (MD5), evitando l'immissione di informazioni errate nella tabella di routing.
  - Semplice: si configura su tutti i router la stessa pw che deve esser inclusa nei pacchetti OSPF ad essi inviati.
  - MD5: funzione hash crittografico a chiavi segrete e condivise, configurate in ogni router.
- Multipath: quando più percorsi verso una destinazione hanno lo stesso costo, OSPF consente di usarli senza doverne scegliere uno, come invece avveniva in RIP
- Su ciascun collegamento, vi possono essere più metriche di costo per differenti **TOS** (es. il costo del satellite sarà "basso" per un best effort; elevato per un real time)
- Supporto integrato per l'instradamento unicast e multicast.
  - Per consentire l'instradamento multicast viene impiegato MOSPF (OSPF multicast) che utilizza il database di collegamenti OSPF.
- Supporto alle gerarchie in un dominio d'instradamento.



Una rete OSPF è divisa in aree. Esse sono gruppi logici di router le cui informazioni posso essere sommarizzate rispetto al resto della rete. Diversi tipi di aree "speciali" sono definite:

- **Area Backbone** : L'area backbone (conosciuta anche come area zero) rappresenta il cuore di una rete OSPF. Tutte le altre aree sono collegate ad essa e il routing inter-area passa tramite un router di questa rete.
- **Stub area** : Per Stub Area si intendono quei tipi di area che non ricevono route esterne. Le route esterne saranno poi definite e distribuite da un altro protocollo di Routing. Quindi, le stub area necessitano di relegare ad una route di default lo scambio per il traffico con quelle esterne al dominio di appartenenza.
- **Totally stubby area** : Una totally stubby area è simile ad una stub area, tuttavia quest'area non permette route riassuntive oltre che le route esterne.

L'unico modo in cui il traffico esce dall'area è una route di default che è l'unica di Tipo-3 LSA pubblicata nell'area. Quando c'è solo una route per uscire dall'area, devono essere effettuate meno decisioni di routing dal processore di route, con minore utilizzo di risorse di sistema. Questa è la versione Cisco della NSSA.

- **Not-so-stubby area** : Identificata anche come NSSA, una not-so-stubby area è un tipo di stub area che può importare route esterne di AS e mandarle al backbone, ma non può ricevere tali route esterne di AS dal backbone o da altre aree.

Un'AS OSPF puo' esser configurato in aree che eseguono diversi algoritmi d'instradamento OSPF, vi e' una gerarchia tra le aree: area locale e dorsale, caratterizzate da:

- Messaggio di link-state **solo** all'interno dell'area

- **Ciascun nodo ha una sua area;** conosce solo la direzione (shortest path) verso le reti nelle altre aree

Il ruolo di un area dorsale e' quello di instradare il traffico tra le altre aree ed essa contiene tutti i router di confine.

- Router interni, sono in aree non dorsali ed eseguono solo instradamento nell'area.
- Router di confine d'area: appartengono sia a un'area generica sia alla dorsale.
- Router di dorsale: effettuano l'instradamento all'interno della dorsale, ma non sono router di confine.
- Router di confine: scambiano informazioni con i router di altri sistemi autonomi.

## Instradamento tra Sistemi Autonomi di Internet (intra AS)

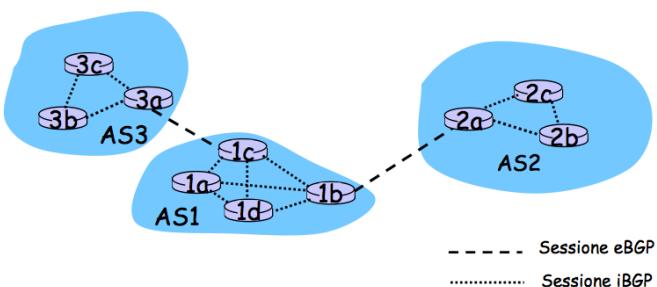
### Border gateway protocol [BGP]

Standard per collegamenti tra AS e mette a disposizione un modo per:

- ottenere *informazioni* sulla raggiungibilità delle sottoreti da parte di sistemi confinanti
- propagare *informazioni* di raggiungibilità a tutti i router interni di un AS
- determinare percorsi buoni verso le sottoreti sulla base delle informazioni di raggiungibilità e delle politiche dell'AS.
- consente a ciascuna sottorete di comunicare la propria esistenza al resto di Internet

**BGP usa connessioni TCP semi-permanenti sulla porta 179, in genere si ha una connessione TCP tra due router di 2 diversi AS collegati fisicamente e una connessione TCP per ogni coppia di router interni di un AS.**

### Fondamenti di BGP:



- I router ai capi di una connessione TCP sono chiamati **peer BGP**, e la connessione TCP con tutti i messaggi BGP che vi vengono inviati è detta **sessione BGP**, che si divide tra **sessioni BGP esterne - eBGP** e **sessioni BGP interne - iBGP**.
- In BGP le destinazioni non sono host, ma **prefissi CIDR** che rappresentano una sottorete o una collezione di sottoreti.
- Notiamo le linee di sessione BGP non sempre corrispondono ai collegamenti fisici.
- Quando AS2 annuncia un prefisso a AS1, AS2 sta in realtà **promettendo** che inoltrerà i datagrammi su un percorso verso il prefisso cui sono destinati.

- AS2 può aggregare più prefissi nel suo annuncio

In una sessione eBGP tra i gateway 3a e 1c, AS3 invia ad AS1 la lista di prefissi raggiungibili. 1c utilizza le proprie sessioni iBGP per distribuire i prefissi agli altri router del sistema autonomo. Anche AS1 e AS2 si scambiano informazioni sulla raggiungibilità dei prefissi attraverso i propri gateway 1b e 2a. Quando un router viene a conoscenza di un nuovo prefisso, lo memorizza in una nuova riga della propria tabella d'inoltro.

Due gateway comunicano tra loro la lista dei prefissi raggiungibili sulle connessioni eBGP, quando ricevono le info le inviano a iBGP in modo che tutti i router interni di AS conoscano i prefissi. In BGP ogni AS viene identificato dal suo numero di **AS (ASN) globale e univoco**.

### Attributi del percorso e rotte BGP

Quando un router annuncia un prefisso per una sessione BGP, include anche un certo numero di **attributi BGP**.

### **prefisso + attributi = (in gergo e' detto) "rotta"**

Due dei più importanti attributi sono:

- **AS-PATH:** elenca i sistemi autonomi attraverso i quali è passato l'annuncio del prefisso (tramite ASN) e viene usato per eliminare gli annunci **reiterati** e per avere una scelta tra i percorsi verso lo stesso prefisso: AS 67 AS 17
- **NEXT-HOP:** quando si deve inoltrare un pacchetto tra due sistemi autonomi, questo potrebbe essere inviato su uno dei vari collegamenti fisici che li connettono direttamente.

Quando un router gateway riceve un annuncio di rotta, utilizza le proprie *politiche d'importazione* per decidere se accettare o filtrare la rotta.

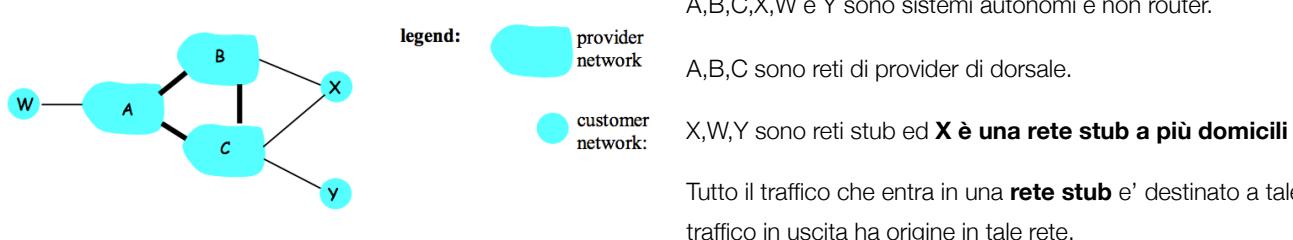
### **Selezione dei percorsi BGP**

Un router può ricavare più di una rotta verso un determinato prefisso, e deve quindi sceglierne una.

Regole di eliminazione:

- Alle rotte viene assegnato come attributo un valore di preferenza locale. Si selezionano quindi le rotte con i più alti valori di preferenza locale.
- Tra le rotte con preferenza uguale si seleziona la rotta con valore AS-PATH più breve.
- Si seleziona quella il cui router di NEXT-HOP è più vicino: instradamento a patata bollente.
- Se rimane ancora più di una rotta ovvero la , il router si basa sugli identificatori BGP.

### **Politiche d'instradamento**



X opera da rete stub nel momento in cui annuncia ai suoi vicini B e C di non avere percorsi verso altre destinazioni tranne se stessa. Ovvero anche se X conoscesse un percorso del tipo XCY non l'annuncerebbe a B.

- X non vuole che il traffico da B a C passi attraverso di lui

- ... e così X non annuncerà a B la rotta verso C

A annuncia a B del percorso AW.

B annuncia a X del percorso BAW.

B deve annunciare a C del percorso BAW?

- Certo che no! B non ha nessun "interesse" nella rotta CBAW poiché né W né C sono clienti di B
- B vuole costringere C ad instradare verso W attraverso A

B vuole instradare **solo** da/verso i suoi clienti!

### **Perché i protocolli d'instradamento inter-AS sono diversi da quelli intra-AS?**

#### **Politiche:**

- Inter-AS: il controllo amministrativo desidera avere il controllo su come il traffico viene instradato e su chi instrada attraverso le sue reti.

- Intra-AS: unico controllo amministrativo, e di conseguenza le questioni di politica hanno un ruolo molto meno importante nello scegliere le rotte interne al sistema

#### **Scala:**

- L'instradamento gerarchico fa “risparmiare” sulle tabelle d'instradamento, e riduce il traffico dovuto al loro aggiornamento

#### **Prestazioni:**

- Intra-AS: orientato alle prestazioni
- Inter-AS: le politiche possono prevalere sulle prestazioni

## Instradamento Broadcast e Multicast

*Fino ad ora c'eravamo concentrati sui protocolli d'instradamento che supportano la comunicazione uniscat (punto-punto), invece ora ci concentreremo sull'istradamento broadcast ,con il quale il livello di rete offre un servizio di consegna di pacchetti spediti da 1 sender verso + receiver all'interno della rete con una singola operazione di spedizione, e sull'istradamento multicast che permette ad un sender di spedire verso alcuni nodi della rete.*

### **Algoritmi d'instradamento broadcast**

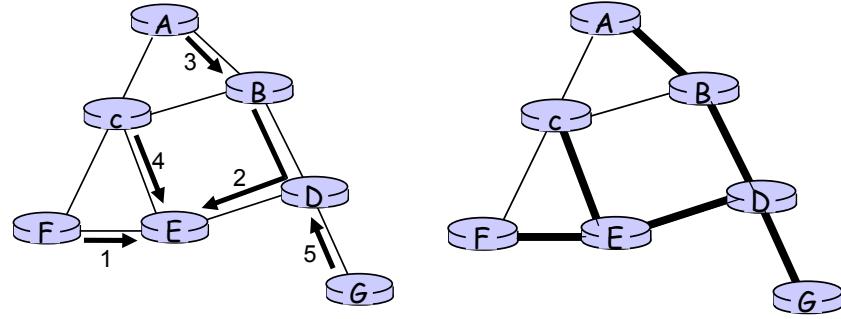
1. **Unicast a N vie:** crea N copie del pacchetto e lo invia alle N destinazioni usando l'instradamento unicast. E' inefficiente perché un router può esser attraversato da N copie dello stesso pacchetto e gli indirizzi dei destinatari devono esser noti al mittente, il che richiederebbe ulteriori meccanismi.
2. **Flooding incontrollato:** Ogni nodo che riceve un pacchetto ne costruisce una copia da inviare a tutti i suoi vicini eccetto chi glielo ha spedito. Se ci sono cicli, riceverò lo stesso pacchetto un numero di volte indefinito e si può dare origine ad una tempesta di broadcast che renderà inutilizzabile la rete.
3. **Flooding controllato:** il controllo può esser effettuato da:
  - 3.1. Controllo con i numeri di sequenza, un nodo origine mette il suo indirizzo e un numero di sequenza broadcast nel pacchetto, ogni nodo ha una lista di indirizzi d'origine e di numeri di sequenza per ogni pacchetto broadcast ricevuto, duplicato ed inoltrato. Se e' nella lista viene eliminato altrimenti inoltrato e aggiornata la lista.
  - 3.2. RPF, inoltro su percorso inverso, quando un router riceve un pacchetto broadcast lo invia su tutti i collegamenti in uscita solo se e' pervenuto attraverso il percorso unicast più breve tra router e origine.

### **Broadcast con spanning tree**

I nodi della rete devono “costruire un albero ricoprente”. Quando un nodo deve spedire un pacchetto broadcast, lo spedisce su tutti i collegamenti che appartengono all'albero.

Gli alberi ricoprenti oltre ad eliminare i pacchetti broadcast ridondanti, possono esser usati dai nodi per dare inizio a un broadcast.

Un nodo non deve conoscere tutto l'albero ma solo i vicini. Per quanto riguarda la determinazione dell'albero :



Usando un approccio con cento si ha che viene scelto un centro per costruire l'albero di coprente.

Viene utilizzato l'instradamento unicast che fa sì che i nodi mandino un messaggio di adesione al nodo centrale, che prosegue lungo le connessioni finché non giunge al nodo centrale o ad un nodo che fa parte dell'albero.

*Il percorso di questo messaggio sarà l'arco dell'albero.*

## Algoritmi di broadcast

## Domande e Problemi

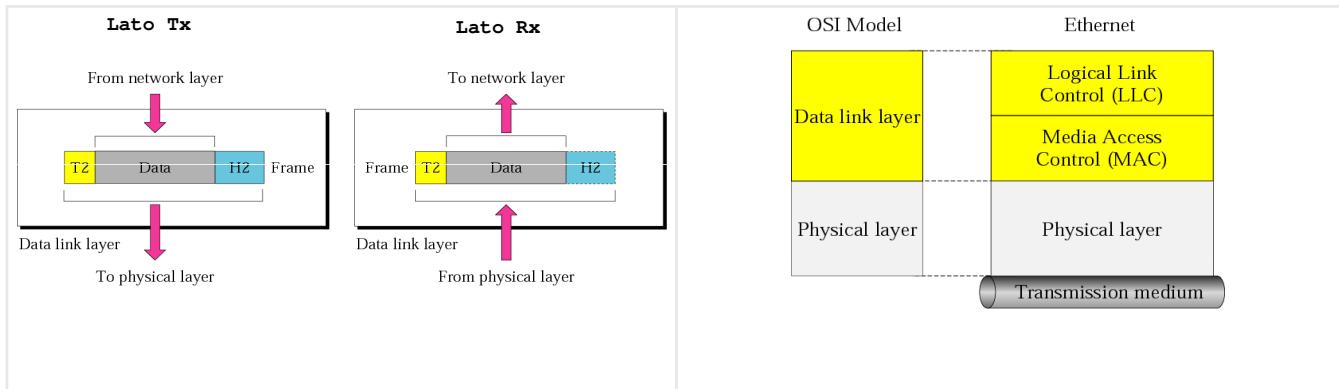
### Protocolli

1. Che cosa vuol dire che un protocollo è "stateful" ?
  - 1.1. Un protocollo stateful è un protocollo in cui parte dei dati scambiati tra client e server includono informazioni di stato. Entrambi i sistemi hanno memoria dello stato della sessione di comunicazione.
2. Che cosa vuol dire che un protocollo è "stateless" ?
  - 2.1. Un protocollo stateless è un protocollo in cui né il client né il server ha l'obbligo di tenere traccia dello stato della sessione di comunicazione. I protocolli connectionless sono tipicamente anche stateless
3. Fornite un esempio di un protocollo stateful e di un protocollo stateless
  - 3.1. UDP è un protocollo connectionless, stateless e unreliable.
  - 3.2. TCP è un protocollo connection-oriented, stateful, e reliable
4. FTP usa un controllo out-of-band.
  - 4.1. Vero
5. Un servizio connectionless non è in grado di garantire il controllo del flusso.
  - 5.1. Vero
6. In un servizio affidabile non c'è riscontro dell'avvenuta ricezione dei pacchetti.
  - 6.1. Falso



7. Il livello 3 di TCP/IP offre un servizio di tipo connectionless.
- 7.1. Vero
8. Accendete il PC e, con un http client, volete recuperare un documento WWW ad un certo URL. L' IP del server http non è noto. Quali protocolli, oltre ad http, sono necessari in questo scenario ?
  - 8.1. DHCP (per farsi assegnare un IP); ARP (per apprendere il MAC address del ROUTER); DNS (per apprendere l'IP del server http) e TCP (per l'http) – UDP (per il DNS).
9. Se è specificato nell'header IP: DF=1 cosa succede ai vari routers incontrati lungo il cammino verso la destinazione ?
  - 9.1. Se c'è una esigenza di frammentazione il Router butta il pacchetto e invia un messaggio ICMP al Source
10. Il campo opzioni nell'header IPv4 ha la lunghezza massima di 40 Byte.
  - 10.1. L'header IPV4 ha una lunghezza massima di 60 bytes, di cui 20 sono sempre riempiti dai campi obbligatori e fino a 40 bytes di options
11. Nel protocollo DHCP il messaggio iniziale di broadcast di richiesta utilizza TCP.
  - 11.1. DHCP è un protocollo indipendente e TCP non può essere comunque usato, dato che TCP è punto-punto
12. Piano di Indirizzamento: dato il pool di indirizzi IP 192.168.1.0 e una subnet mask di 255.255.255.0 dividere lo spazio di indirizzamento in:
  - 12.1. 16 sottoreti disgiunte (3)
13. e per ognuna delle reti indicare
  - 13.1. l'indirizzo di rete (3) e di broadcast (3)
14. La prima delle 16 reti (indirizzo IP più basso) vogliamo usarla solo per collegamenti Punto- Punto:
  - 14.1. quanti link PP possiamo individuare (MAX) ? (3) quali sono ? (3)

# Capitolo 5 Livello DataLink



**LCC** e' comune a tutte le LAN ed e' l'interfaccia verso il livello network, invece **MAC** e' specifico per ogni LAN e risolve il problema della condivisione del mezzo di trasmissivo.

## Livello di link: introduzione e servizi

Il protocollo a livello link si occupa del trasporto di **frame** lungo un *singolo* canale di comunicazione. Alcuni termini utili:

- host e router sono i **nodi** (nel senso che per il DL layer sono indistinguibili)
- i canali di comunicazione che collegano nodi adiacenti lungo un cammino sono i **collegamenti (link)**
  - collegamenti cablati
  - collegamenti wireless
  - LAN
- Le unità di dati scambiate dai protocolli a livello di link sono chiamate **frame**.

Un frame può esser gestito da diversi protocolli, su differenti collegamenti, Ethernet, PPP e WAN ad esempio su tre collegamenti differenti. E i servizi erogati da ogni protocollo possono differire.

I protocolli a livello Data link sono : Ethernet – 802.11 Wireless LAN (Wi-Fi) – Token Ring – PPP – ATM\* – Frame Relay – X.25 – HDLC – FDDI ecc..

## Servizi offerti dal livello DataLink

Possiamo così elencarli:

- *Framing*: Unita' di dati scambiati, frame che incapsulano un datagramma e contengono altri campi intestazione tra cui e' presente anche eventuali campi trailer.
- *Accesso al Link (link access)*
  - Per identificare origine e destinatario vengono utilizzati indirizzi "MAC" [00:01:02:03:04:05], il protocollo controlla l'accesso al mezzo (MAC, medium access control) il quale specifica le regole con cui immettere i frame nel collegamento.
  - Nei collegamenti punto-punto, il mittente può inviare il frame quando determina che il canale e' libero ( MAC semplice o assente)
  - Caso più interessante, canale broadcast.
- *Consegna affidabile ( reliable delivery )* : il protocollo link che fornisce un servizio di consegna affidabile, garantisce il trasporto senza errori di ciascun datagramma.

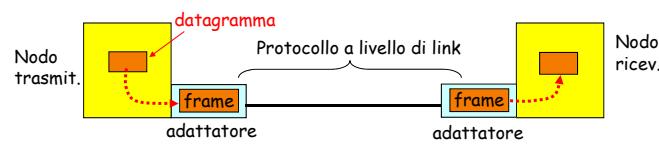
- Generalmente con ack e ritrasmissioni (es, wireless) con lo scopo di risolvere gli errori localmente anzi che tra mittente e destinatario.  
Ovvero non end-to-end ma hop-to-hop.
  - È considerata non necessaria nei collegamenti che presentano un basso numero di errori sui bit (fibra ottica, cavo coassiale e doppino intrecciato)
  - È spesso utilizzata nei collegamenti soggetti a elevati tassi di errori (es.: collegamenti wireless)
- *Controllo di flusso (flow control):* I due nodi agli estremi del collegamento hanno un *buffer limitato*, quindi evita che si saturi.
  - *Rilevazione degli errori (error detection)*
    - Gli errori sono causati dall'attenuazione del segnale e da rumore elettromagnetico.
    - Il nodo ricevente individua la presenza di errori
      - è possibile grazie all'inserimento, da parte del nodo trasmittente, di un bit di controllo di errore all'interno del frame.
    - Come abbiamo visto il livello di trasporto Internet e il livello di rete forniscono una limitata modalità di rilevazione dell'errore, che è solitamente più sofisticata a livello di collegamento ove è implementata nell'hardware.
  - *Correzione degli errori (error correction):* Il nodo ricevente sa anche dove s'è verificato l'errore e lo corregge. Alcuni protocolli forniscono questo servizio solo per gli header (ATM).
  - *Half-duplex e full-duplex*
    - Nella trasmissione full-duplex gli estremi di un collegamento possono trasmettere contemporaneamente: non in quella half-duplex.

Da notare che il protocollo di trasporto implementa il servizio tra due processi su base **end-to-end**, mentre il protocollo affidabile del livello di collegamento fornisce il servizio di consegna affidabile tra i nodi di un singolo collegamento.

## Chi implementa il DataLink Layer

Per un dato collegamento il protocollo a livello link è sostanzialmente realizzato da un **adattatore**, noto anche come *scheda di interfaccia di rete (NIC)*, e molti dei servizi offerti sono realizzati a livello hw nell'adattatore (Adattatori Ethernet, adattatori PCMCIA e adattatori 802.11).

Il cuore dell'*adattatore* è il controllore a livello di collegamento, che è di solito un solo chip specifico, che implementa molti dei servizi a livello di collegamento (framing, accesso al collegamento, controllo di flusso, rilevazione degli errori ecc).



### Lato trasmittente:

- Il controllore prende un datagramma che è stato creato e memorizzato nella memoria dell'host dai livelli più alti, lo incapsula in un frame.

- Riempie i vari campi d'intestazione, imposta il bit rilevazione degli errori, trasferimento dati affidabile, controllo di flusso, etc.
- Lo trasmette nel canale di comunicazione, seguendo il protocollo di accesso al canale.

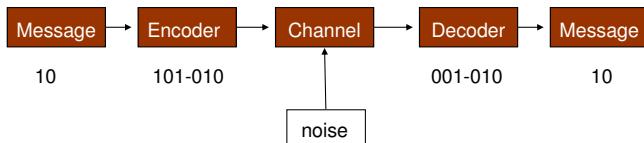
### Lato ricevente:

- L'adattatore riceve l'intero frame
- Individua gli errori, trasferimento dati affidabile, controllo di flusso, etc.
- Estrae i datagrammi e li passa al nodo ricevente

Se previsto il controllo del flusso, i due controlli mittente e ricevente si scambieranno informazioni.

**L'Adattatore** è un'unità semi-autonoma che lavora a livello di link fisico. La quale trasferisce i frame da uno all'altro. Essa è realizzata in hardware e in software, che viene eseguito dalla CPU dell'host.

## Tecniche di rilevazione e correzione degli errori



Al nodo trasmittente, ai dati D che devono esser protetti da errori vengono aggiungi dei bit detti **EDC ( error detection and correction )**.

Il nodo ricevente deve determinare se D' coincide con D, potendo contare solo su D' e EDC'.

Anche con l'utilizzo dei bit di rilevazione degli errori e' possibile che ci siano degli **errori non rilevati**, vale a dire che il nodo ricevente potrebbe non accorgersi che le informazioni contengono errori.

Dobbiamo quindi cercare di ridurre al possibilità di questo evento, le tecniche più sofisticate prevedono un'elevata ridondanza.

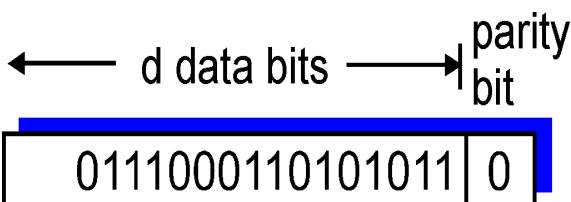
Le tecniche per la rivelazione degli errori sono:

- **Controllo di parità ( parity check )**
- **Controllo basato sulla Checksum**
- **Controllo a Ridondanza Ciclica (Cyclic redundancy check)**

### Controllo di Parità

## Unico bit di parità:

Si è verificato almeno un errore in un bit



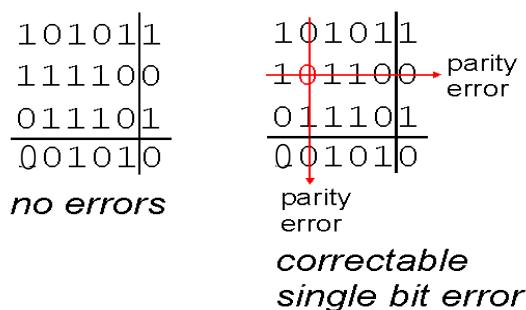
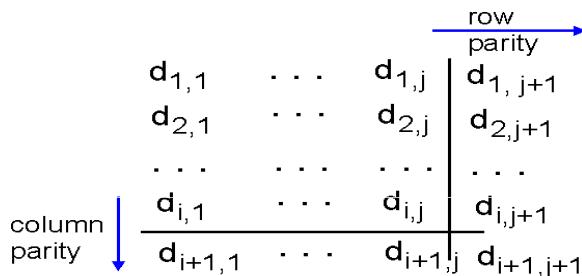
La forma più semplice di rilevamento degli errori e' quella che utilizza un unico **bit di parità**. In questo caso il mittente non fa' altro che aggiungere ai dati d trasmessi un bit che determina la parità/disparità dei bit posti a 1 in d. Invece il destinatario si limita a contare il numero di bit e trarne le conclusioni banali.

Generalmente il numero di errori e' maggiore di uno, ed in molti casi essi non sono indipendenti l'uno dall'altro. Nel caso di un numero di errori pari, siamo in presenza di un **errore non rilevato**.

Una strategia che si basa sempre sui bit di parità e che ci permette un

## Parità bidimensionale:

**Individua e corregge il bit alterato**



primo approccio alla tecnica di correzione dell'errore, prevede una **generalizzazione bidimensionale dello schema di parità con un bit**. In questo caso, i  $d$  bit del dato  $D$  sono suddivisi in  $i$  righe e  $j$  colonne per ognuna delle quali è stato calcolato un valore di parità.

Nel caso si verifichi un errore in un bit esso può essere determinato e corretto andando ad analizzare il tutto righe per colonne. E lo stesso può essere immediatamente corretto.

La capacità di rilevare e correggere gli errori è conosciuta come **correzione degli errori in avanti (FEC, forward error correction)**.

Tali tecniche sono molto utili perché possono diminuire il numero di ritrasmissioni, ma forse è più importante il fatto che permettono al ricevente l'immediata correzione degli errori.

Evitando l'attesa del round-trip.

## Somma di controllo

Ha come obiettivo quello di rilevare gli errori ma viene usata solo a livello di trasporto, infatti il Checksum di Internet prevede:

### il mittente

- I dati sono trattati come interi da 16 bit e sommati.
- Checksum è il complemento a 1 di questa somma
- Il mittente inserisce il valore della checksum nell'intestazione dei segmenti

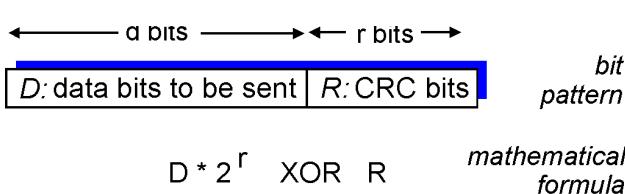
### il destinatario

- Il ricevente controlla la checksum.
- Calcola il complemento a 1 della somma dei dati ricevuti e verifica che i bit del risultato siano 1:
  - NO**, non lo sono: segnala un errore;
  - Sì** lo sono: non sono stati rilevati errori.

Da notare che nei protocolli TCP e UDP, la checksum è calcolata per tutti i campi intestazione e dati, mentre IP la calcola solo sull'intestazione perché i dati hanno la propria.

Quindi si vengono a creare due calcoli distinti del checksum, uno per il livello applicazione, realizzato in software; e uno per il livello di collegamento realizzato ad hardware e quindi più efficiente.

## Controllo a Ridondanza Ciclica



- Esamina i dati,  $D$ , come numeri binari.
- Origine e destinazione si sono accordati su una stringa di  $r+1$  bit, conosciuta come generatore,  $G$ .
- Obiettivi: scegliere  $r$  bit addizionali,  $R$ , in modo che:
  - $\langle D, R \rangle$  siano esattamente divisibili per  $G$  (modulo 2)
  - Il destinatario conosce  $G$ , e divide  $\langle D, R \rangle$  per  $G$ . Se il resto è diverso

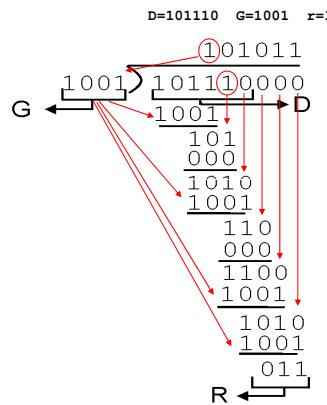
da 0 si è verificato un errore!

- CRC può rilevare errori a raffica inferiori a  $r+1$  bit.
- Nella pratica è molto usato (ATM, HDCL).

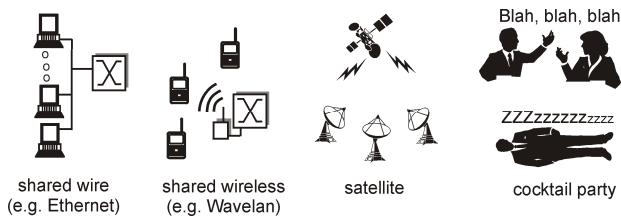
Esempio:

*Vogliamo:*  
 $D \cdot 2^r \text{ XOR } R = nG$   
*Ovvero:*  
 $D \cdot 2^r = nG \text{ XOR } R$   
*Quindi:*  
 se dividiamo  $D \cdot 2^r$  per  $G$ , otteniamo il valore  $R$ .

$R = \text{resto di } \frac{D \cdot 2^r}{G}$



## Protocolli di accesso multiplo



Esistono due tipi di collegamenti di rete:

### 1. Collegamento Punto a Punto (PPP, HDLC)

1.1. impiegato per connessioni telefoniche e nei collegamenti punto a punto tra Ethernet e host.

### 2. Collegamento Broadcast (cavo o canale condiviso)

#### 2.1. Ethernet tradizionale, HFC in upstream, Wireless LAN

L'accesso multiplo prevede l'utilizzo di un canale Broadcast condiviso, sul quale tutti i nodi comunicano liberamente. Ciò fa nascere il problema di **collisioni** quando i nodi ricevono due o più frame contemporaneamente, i quali risulteranno in un certo senso ingarbugliati tra loro. Di conseguenza con una collisione si ha una perdita di frame mentre il canale rimane inutilizzato.

## I protocolli al accesso multiplo

Fissano le modalità con cui i nodi regolano le loro trasmissioni sul canale conclusivo, da notare che per tale coordinazione non e' previsto un canale "out-of-band".

Caratteristiche del canale broadcast con velocità di  $R$  bps :

- Quando un nodo deve inviare dati, questo dispone di un tasso trasmisivo pari a  $R$  bps.
- Quando  $M$  nodi devono inviare dati, questi dispongono di un tasso trasmisivo pari a  $R/M$  bps.
- Il protocollo è decentralizzato:
  - non ci sono nodi master – non c'è sincronizzazione dei clock
- Il protocollo è semplice.

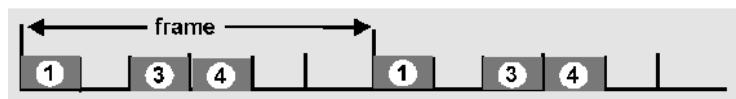
I protocolli si possono classificare in queste 3 categorie:

### Protocollo a suddivisione del canale (Channel partitioning)

[condividono il canale equamente ed efficientemente con carichi elevati, ma risultano inefficienti con carichi non elevati.]

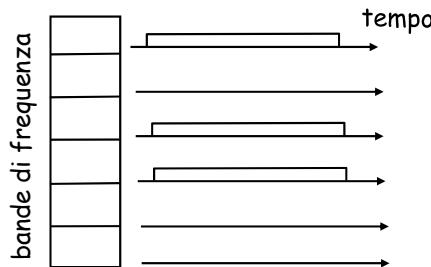
## 1. TDMA [ time division multiplexing ] accesso multiplo a divisione di tempo

- Suddivide il canale condiviso in intervalli di tempo. Se abbiamo K nodi dividiamo il tempo in intervalli ed ogni intervallo in K slot, ogni slot e' assegnato a uno dei K nodi.
- Gli slot non usati rimangono inattivi
- TDM viene utilizzato in quanto riesce a evitare le collisioni ed e' perfettamente imparziale, ciascun nodo ottiene, durante ciascun intervallo di tempo, un tasso trasmittivo di R/N bps, una delle inconvenienze e' che se c'e' un solo nodo che vuol trasmettere, esso dovrà attendere inutilmente il proprio slot ed inoltre non potrà superare il tasso trasmittivo medio di N/R.
- Generalmente, le dimensioni dello slot sono stabilite in modo da consentire la trasmissione di un singolo pacchetto.
- Esempio: gli slot 1, 3 e 4 hanno un pacchetto, 2, 5 e 6 sono inattivi.



## 2. FDMA [ frequency division multiplexing ] accesso multiplo a divisione di frequenza

- Suddivide il canale in bande di frequenza.
- A ciascuna stazione è assegnata una banda di frequenza prefissata. Se ho un canale R allora creo K canali di capacità R/K.
- Vantaggi: no collisioni e divide equamente la larghezza di banda tra i K nodi.
- Svantaggi: la frequenza e' R/K anche se solo uno deve spedire
- Esempio: gli slot 1, 3 e 4 hanno un pacchetto, 2, 5 e 6 sono inattivi.



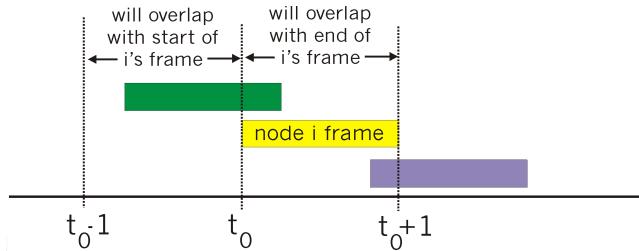
## Protocollo ad accesso casuale (Random access)

**[Efficiente anche con carichi non elevati, ma con carichi elevati sottoposto ad un eccesso di collisioni.] I canali non vengono divisi e si può verificare una collisione. – I nodi coinvolti ritrasmettono ripetutamente i pacchetti dopo aver aspettato un tempo casuale (random delay).**

1. Quando un nodo deve inviare un pacchetto trasmette sempre alla massima velocità consentita dal canale, cioè R bps e non vi è coordinazione a priori tra i nodi. Quindi due o più nodi trasmittenti → "collisione"
2. Il protocollo ad accesso casuale definisce: Come rilevare un'eventuale collisione e come ritrasmettere se si è verificata una collisione.
3. Esempi di protocolli ad accesso casuale: slotted ALOHA – ALOHA – CSMA, CSMA/CD, CSMA/CA

### 4. ALOHA [successo 20%]

- 4.1. Quando arriva il primo pacchetto, lo trasmette immediatamente e integralmente nel canale broadcast;
- 4.2. Controlla retroazione;
- 4.3. Elevate probabilità di collisione:
  - 4.3.1. Il pacchetto trasmesso a t0 si sovrappone con la trasmissione dell'altro pacchetto inviato in [t0-1, t0+1].



- $P(\text{trasmissione con successo da un dato nodo}) = P(\text{il nodo trasmette}) * P(\text{nessun altro nodo trasmette in } [t0-1, t0]) * P(\text{nessun altro nodo trasmette in } [t0, t0+1]) = p \cdot (1-p)^{N-1} \cdot (1-p)N-1 = p \cdot (1-p)^2(N-1)$
- ... scegliendo  $p$  migliore e lasciando  $N \Rightarrow$  infinito ...  $= 1/(2e) = 0,18$

#### 4.3.2.

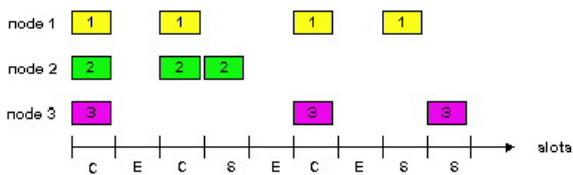
- 4.3.3. In genere il protocollo avrà una **buona probabilità** di riuscita quando il traffico è *molto basso* (poche stazioni, e trasmissioni rarefatte). In casi di *alti carichi*, le collisioni cresceranno, e con esse le ritrasmissioni, fino a rendere la trasmissione di fatto **impossibile**. In protocollo ha quindi una **efficienza massima** quando per ogni tempo di frame ci sono  $G = 0.5$  tentativi di trasmissione; in questa condizione l'efficienza di utilizzo del canale è pari circa al 18%.

### 5. Slotted ALOHA

5.1. Assumiamo che:

- 5.1.1. tutti i pacchetti hanno la **stessa dimensione L**
- 5.1.2. il tempo è **suddiviso in slot ed ogni slot equivale al tempo di trasmissione di un pacchetto L/R secondi**
- 5.1.3. I Nodi iniziano la trasmissione dei pacchetti **solo all'inizio degli slot** e i vari nodi sono sincronizzati
- 5.1.4. Ed in fine se in uno slot due o più pacchetti collidono, i nodi coinvolti **rilevano l'evento prima del termine** dello slot.

5.2. **Operazioni:** Quando a un nodo arriva un nuovo pacchetto da spedire, il nodo attende fino all'inizio dello slot successivo. Se non si verifica una collisione il nodo può trasmettere un nuovo pacchetto nello slot successivo, se invece si verifica una collisione, il nodo la rileva prima della fine dello slot e ritrasmette con probabilità  $P$  il suo pacchetto durante gli slot successivi.



**Pro:** Consente a un singolo nodo di trasmettere continuamente pacchetti alla **massima velocità**. È fortemente **decentralizzato**, ciascun nodo rileva le collisioni e decide indipendentemente quando ritrasmettere e inoltre è estremamente semplice. Richiede che i nodi sincronizzino le loro trasmissioni a partire dall'inizio degli slot.

**Contro:** Una certa frazione degli slot presenterà collisioni e di conseguenza andrà "sprecata". Un'altra frazione degli slot rimane vuota, quindi inattiva.

- 5.2.1. L'efficienza è definita come la frazione di slot vincenti in presenza di un elevato numero di nodi attivi, che hanno sempre da spedire un elevato numero di pacchetti. In questo caso, il caso migliore sono il 37% degli slot i quali compiono lavoro utile.
- 5.2.2. La probabilità che un nodo arbitrario abbia successo è  **$Np(1 - p)^{N-1}$** .

### Protocolli con rilevamento delle portate

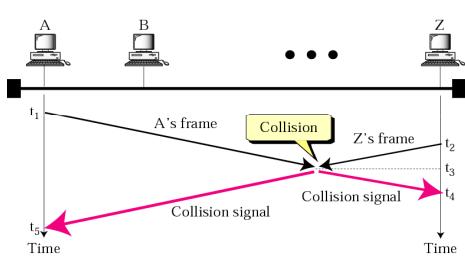
[una delle maggiori inefficienze dei protocolli aloha è determinata dal fatto che la trasmissione viene fatta senza controllare prima se il canale è libero]

Ci sono due regole importanti:

- La **Rilevazione delle portate (carrier sensing)**, ovvero si ha che prima di trasmettere, il nodo ascolterà se il canale risulta libero.
- la **Rilevazione delle collisioni ( collision detection )**, ovvero nel momento in cui un nodo trasmette, esso rimane contemporaneamente in ascolto, e se rileva che un altro nodo sta' trasmettendo un frame che interferisce con il suo, cessa immediatamente la trasmissione. Miigiorandone le prestazioni, evitando l'inutile trasmissione dell'intero frame danneggiato dall'interferenza con un altro frame.

## 6. CSMA

- 6.1. si pone in ascolto prima di trasmettere, se rileva che il canale e' libero, trasmette l'intero pacchetto, se il canale sta già trasmettendo, il nodo aspetta un altro intervallo di tempo (casuale).

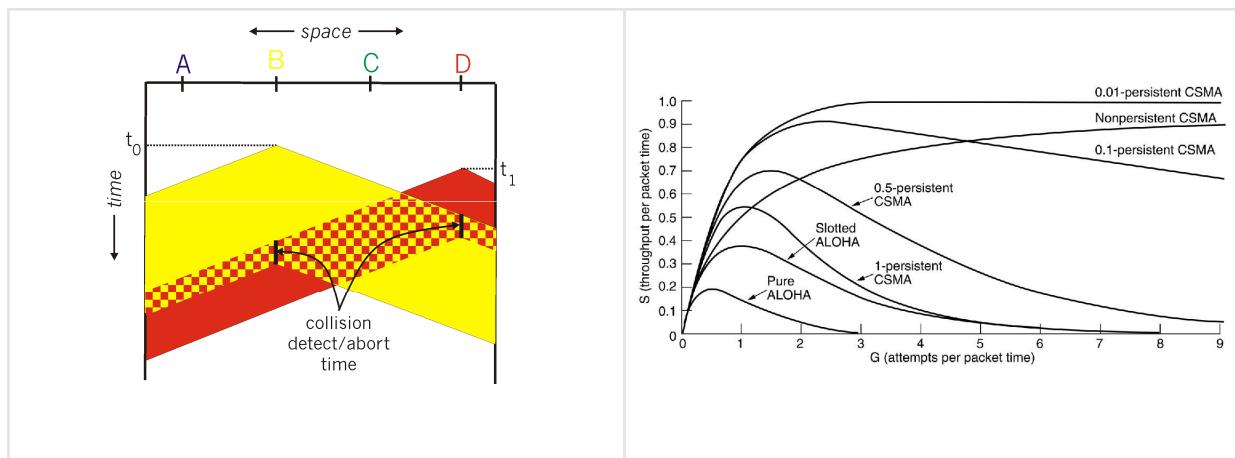


6.2. Ciò non basta ad evitare le collisioni, infatti il ritardo di propagazione fa sì che due nodi non rilevino la reciproca trasmissione. Da ciò possiamo notare che la distanza e il ritardo di propagazione giocano un ruolo importante nella determinare la probabilità di collisione

6.3. Nel caso in cui un nodo rileva una collisione, cessa immediatamente la trasmissione.

## 7. CSMA / CD collision detection (Ethernet)

- 7.1. caratterizzato dal rilevamento della portante differito, ovvero si ha che riesce a rilevare la collisione in poco tempo, annulla la trasmissione non appena si accorge che c'e' un'altra trasmissione in corso.
- 7.2. La rilevazione delle collisioni e' facile nelle LAN cablate e più difficile nelle LAN wireless.



## Protocollo a rotazione. (taking-turn)

[ Prende il meglio dei due protocolli precedenti ], i protocolli aloha e csma, possiedono la proprietà che garantisce che con un nodo si abbia, un throughput di  $R$  bps, ma non garantiscono la proprietà auspicabile di un throughput di  $M/R$  bps con  $M$  nodi attivi. Così sono nati i protocolli a rotazione dove ciascun nodo ha il suo turno di trasmissione, ma i nodi che hanno molto da trasmettere possono avere turni più lunghi.

## 1. Polling ( di sondaggio )

- 1.1. **Master/Slave**, Uno dei nodi e' designato ad arbitro/sonda e decide chi può spedire. Esso invia un messaggio ad un nodo indicandogli fino a che dato numero di frame può spedire e in caso lui spedisce, successivamente determina che la spedizione e' finita dalla mancanza di segnale sul canale.

- 1.2. **Vantaggi:** Elimina le collisioni, elimina gli slot vuoti
- 1.3. **Svantaggi:** pago il ritardo dei messaggi di polling, se per esempio vi e' un solo nodo attivo esso avra' un throghput inferiore a R bps dato che il master deve testare anche tutti gli altri nodi prima di dargli di nuovo la parola. Se il nodo centrale si guasta l'intero canale diventa inattivo.

## 2. Token-Passing

- 2.1. I nodi si passano un **token** (messaggio di controllo) e chi lo possiede può inviare. Il passaggio segue un ordine prefissato.
- 2.2. Tale protocollo e' decentralizzato, altamente efficiente ma il guasto di un nodo può metter fuori uso l'intero canale. Inoltre sono previste delle tecniche per rigenerare il token nei casi in cui esso venga smarrito.

## 3. Token ring ( round robin )

- 3.1. questo protocollo prevede l'utilizzo di una topologia ad anello, sull'anello circola un piccolo frame, detto token (gettone) che le stazioni ricevono da una parte e ritrasmettono dall'altra in continuazione. Una stazione e' autorizzata a trasmettere dati solo quando e' in possesso del token. La stazione riceve il token, lo trattiene ed inizia a trasmettere dati terminata la trasmissione, ritrasmette il token in coda ai frame di dati esistono specifiche a 4 e 16 Mbps.
- 3.2. Il protocollo token ring (come tutti quelli a turno) e' poco efficiente in condizioni di basso carico
  - la stazione che deve trasmettere deve attendere di ricevere il token (o in generale deve attendere il suo turno) prima di poterlo fare, anche se il canale non e' occupato
- 3.3. In condizioni di carico elevato, quando tutti vogliono trasmettere, l'efficienza del protocollo sfiora il 100%
  - I solo overhead e' dovuto alla necessita' che ha una stazione di identificare il token prima di poter trasmettere
  - in questi protocolli il token e' scelto in modo opportuno per minimizzare l'overhead
- 3.4. Una importante caratteristica di questo genere di protocolli e' la possibilità di valutare un tempo massimo di ritardo per le trasmissioni
  - una stazione che desidera trasmettere dovrà attendere al piu' N tempi di trasmissione (uno per stazione, nel caso tutti debbano trasmettere) prima che tocchi nuovamente ad essa
  - questo permette l'utilizzo del protocollo in situazioni in cui i tempi di risposta possono essere determinanti (ad esempio una catena di montaggio)

## Tecnologie LAN

Sono su reti che si estendono in un'area geografica limitata. Hanno velocità trasmissive elevate (10/100/1000). E sostanzialmente ne esistono 2 categorie:

- Accesso multiplo: Ethernet
- Token: token-ring e FDDI.

## FDDI e Token-ring

In una LAN token-ring, N nodi sono collegati in un anello da connessioni dirette. La tipologia definisce l'ordine del passaggio del token: quando un nodo ottiene il token e invia un frame, questo si proponga all'interno dell'intero anello, creando quindi un canale broadcast virtuale. La particolarita' di questo protocollo e' che esso prevede che il nodo mittente ha la responsabilità di rimuoverlo dall'anello.

## Indirizzi a Livello di Link

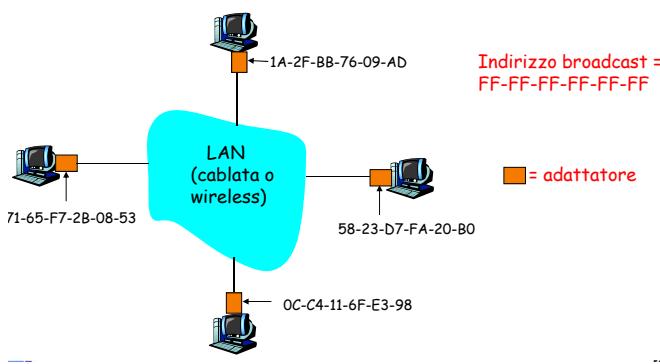
Ricapitolando, abbiamo fin ora visto soltanto indirizzi IP a 32 bit, indirizzi a livello di rete, quali hanno una struttura gerarchica e devono esser aggiornati. Adesso vediamo altri tipi di indirizzi, come gli indirizzi del livello Link, i quali sono attribuiti alle schede dei nodi, ed prendono il nome di indirizzi MAC.

### Indirizzi MAC e ARP

Un **indirizzo MAC (media access control)** e' caratterizzato da un'analogia con il codice fiscale di una persona, ha una struttura orizzontale/piatta (e' possibile spostare una scheda LAN da una LAN a un'altra) e non varia a seconda del luogo in cui una persona si trasferisce. Non esistono due adattatori con lo stesso indirizzo.

Per molte LAN e' lungo 48 bit (6 byte) prevedendo quindi un'insieme di  $2^{48}$  indirizzi. E' espresso in notazione esadecimale, dove per ogni byte si hanno due cifre esadecimale, era permanente, ora e' modificabile via sw.

Ciascun adattatore di una LAN ha un indirizzo LAN univoco .

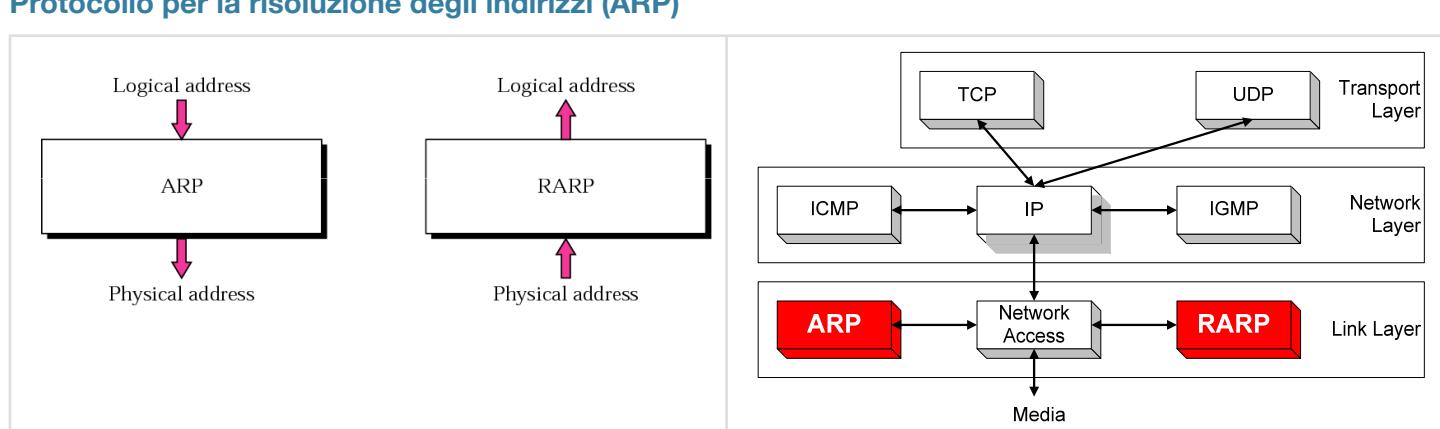


E' unico, le società comprano blocchi di  $2^{24}$  indirizzi da IEEE, la quale riserva questo blocco e fissa i primi 24 bit dell'indirizzo, lasciando alla società il compito di assegnare agli adattatori una combinazione dei restanti 24 bit.

L'indirizzo MAC, viene inserito nell'header del frame, e solamente se il ricevente ha il medesimo indirizzo MAC, lo passa al livello superiore, altrimenti lo scarta.

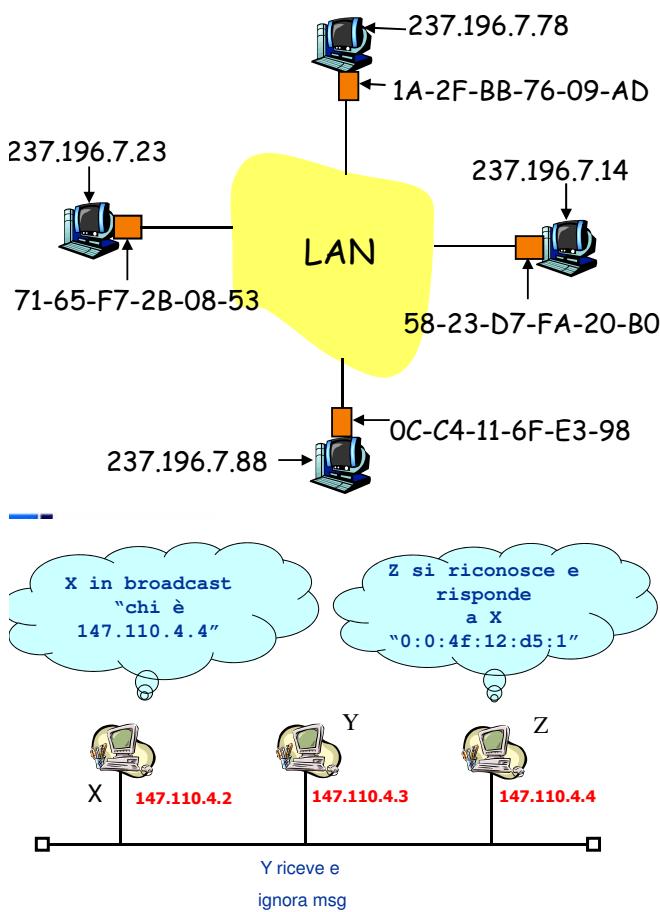
Durante la comunicazione LAN, i frame ricevuti da ogni adattatore vengono esaminati per verificare che l'indirizzo MAC corrisponda con il proprio. Nel caso in cui volessimo che un frame sia letto da tutti dovremmo inserire come indirizzo del destinatario l'indirizzo MAC broadcast (esempio LAN = una stringa in cui i 48 bit sono tutti 1 => FF-FF-FF-FF-FF-FF).

### Protocollo per la risoluzione degli indirizzi (ARP)



**ARP ( address resolution protocol)**, protocollo con il compito di condividere gli indirizzi MAC degli adattatori presenti nella sottorete di appartenenza.

Protocollo utilizzato da Internet per convertire indirizzi MAC e IP. Un nodo quando deve spedire, preleva l'IP ad esempio dal DNS (**tutte le reti**) e poi prende il MAC dalla tabella ARP (**applicabile solo nella sottorete**). Tali informazioni gli sono necessarie per creare il frame da inviare, infatti necessita sia dell'indirizzo IP che del MAC associato.



Per determinare l'indirizzo MAC a partire dall'Indirizzo IP si hanno a disposizione tali componenti:

- Ogni nodo IP (host,router) nella RAM ha una **Tabella ARP**.
- La Tabella ARP, contiene la corrispondenza tra indirizzi IP e MAC e il TTL time to live (tipicamente 20 min).
- Non necessariamente ha una voce per ogni nodo della sottorete.
- La tabella ARP è detta **"Plug-and-play"**, dato che un nodo si costruisce automaticamente la sua tabella e non deve esser configurata dall'amministratore del sistema.

#### Esempio:

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. Per fare ciò:

- B esamina il campo "operazione"
  - se op= risposta ...
  - se op = richiesta allora

- se (ind IP = IP B) inverte ind mitt e destinatario
  - inserisce il proprio indirizzo MAC nel campo mittente
  - Imposta il tipo di operazione = risposta
  - estrae il MAC del mittente e lo memorizza nella cache.
- il frame viene inviato all'indirizzo MAC di A.

Nell'header del frame ethernet il campo "type" è ARP (=0x806)

Nel messaggio ARP il campo "hardware type" è Ethernet (= 1)

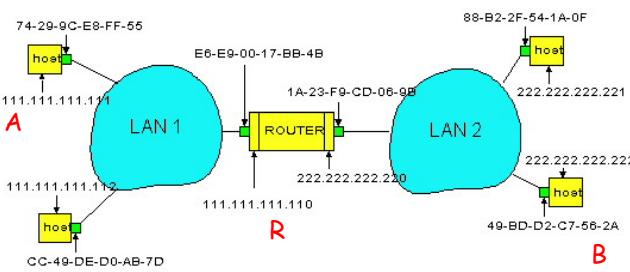
Protocol type IP = 0x800

Hw len Ethernet = 6 byte

ProtlenIP=4byte

ARP Operation : ARP req = 1, ARP reply = 2 – RARP req = 3, RARP reply = 4

## Invio verso un nodo esterno alla sottorete



Se vogliamo che A invii un datagramma a B attraverso R, ipotizziamo che A conosca l'IP di B.

Inizialmente vediamo l'esistenza di due tipi di nodi, gli host e i router, i quali hanno un'indirizzo IP per ogni interfaccia che, a sua volta, dispone anche di propri moduli ARP e di propri adattatori.

In questo caso si avranno due tabelle ARP nel router R, una per rete IP (LAN).

- A crea un datagramma con origine A, e destinazione B.
- A usa ARP per ottenere l'indirizzo MAC di R.
- A crea un collegamento a livello di rete con l'indirizzo MAC di destinazione di R, il frame contiene il datagramma IP da A a B.
  - L'adattatore di A invia il datagramma.
  - L'adattatore di R riceve il datagramma.
- R rimuove il datagramma IP dal frame Ethernet, e vede che la sua destinazione è B.
  - R usa ARP per ottenere l'indirizzo MAC di B.
  - R crea un frame contenente il datagramma IP da A a B IP e lo invia a B.

vedi allegato 3

**Arp ok**

## Ethernet

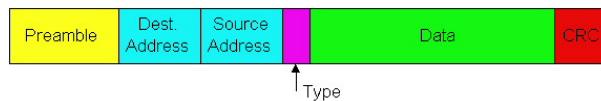
Detiene una posizione dominante nel mercato delle LAN cablate. È stata la prima LAN ad alta velocità con vasta diffusione. Più semplice e meno costosa di token ring, FDDI e ATM. Sempre al passo dei tempi con il tasso trasmittivo.

### Tipologia a stella

Quasi tutte le odierne reti Ethernet sono progettate con topologia a stella, con al centro della stella un hub (dispositivo fisico che agisce sui singoli bit, piuttosto che sui frame, amplificandone la potenza e ritrasmettendolo) o commutatore(switch), quest'ultimo e' stato introdotto nel 2000, portando con sé numerosi miglioramenti, infatti con il suo utilizzo non solo e' stato eliminato il problema delle collisioni, ma uno switch e' un vero e proprio commutatore di pacchetti store-and-forward (lavora fino al livello 2, i router fino a livello 3).

### Struttura dei pacchetti Ethernet

L'adattatore trasmittente incapsula i datagrammi IP in un **pacchetto Ethernet**.



**Preamble:** I pacchetti Ethernet iniziano con un campo di 8 byte: sette hanno i bit 10101010 e l'ultimo e' 10101011, ciò viene effettuato per "attivare" gli adattatori del ricevente e sincronizzare i loro orologi con quello del trasmittente. Ciò viene effettuato per gestire la variazione rispetto all'esatto tasso prefissato di trasmissione, che non e' conosciuto a priori dagli altri adattatori sulla LAN. Gli ultimi 2 bit degli 8 byte del preambolo avvisano l'adattatore che i successivi 6 byte sono l'indirizzo di destinazione. Le schede individuano la terminazione di un pacchetto semplicemente rilevando l'assenza di corrente.



**Campo dati:** Da 46 a 1500 byte, esso contiene il datagramma IP, dato che l'unita' massima trasferita, MTU, per Ethernet e' di 1500 byte, qualora il datagramma fosse più grande si avrà il fenomeno della frammentazione, invece nel caso in cui sia piu' piccolo esso verrà completato con bit di completamento(stuffed).

**Indirizzo di destinazione:** 6 byte; quando un adattatore riceve un pacchetto contenente l'indirizzo di destinazione o con l'indirizzo di broadcast, trasferisce il contenuto del campo dati del pacchetto a livello di rete. I pacchetti con altri indirizzi MAC vengono ignorati.

**Indirizzo Sorgente:** 6 byte, indirizzo MAC della scheda di rete che trasmette il pacchetto.

**Campo tipo:** 2 byte, consente ad Ethernet di supportare vari protocolli di rete (de/multiplexare i protocolli), ogni protocollo di rete ha il proprio numero di tipo.Come avviene per ARP, IP ecc.

**Controllo CRC:** 4 byte, consente all'adattatore ricevente di rilevare la presenza di un errore nei bit del pacchetto andando a calcolare il bit del campo CRC alla ricezione del frame e confrontandolo con quello precedentemente calcolato dal trasmittente.

## Servizio senza connessione non affidabile

Senza connessione, dato che non e' previsto nessun handshake prima dell'invio del pacchetto; Non affidabile, l'adattatore ricevente non invia un riscontro ne se il pacchetto supera il controllo CRC ne in caso contrario(se tali problemi non vengono rilevati, di solito tramite TCP, potrebbe accusare problemi a causa dell'incompletezza dei dati).Cio' viene effettuato mantenere Ethernet semplice ed economica. E comunque utilizzando come protocollo di trasporto TCP, si hanno delle ritrasmissioni che comunque rimangono "nascoste" ad Ethernet il quale non riesce a distinguere tra una ritrasmissione e una semplice trasmissione.

## Ethernet utilizza il protocollo di accesso multiplo CSMA/CD

Quando i nodi di una LAN Ethernt sono connessi tramite un hub, invece che con uno switch, abbiamo delle LAN broadcast ed e' per questo che e' necessario un protocollo ad accesso multiplo. Gli adattatori Ethernet misurano il livello di tensione prima e durante la trasmissione, comunque le principali caratteristiche sono:

- L'adattatore può iniziare a trasmettere in qualsiasi momento, ovvero non utilizza slot temporali.
- Non può trasmettere un pacchetto quando rileva che altri adattatori stanno trasmettendo: **rilevazione della portante**.
- Annulla la propria trasmissione non appena rileva che un altro adattatore sta trasmettendo: **rilevazione di collisione**.
- Prima di ritrasmettere, l'adattatore resta in attesa per un lasso di tempo stabilito arbitrariamente.

Per questi motivi si ha che CSMA/CD e' preferito ad ALHOA in ambienti locali, infatti CSMA/CD con dei ritardi di propagazione limitati ha un'efficienza che si approssima al 100%. Occorre pero' che gli adattatori Ethernet siano in grado di rilevare la presenza di errori e riscontrare eventuali collisioni durante la trasmissione.

Le fasi operative del protocollo sono:

1. L'adattatore riceve un datagramma di rete dal nodo cui è collegato e prepara un pacchetto Ethernet sistemandolo in un Buffer.
2. Se il canale è inattivo, inizia la trasmissione. Se il canale risulta occupato, resta in attesa fino a quando non rileva più il segnale.
3. Verifica, durante la trasmissione, la presenza di eventuali segnali provenienti da altri adattatori. Se non ne rileva, considera il pacchetto spedito.
4. Se rileva segnali da altri adattatori, interrompe immediatamente la trasmissione del pacchetto e invia un segnale di disturbo (jam) di 48 bit .
5. L'adattore rimane in attesa (entra nella fase di **attesa esponenziale**). Quando riscontra l'n-esima collisione consecutiva, stabilisce un valore k tra {0,1,2,...,2m-1}. L'adattatore aspetta un tempo pari a K volte 512 bit e ritorna al Passo 2.

**Segnale di disturbo (jam):** la finalità è di avvisare della collisione tutti gli altri adattatori che sono in fase trasmisiva.

**Tempo di Bit :** corrisponde a 0,1 microsec per Ethernet a 10 Mbps; per K=1024, il tempo di attesa è di circa 50 msec.

**Ethernet** stabilisce una distanza massima tra i nodi.

#### Attesa esponenziale:

- Obiettivo: l'adattatore prova a stimare quanti siano gli adattatori coinvolti. (in presenza di un numero ridotto di adattatori sarebbe accettabile anche un numero K di attesa, ma in condizioni opposte non andrebbe più bene)
  - Se sono numerosi il tempo di attesa potrebbe essere lungo.
- Prima collisione: sceglie K tra {0,1}; il tempo di attesa è pari a K volte 512 bit.
- Dopo la seconda collisione: sceglie K tra {0,1,2,3}...
- Dopo dieci collisioni, sceglie K tra {0,1,2,3,4,...,1023}.

#### Efficienza di Ethernet

Se un solo nodo deve spedire, trasmette alla massima velocità, ma se sono di più l'effettiva velocità del canale è inferiore.

Quindi definiamo l'efficienza la frazione di tempo medio durante la quale i frame sono trasferibili sul canale senza collisioni con un buon numero di nodi attivi che devono spedire un'elevata quantità di frame. Per approssimazione usiamo prop tempo massimo di propagazione tra due adattatori.

- **T<sub>prop</sub>** = tempo massimo che occorre al segnale per propagarsi fra una coppia di adattatori.
- **T<sub>trasm</sub>** = tempo necessario per trasmettere un pacchetto della maggior dimensione possibile.

$$\text{efficienza} = \frac{1}{1 + 5t_{prop} / t_{trasm}}$$

- Si evince che quando t<sub>prop</sub> tende a 0, l'efficienza tende a 1.
- Al crescere di t<sub>trasm</sub>, l'efficienza tende a 1.
- Molto meglio di ALOHA, ma ancora decentralizzato, semplice, e poco costoso.

#### La probabilità di eventuali collisioni viene così calcolata:

Ethernet ok

# Sicurezza nelle Reti

## Definizione

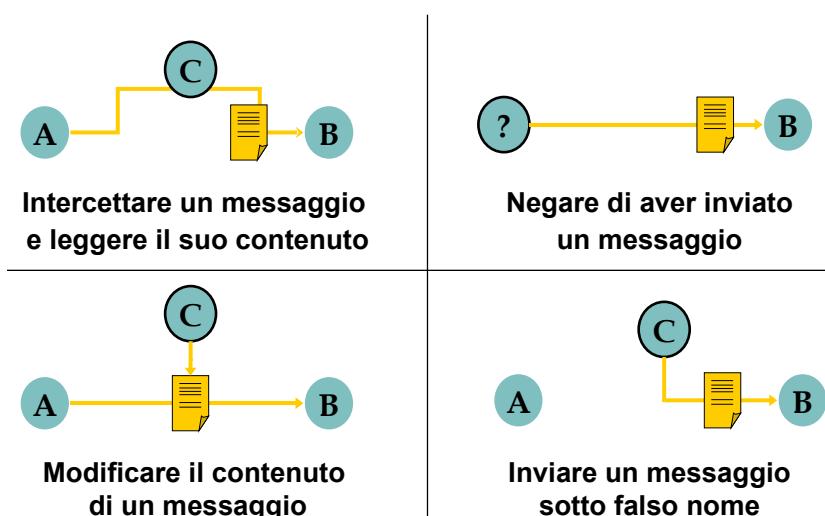
Sicurezza = Assenza di rischio, ma in informatica assume un altro significato:

**Prevenzione o protezione contro l'accesso, distribuzione o alterazione di risorse/informazioni da parte di utenti non autorizzati.**

**Def:** Abilità di un sistema di proteggere informazioni, risorse ed il sistema stesso, rispetto alle nozioni di :

- Confidentialità (confidentiality), solo il mittente e destinatario dovrebbero poter comprendere il messaggio.
- Integrità (integrity), vogliamo che il messaggio non subisca alterazioni durante la transizione dovute a manipolazioni
- Autenticazione (authentication), mittente e destinatario devono esser sicuri della loro identità.
- Controllo degli Accessi (control access), sicurezza operativa, ovvero vengono usati firewall e sistemi di rilevamento delle intrusioni
- Non ripudio (non-repudiation)
- Disponibilità (availability)
- Privacy (privacy)

## Le principali minacce nello scambio messaggi



## Sicurezza dei protocolli di rete

il protocollo attualmente più diffuso è il TCP/IP, esso ha delle debolezze intrinseche perché fu creato nel 1974 pensando all'efficienza delle connessioni piuttosto che alla sicurezza. Queste debolezze permettono attacchi di diverso tipo.

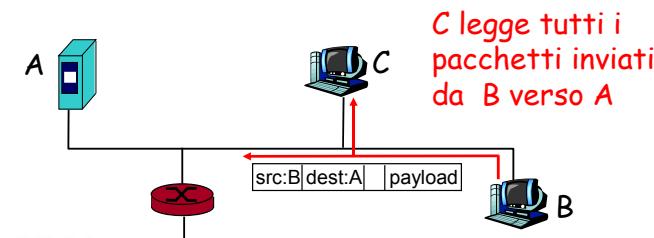
**Attacchi passivi:** sottrarre informazioni senza interferire sul sistema

**Attacchi attivi:** interferire sul sistema con vari scopi

Anche se tale classificazione risulta esser storica, dato che adesso sono praticamente tutti attivi.

## Minacce alla sicurezza in Internet

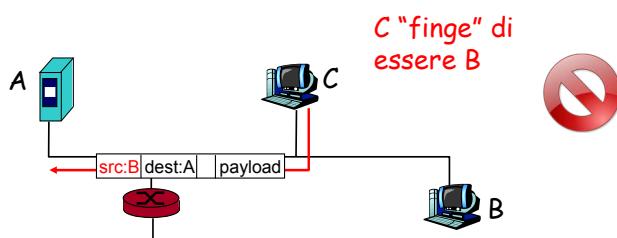
### Packet sniffing (to sniff = odorare)



Evidente in mezzi condivisi. Un adattatore di rete(NIC) programmato ad hoc, legge tutti i pacchetti in transito e tutti i dati non cifrati possono esser letti.

Soluzione: Crittografia

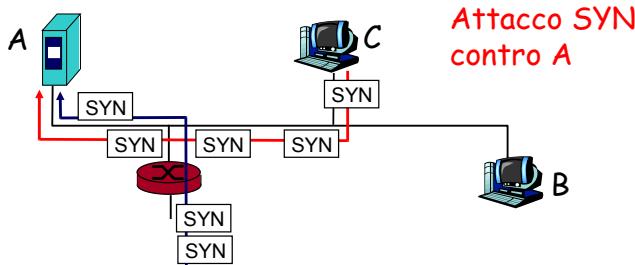
### IP Spoofing



Un host genera pacchetti IP con indirizzi di sorgente falsi, ed il ricevente non e' in grado di stabilire se l'origine dei pacchetti sia quella autentica

Soluzione: Autenticazione "end-to-end"

### Denial of Service (DoS)



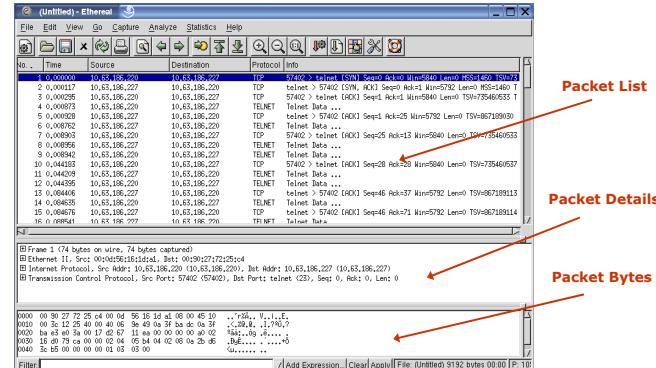
Flusso di pacchetti "maligni" che sommerge il ricevente. Distributeb Dos (DDoS), attacco coordinato multiplo.

Soluzione: Autenticazione "end-to-end"

Ne manca uno hisack.. e DoS manca la trattazione di due casi snurf arrack e cyn flooding

## Strumenti Utilizzati

Ethereal (wireshark) Network Analyzer

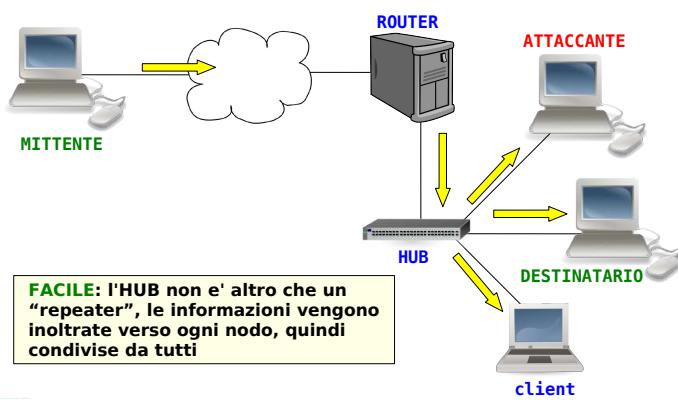


• Analizza traffico di rete

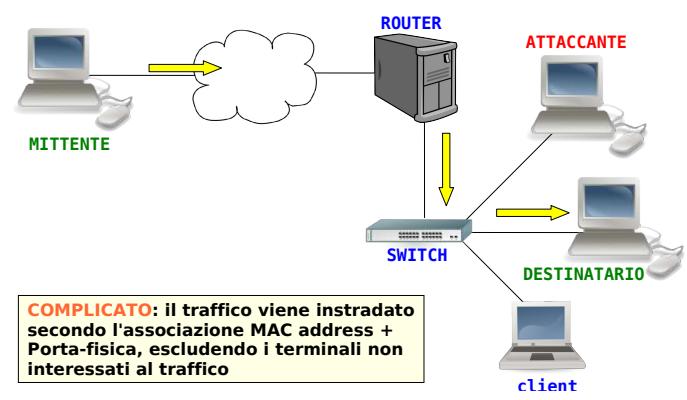
- In grado di catturare pacchetti indirizzati ad altri host (Promiscous\_Mode)
- Qualsiasi protocollo anche personale (plug-in)
- Packet Filtering

## Sniffing su reti

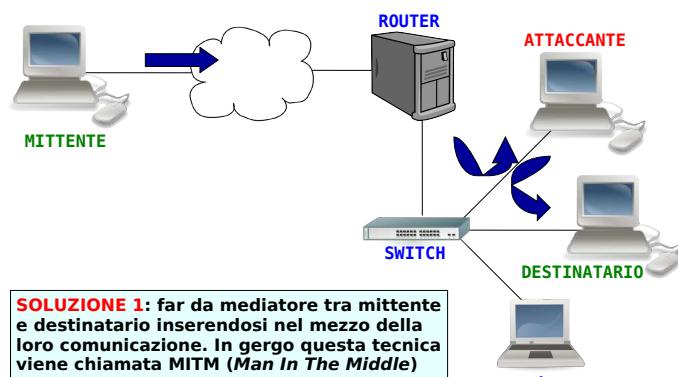
### Sniffing su reti non-switchate



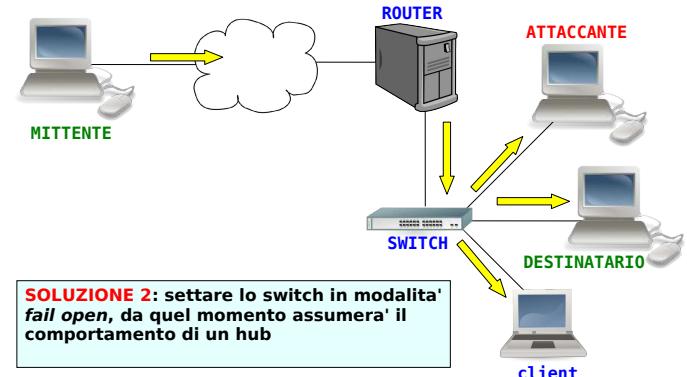
### Sniffing su reti switchate



### Sniffing su reti switchate



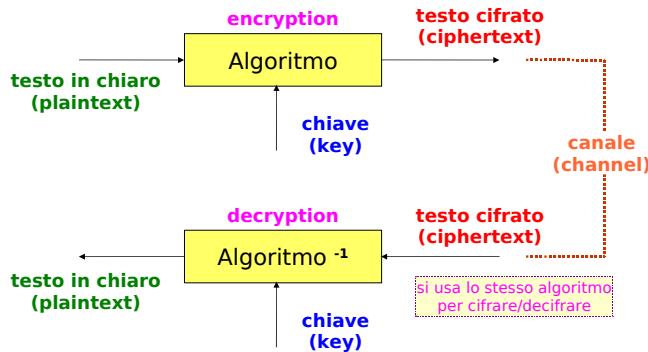
### Sniffing su reti switchate



## Crittografia

\*\* Purtroppo questo e' l'ultima parte di riassunto che ho fatto e come potete immaginare e' abbastanza incasinata e poco chiara..

Schema a blocchi:



La sicurezza in crittografia e' valutata in base alle **risorse di tempo e di calcolo** necessarie per dedurre informazioni.

- Ogni protocollo crittografico puo' esser "rotto" con sufficienti risorse di tempo e di calcolo
- Se un algoritmo puo' esser "rotto" usando per 30 anni un sistema di calcolo del valore di 10 miliardi di euro allora puo' esser ritenuto sicuro.

La sicurezza di un algoritmo dipende dal campo di applicazione e dal numero di possibili chiavi: 20 bit = circa un milione di chiavi non e' sicuro, 56 bit poco sicuro, 512 bit oggi e' sicuro domani non si sa.

La **Crittoanalisi** studia le modalita' di attacco dei protocolli crittografici

Diversi tipi di attacco basati su

- **Conoscenza** di testo cifrato
- **Conoscenza** testo in chiaro e corrispondente testo cifrato
- **Scelta** di testo in chiaro e conoscenza del corrispondente testo cifrato
- **Scelta** di testo crittato e conoscenza del corrispondente testo in chiaro

## Algoritmi di crittografia

Gli algoritmi di crittografia possono esser classificati come:

- **Simmetrici**, anche detti **a chiave segreta/privata** i quali usano la stessa chiave per codificare e decodificare
- **Asimmetrici**, anche detti **a chiave pubblica**, i quali usano due chiavi distinte, una per codificare e una per decodificare.

Tutti codificano il testo a blocchi es 66, 128 byte.

**Cifrario di cesare:** Shiftiamo di K posizioni, K = 3, CIAO = FLDR

**Cifrario Monoalfabetico:** scambiamo una lettera dell'alfabeto con un'altra, esso presenta una maggiore efficienza, in quanto abbiamo  $10^{26}$  permutazioni nel alfabeto inglese. Possiamo ridurre il numero di tentativi andando ad analizzare le lettere che si ripetono molte volte: inglese = e, t; italiano = che, mente, zione...

**Cifratura Polialfabetica:** molte cifrature monolafabetiche a seconda della posizione che occupano.

**Cifrari a Blocchi:** messaggio diviso in tot blocchi di K bit, dove ciascun blocco cifrato indipendentemente dagli altri.  $2^k$  bit corrispondenze. Robusto ma difficile da implementare. DES, 3DES, AES tutti quanti usano una stringa di bit come chiave ma con dimensioni differenti sia per i blocchi che per la chiave.

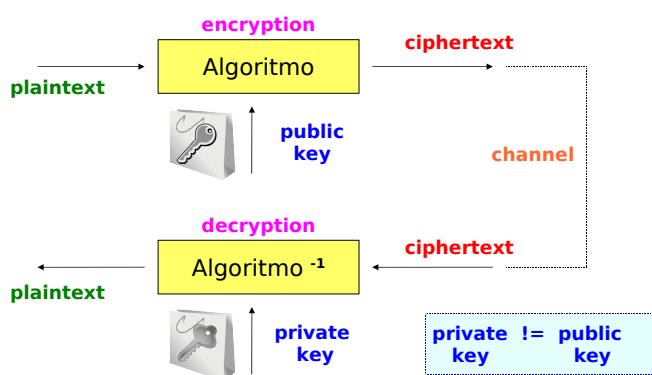
## La cripitura a chiave asimmetrica

## Crittografia a chiave Pubblica

L'utilizzo della crittografia a chiave pubblica e' concettualmente molto semplice, ogni utente ha:

- **Ogni utente ha due chiavi** 
- **Una delle chiavi è resa pubblica** 
- **La chiave segreta (privata) è nota soltanto al suo proprietario** 

Useremo la notazione  $K+R$  per indicare la chiave pubblica di Roberto e  $K-R$  per indicare la sua chiave privata.



Ovvero, applicando la chiave pubblica di Roberto a un messaggio  $m$  e quindi applicando la chiave privata di Roberto alla versione cifrata di  $m$ , si ottiene nuovamente  $m$

$$K-R(K+R(m)) = m$$

Tale algoritmo propone immediatamente alcune obiezioni:

- l'intruso che intercetta il messaggio potra' venire a conoscenza sia della chiave pubblica del destinatario sia dell'algoritmo che il mittente ha usato per la codifica. Così facendo Tommaso puo' tentare una attacco con testo in chiave scelto, impiegano l'algoritmo di cifratura standard e al chiave pubblica di Roberto per codificare i messaggi.

- Affinché tutto funzioni correttamente, la selezione delle chiavi di codifica e decodifica devono rendere praticamente impossibile all'intruso determinare la chiave privata di Roberto per codificare i messaggi.

**L'algoritmo RSA** e' diventato praticamente sinonimo di crittografia a chiave pubblica. Vi sono due punti principali su cui si basa RSA:

- la scelta della chiave pubblica e di quella privata
- gli algoritmi di cifratura e di decifratura

Per ottenere la chiave pubblica e privata il destinatario deve seguire i seguenti passi:

1. Scegli due numeri primi  $p, q$ . (es., 1024 bit ciascuno)
2. Calcola  $n = pq$  e  $\phi = (p-1)(q-1)$
3. Scegli  $e$  (con  $e < n$ ) *encryption* privo di fattori comuni con  $\phi$ . (es.  $e=99$  e  $\phi=100$  non sono primi ma non hanno fattori comuni)
4. scegli  $d$  *decryption* tale che  $ed \equiv 1 \pmod{\phi}$ .
5. **La chiave Pubblica è  $(n,e)$ , quella Privata è  $(n,d)$ .**

Come vengono eseguite la cifratura e la decifratura:

Dati  $(n,e)$  e  $(n,d)$  calcolati come al punto precedente

Per cifrare la stringa di bit corrispondente a  $m$ , calcola  
 $c = m^e \text{ mod } n$  (resto di  $m$  diviso  $n$ ) <sup>$e$</sup>

Per decifrare  $c$ , calcola  
 $m = c^d \text{ mod } n$  (resto di  $c$  diviso  $n$ ) <sup>$d$</sup>

Questa è vera!  $m = (m^e \text{ mod } n)^d \text{ mod } n$

## RSA: Esempio

Bob sceglie  $p=5$ ,  $q=7$ . Allora  $n=35$ ,  $z=24$ .

$$\begin{aligned} e &= 5 \\ d &= 29 \text{ (ed-1 esattamente divisibile per } z) \end{aligned}$$

Cifra:  $\begin{array}{cccc} \text{lettera} & m & m^e & c = m^e \text{ mod } n \\ | & 12 & 1524832 & 17 \end{array}$

Decifra:  $\begin{array}{ccc} c & c^d & m = c^d \text{ mod } n \text{ letter} \\ 17 & & 12 \quad | \end{array}$

481968572106750915091411825223072000



## RSA: Implementazione

### Come trovare le chiavi

- Genera due primi  $p,q$
- Calcola  $n = pq$
- Scegli a caso  $e$
- Se  $\text{MCD}(e, (p-1)(q-1)) = 1$  (MCD= Massimo Comun Divisore)
  - Allora  $d = e^{-1} \text{ mod}(p-1)(q-1)$
  - Altrimenti vai a 3

## RSA: Implementazione (cont.)

### Come trovare un numero primo

- Scegli un numero dispari a caso
- Verifica se è primo (Esistono algoritmi veloci → polinomiali)

I numeri primi sono frequenti:  $\pi(x) = \text{n. primi tra } 2 \text{ e } x$ ,  $\pi(x)$  appross.  $x/\ln x$

Es.: in  $[2^{511}, 2^{512}]$  la probabilità che un numero dispari sia primo è  $> 1/178$

## RSA: Implementazione (cont.)

### Come calcolare $\text{MCD}(x,y)$ : algoritmo di Euclide

Come calcolare la potenza: il metodo di elevazione a potenza modulare esegue un numero logaritmico di moltiplicazioni

#### ESEMPIO

$$40 = 0 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 + 0 \cdot 2^4 + 1 \cdot 2^5$$

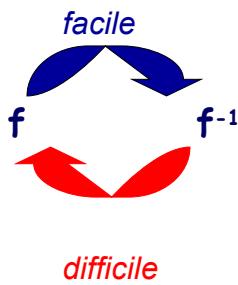
$$x^{40} = (x^1)^0 \cdot (x^2)^0 \cdot (x^4)^0 \cdot (x^8)^1 \cdot (x^{16})^0 \cdot (x^{32})^1$$

Spesso  $e=3$  oppure  $65537$  (solo due 1 in binario!)

## RSA: Sicurezza

Funzione one way:

- facile da calcolare
- difficile da invertire

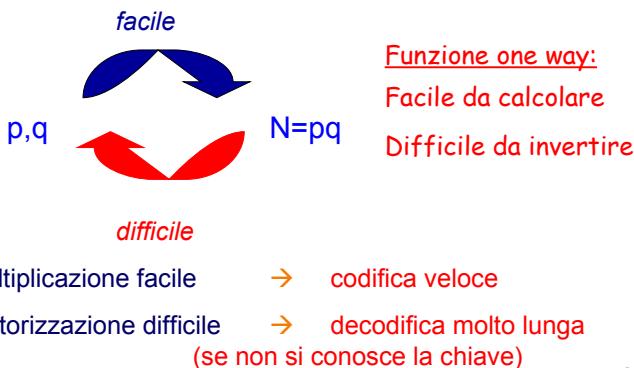


**Facile:** Esiste Algoritmo veloce (tempo polinomiale)

**Difficile:** Non esiste (o si crede che non esista) un algoritmo polinomiale

## RSA: Sicurezza (cont.)

Moltiplicare e fattorizzare:



## RSA: Sicurezza (cont.)

Test primalità : facile

Fattorizzazione: difficile (algoritmi noti non sono polinomiali)

Nota bene: algoritmo polinomiale nella lunghezza dell'input. La rappresentazione di N richiede ( $\log N$ ) bit.

Es.: N da 1024 bit (155 cifre decim.)

$$O(N^{1/2}) = O(2^{512})$$

## RSA: Sicurezza (cont.)

Esempio: Ago. 1999 RSA-155 è stato fattorizzato usando 300 elaboratori (tipo Pentium 2, WS Sun-400 MHz) in 7.4 mesi

- I numeri più difficili da fattorizzare sono quelli  $n = pq$ , dove  $p$  e  $q$  hanno la stessa lunghezza
- Per essere tranquilli
  - 768 bit (230 digit) uso personale
  - 1024 bit per aziende
  - 2048 per chiavi importanti

## RSA: utilizzo in pratica

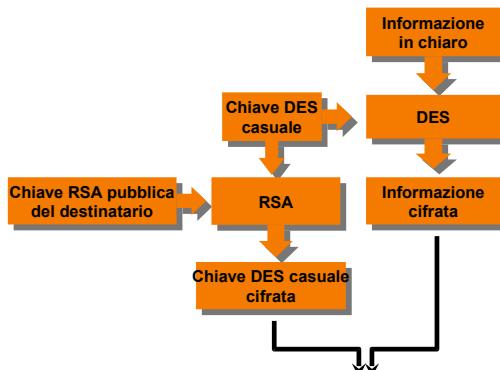
Per molte applicazioni RSA è lento:

si può usare RSA con DES (o con altro metodo a chiave segreta)

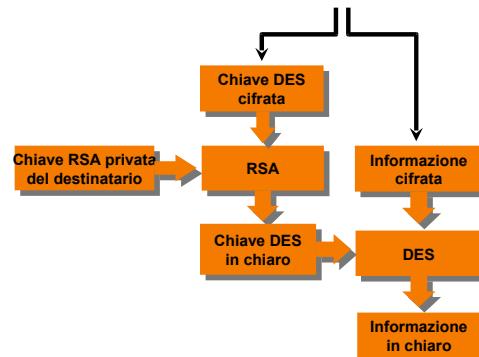
Esempio: A invia un messaggio M a B

- A genera una chiave C, cifra C con RSA e M con DES (con chiave C)
- A invia il documento cifrato con DES e la chiave cifrata con RSA a B
- B usa la sua chiave privata RSA per conoscere la chiave C e poi usa DES per ottenere il messaggio.

## RSA: utilizzo in pratica (cont.)



## RSA: utilizzo in pratica (cont.)



### Chiavi di sessione

L'elevamento a potenza richiesto da RSA richiede molto tempo, mentre DES è almeno 100 volte più veloce ed a livello hardware il divario aumenta. Ciò ha portato, nella prassi comune, ad utilizzare RSA congiuntamente alla crittografia a chiave simmetrica.

Ovvero se Alice vuol inviare a Roberto una grande quantità di dati, per codificarli:

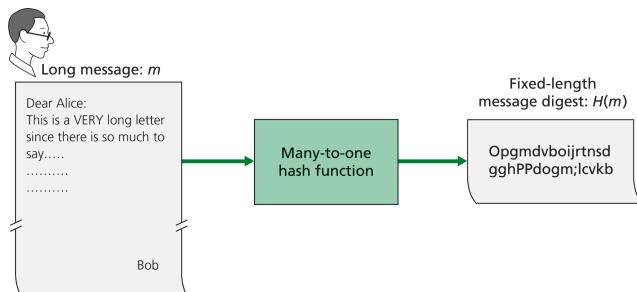
- Sceglie una chiave di sessione  $K_s$
- Codifica il valore della chiave di sessione con la chiave pubblica RSA di Roberto  $c = (k_s)^e \text{ mod } n$ .
- Roberto riceve la chiave di sessione cifrata, la decifra.
- Da ora in poi i dati scambiati saranno cifrati con la chiave di sessione.

### Integrità dei messaggi

Abbiamo appena visto come la cifratura può essere usata per fornire la riservatezza tra due entità comunicanti. Adesso ci concentriamo sull'uso della crittografia per fornire l'integrità ovvero l'autenticazione dei messaggi:

- Verificare che il messaggio sia stato effettivamente creato da Alice
- che non sia stato alterato lungo il cammino verso Roberto

### Funzioni Hash Crittografiche



◆ Hash functions are used to create message digests.

**Def:** Una funzione hash prende in ingresso,  $m$ , e ne calcola una stringa di lunghezza fissata, detta hash.

La checksum di Internet e il CRC soddisfano questa definizione, ma una funzione hash crittografica deve soddisfare anche questa definizione:

**Def:** Deve essere computazionalmente impossibile trovare due messaggi  $x$  e  $y$  diversi, tali che  $H(x) = H(y)$ .

Ovvero che un malintenzionato non riesca a falsificare il contenuto di un messaggio andando a sostituirlo con un altro.

E' facile verificare che la checksum di Internet non rispetta la seconda definizione.

Un algoritmo molto utilizzato per l'hash dei messaggi e' **MD5**, in grado di calcolare un hash da 128 bit con un processo a quattro fasi.

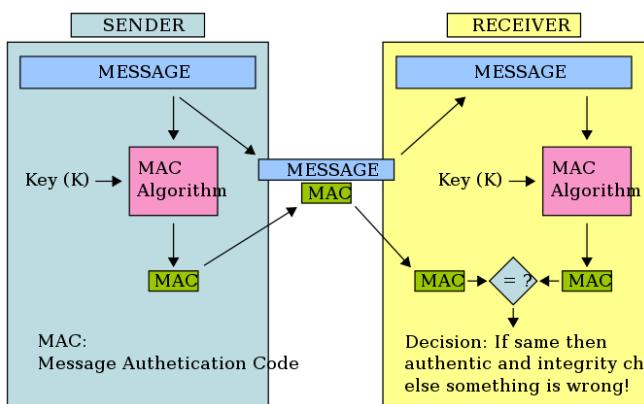
Un altro importante algoritmo hash e' **hash sicuro SHA-1 secure hash algorithm**, basato su principi simili a quelli di MD4. Esso e' uno standard utilizzato nelle applicazioni federali USA. Più grande e' la lunghezza del risultato, più SHA-1 e' sicura.

## Codice di autenticazione dei messaggi

### Primo tentativo:

1. Alice crea un messaggio  $m$  e calcola la stringa hash  $H(m)$  con SHA-1
2. Alice crea un messaggio esteso  $(m, H(m))$  e lo manda a Roberto;
3. Roberto lo riceve  $(m, h)$ , calcola  $H(m)$  e verifica che sia uguale a  $h$

**Difetto:** Tommaso puo' creare un messaggio falso,  $m_1$ , nel quale dice di esser alice , calcola  $H(m_1)$  e manda il tutto a roberto il quale non si accorge di nulla e tutto va a buon fine.



Per realizzare l'integrità dei messaggi oltre alla funzione hash c'e' bisogno di un segreto condiviso. Ovvero una stringa di bit chiamata chiave di autenticazione.

1. Alice crea il messaggio  $m$ , lo concatena con  $s$ , e crea il **codice di autenticazione del messaggio MAC (message authentication code)**  $H(m+s)$  per esempio con SHA-1.
2. Alice aggiunge il MAC al messaggio  $m$  creando il messaggio esteso  $(m, H(m+s))$  e lo manda a Roberto
3. Roberto riceve il messaggio esteso  $(m, h)$  e avendo ricevuto  $m$  e conoscendo  $s$  può calcolare  $H(m+s)$  e confrontarlo con  $h$ .

In crittografia un **Message Authentication Code (MAC)** è un piccolo blocco di dati utilizzato per autenticare un messaggio digitale. Un algoritmo MAC accetta in ingresso una chiave segreta ed un messaggio da autenticare di lunghezza arbitraria, e restituisce un MAC (alle volte chiamato anche *tag*). Il valore MAC protegge sia l'integrità dei dati del messaggio sia la sua autenticità permettendo al destinatario dello stesso (che deve anch'egli possedere la chiave segreta) di rilevare qualsiasi modifica al messaggio: ecco perché dovrebbe essere chiamato *Message Authentication and Integrity Code*, MAIC.

Gli algoritmi di MAC possono essere realizzati anche partendo da altre primitive crittografiche, come le funzioni crittografiche hash(vedi l'[HMAC](#)) od i cifrari a blocchi ([One-key MAC](#), [CBC-MAC](#) e [PMAC](#)).

## Firme Digitali

La firma digitale è basata sulla tecnologia della crittografia a chiave pubblica (o PKI). Dal punto di vista informatico essa rappresenta un sistema di autenticazione di documenti digitali tale da garantire il cosiddetto non ripudio.

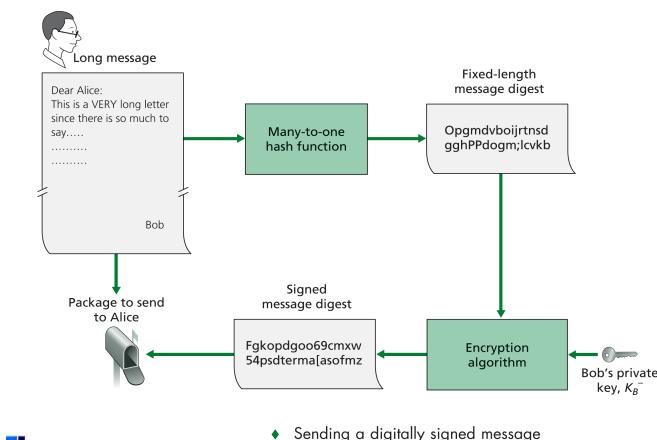
La firma digitale di un documento informatico si propone di soddisfare tre esigenze:

- che il destinatario possa verificare l'identità del mittente (autenticità);
- che il mittente non possa disconoscere un documento da lui firmato (non ripudio);
- che il destinatario non possa inventarsi o modificare un documento firmato da qualcun altro (integrità).

Un tipico schema di firma digitale consiste di tre algoritmi:

- un algoritmo per la generazione della chiave  $G$  che produce una coppia di chiavi ( $PK$ ,  $SK$ ):  $PK$  (Public Key, chiave pubblica) è la chiave pubblica di verifica della firma mentre  $SK$  (Secret Key) è la chiave privata posseduta dal firmatario, utilizzata per firmare il documento.

- un algoritmo di firma S che, presi in input un messaggio m ed una chiave privata SK produce una firma  $\sigma$ .
- un algoritmo di verifica V che, presi in input il messaggio m, la chiave pubblica PK e una firma  $\sigma$ , accetta o rifiuta la firma.

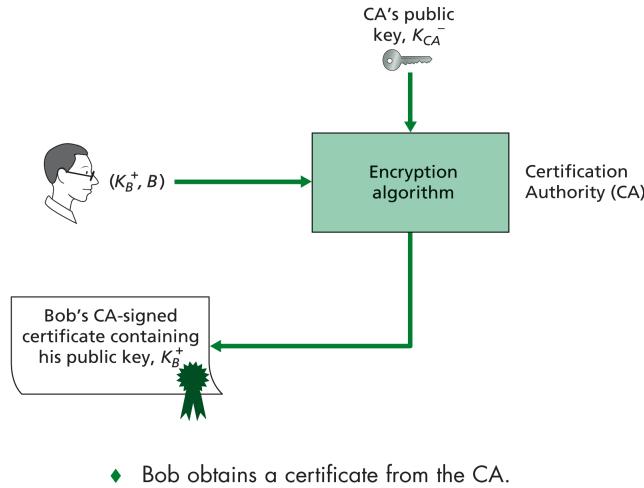


L'utilizzo della crittografia a chiave pubblica per le firme digitali presenta il problema che cifratura e decifratura dei dati sono onerose dal punto di vista computazionale. Un'approccio più efficiente è rappresentato dall'introduzione delle funzioni hash nella firma digitale.

Infatti adesso il mittente invece di firmare l'intero messaggio, si limita a firmarne l'hash  $K\text{-R}(H(m))$  il quale generalmente è più piccolo del messaggio originale, così da andar a diminuire il costo computazionale.

## Certificazione della chiave pubblica

Ovvero la certificazione che una chiave pubblica appartenga a una specifica entità. Generalmente tale relazione è stabilita **dall'autorità di certificazione CA**, il cui compito è quello di validare l'identità ed emettere certificati:



## Autenticazione end-to-end

Adesso ci concentreremo su come un'entità possa esser autenticare un'altra mentre è in atto la comunicazione in rete. Tale autenticazione si basterà solamente sullo scambio di messaggi o di dati come parte di un **protocollo di autenticazione** chiamato **ap (authentication protocol)**.

### ap 3.1

Consente di criptare la password in modo da impedire all'intruso di appropriarsene. Ma purtroppo ciò non basta, siamo ancora soggetti al così detto **attacco di replica**, ovvero Roberto registra la comunicazione per poi riprodurla quando vuole.

### ap 4.0

Nella precedente versione il punto cruciale era che Roberto non poteva accorgersi della differenza tra l'autenticazione originale di Alice e la successiva. Per risolvere il problema possiamo utilizzare la stessa tecnica dell'handshake TCP sostituendo il numero di sequenza con il **nonce** che rappresenta un numero che il protocollo userà una volta nel seguente modo:

1. Alice invia un messaggio, "sono alice" a Roberto
2. il ragazzo sceglie un nonce, R, e lo trasmette a Alice

3. alice utilizza KA-R, la chiave simmetrica segreta che condivide con Roberto per codificare il nonce e gli re invia il valore risultante KA-R(R). in questo caso il nonce viene utilizzato per verificare che alice sia attiva.
4. Roberto decifra il messaggio e controlla che il nonce sia quello da lui inviato precedentemente.

## ap 5.0

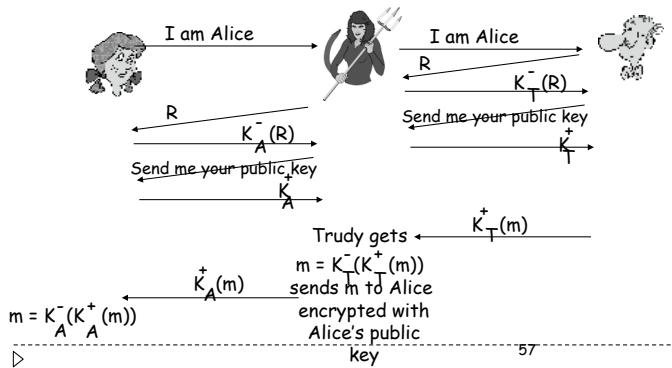
Per risolvere il problema dell'autenticazione in ap 4.0 e stato applicato un nonce e la crittografia a chiave simmetrica. Ma in realtà possiamo utilizzare la crittografia a chiave pubblica e ovviare al problema del primo scambio delle chiavi segrete condivise.

1. Alice invia il messaggio "Sono Alice" a roberto
2. Roberto sceglie un nonce R e lo trasmette alla ragazza per assicurarsi che sia proprio lei
3. la ragazza codifica il nonce con la sua chiave privata e gli ritrasmette il valore risultante KA-(R), il quale può essere generato solo da alice
4. Roberto con la chiave pubblica di alice calcola il nonce e lo confronta con quello da lui scelto precedentemente.

Tale tecnica presenta un **difetto di sicurezza**:

### ap5.0: security hole

Man (woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



### ap5.0: security hole

Man (woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



Difficult to detect:

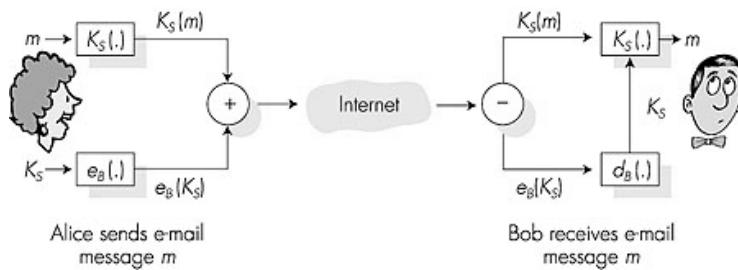
- Bob receives everything that Alice sends, and vice versa. (e.g., so Bob, Alice can meet one week later and recall conversation)
- problem is that Trudy receives all messages as well!!

## E-mail sicure

Le caratteristiche di sicurezza che deve disporre il progetto sono:

- **Riservatezza**, ottenuta mediante una chiave di sessione

- Alice vuole inviare una mail sicura a Bob.



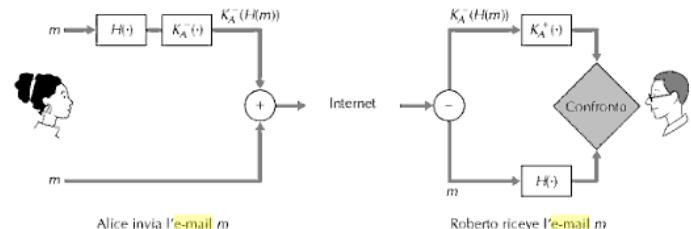
- genera una chiave simmetrica  $K_S$ .
- cifra il messaggio con  $K_S$
- cifra  $K_S$  con la chiave pubblica di Bob.
- invia a Bob  $K_S(m)$  e  $e_B(K_S)$ .

#### • autenticazione del mittente

#### • integrità

Ottenute mediante le firme digitali e la sintesi del messaggio.

1. Alice applica la MD5 al messaggio  $m$ , per ottenerne una sintesi
2. cifra la sintesi con la sua chiave privata, per creare la firma digitale
3. concatena  $m$  in chiaro e lo firma
4. lo invia come unico pacco



Per raggiungere i tre obiettivi non dobbiamo far altro che concatenare le due soluzioni, ovvero prendere il pacco applicargli la chiave di sessione, e concatenare il tutto con la chiave di sessione cifrata con la chiave pubblica del destinatario.

## Rendere sicure le connessioni TCP: SSL

Con SSL, andiamo a fornire riservatezza, integrità dei dati e autenticazione end-to-end a TCP. **Secure Sockets Layer (SSL)** è supportato da tutti i più comuni browser e server web e viene essenzialmente usato da tutti i siti di commercio elettronico in internet.

SSL viene sponsorizzato per fornire la sicurezza delle transazioni che anno luogo su HTTP, tuttavia, dato che SSL rende sicuro TCP, può esser sfruttato da qualsiasi applicazione che gira su TCP, esso non fa altro che fornire una semplice API delle socket, che è analoga a TCP.

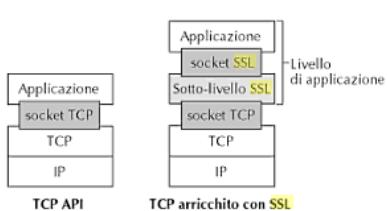


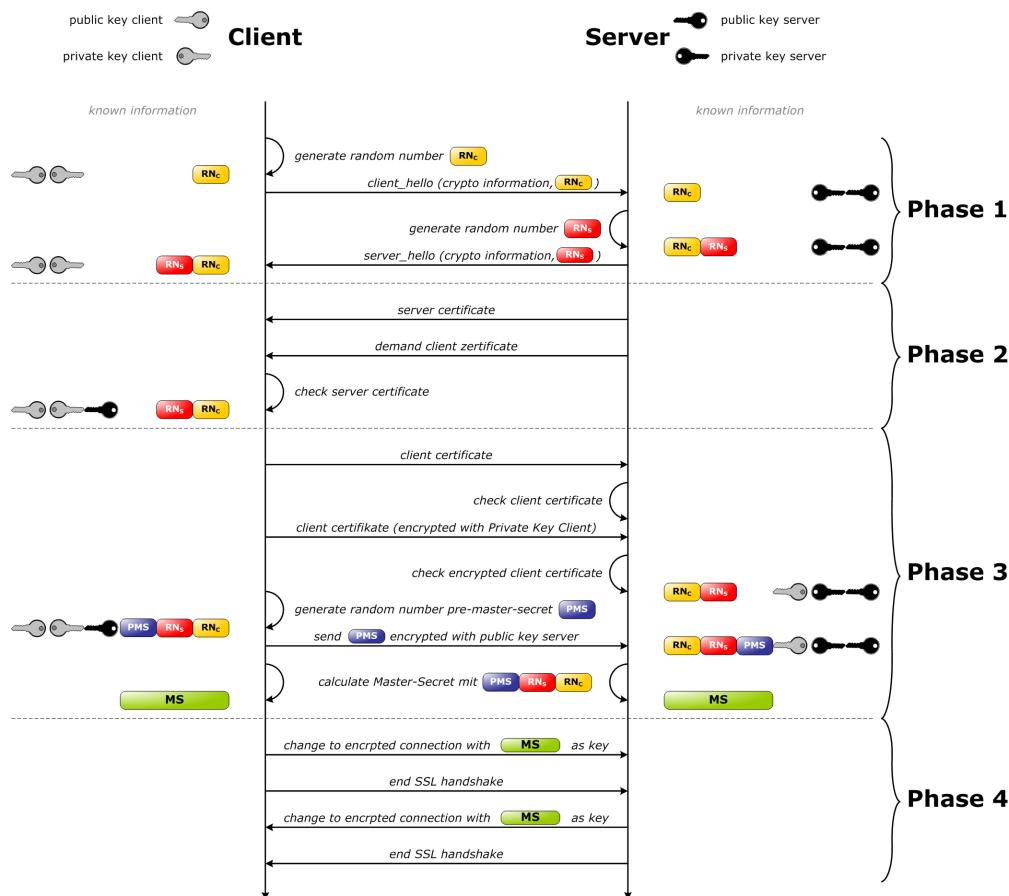
Figura 8.27 Sebbene SSL risieda tecnicamente nel livello applicativo, dal punto di vista dello sviluppatore, è un protocollo a livello di trasporto.

Sebbene SSL risieda tecnicamente nel livello applicativo, dal punto di vista dello sviluppatore è un protocollo di trasporto che fornisce il servizio di TCP arricchito con servizi di sicurezza.

Problemi:

- Se non venisse assicurata la **riservatezza** (cifratura), un'intruso potrebbe intercettare l'ordine d'acquisto e ottenere le informazioni relative alla carta di credito
- Se non venisse assicurata l'**integrità** dei dati, un intruso potrebbe modificare l'ordine.

- Se non venisse usata l'**autenticazione** del server, potrei acquistare su un sito "fasullo".



## Quadro Generale

SSL hanno tre fasi: handshake, derivazione delle chiavi e trasferimento dati.

### 1. handshake

- stabilire una connessione TCP con il Server
- verificare che il Server sia realmente lui
- inviare una chiave segreta principale che verra' utilizzata da entrambi per generare tutte le chiavi simmetriche di cui hanno bisogno per la sessione SSL.

Dopo aver stabilito la connessione TCP, si ha il primo messaggio di hello al quale il server risponde con il proprio certificato che contiene la sua chiave pubblica. Il client genera il valore master segreto MS, che verra' usato solo per questa sessione SSL, lo cifra con la chiave pubblica del server per creare il valore master segreto cifrato EMS e lo invia al server.

SSL non richiede che le due parti usino uno specifico algoritmo di crittografia, ma consente loro di accordarsi all'inizio della sessione SSL, durante la fase di handshake, su quali algoritmi crittografici useranno. Gli step sono i seguenti:

- il client invia, assieme al proprio nonce, la lista degli algoritmi crittografici da lui supportati.
- Dalla lista, il server sceglie un algoritmo a chiave simmetrica, uno a chiave pubblica e un algoritmo MAC. Restituisce al client le proprie scelte, insieme a un certificato e al nonce del server.
- Il cliente verifica il certificato, estrae la chiave pubblica del server, genera il valore master segreto, cifra MS con la chiave pubblica del server e manda MS cifrato allo stesso
- Usando la stessa funzione di derivazione della chiave, come specificato dallo standard SSL, il client e il server indipendentemente calcolano le chiavi di cifratura e MAC da MS e nonce. D'ora in poi tutti i messaggi inviati tra client e server sono cifrati e autenticati con il MAC
- il client invia un MAC di tutti i messaggi di handshake
- il server manda un MAC di tutti i messaggi di handshake

Ultimi due passi necessari per proteggere da manomissioni l'handshake. Scambiandosi il MAC della concatenazione di tutti i messaggi di handshake, che ha spedito e ricevuto.

## 2. Derivazione delle chiavi

E' generalmente considerato più sicuro che il server e client usino ciascuno chiavi crittografiche differenti, oltre che impiegare chiavi diverse per la cifratura e la verifica dell'integrità.

**Er** = chiave di cifratura di sessione per i dati inviati dal client al server

**Mr** = chiave MAC di sessione per i dati inviati dal client al server

**Ea** = chiave di cifratura di sessione per i dati inviati dal server al client

**Mr** = chiave MAC di sessione per i dati inviati dal server al client

Questo può esser semplicemente fatto suddividendo MS in 4 chiavi.

## 3. Trasferimento dati

SSL, suddivide il flusso dei dati in record cui aggiunge un MAC per verifica dell'intergita' e poi cifra il record piu' MAC. Per creare il MAC, Roberto fa entrare i dati del record assieme alla chiave Mr in una funzione hash. Per cifrare il pacchetto Roberto usa la sua chiave di cifratura di sessione Er, ed infine il pacchetto viene passato ad TCP.

Tale struttura sottopone i pacchetti agli attacchi "man in the middle", il quale semplicemente intercettando i pacchetti andrebbe a modificare/scambiare i numeri dei segmenti.

Per evitare tale attacco, SSL fa sì che il mittente (Roberto,client) mantenga un contatore di numeri di sequenza che inizia da zero e viene incrementato per ciascun record che SSI invia. Roberto include tale numero nel MAC, il quale adesso risulta composto da:

- record
- chiave Mr
- numero di sequenza

## 4. Record SSL



Figura 8.29 Formato del record SSL

## 5. Chiusura della connessione

Il capo tipo del record SSL viene utilizzato anche per questo, ovvero per indicare in modo sicuro ed evitare attacchi indesiderati, la volontà di terminare al connessione. Il tipo come sappiamo viene inviato in chiaro, ma è comunque autenticato dal MAC.