

RETI DI CALCOLATORI I

Riassunto basato sul libro:

Kurose Ross – Reti di Calcolatori. Un approccio Top Down (7a ed.)

CAPITOLO 1 – RETI DI CALCOLATORI E INTERNET

INTERNET

Internet è una rete ad accesso pubblico che connette vari dispositivi in tutto il mondo. Questi sono detti **host** (ospiti) o **end system** (sistemi periferici). Gli end system sono connessi tra loro tramite una **communication link** (rete di collegamenti) e **packet switch** (commutatori di pacchetti). I collegamenti possono essere costituiti da varie tipologie con relative **trasmission rate** (velocità di trasmissione).

Due host collegati alla rete suddividono i dati che si scambiano in **pacchetti**, che trasmettono tramite **router**, posti nel nucleo della rete o tramite **link-layer switch** (commutatori a livello di collegamento) sulle reti di accesso.

La sequenza di collegamenti e commutatori attraversati da un pacchetto per arrivare da un end-system all'altro è detta **path** (percorso).

I sistemi periferici accedono a Internet tramite **Internet Service Provider (ISP)**. Un provider è un insieme di commutatori di pacchetto e di collegamenti, che sotto sottoscrizione di contratto fornisce ai clienti diversi tipi di accesso alla rete (DSL, wireless, dial-up).

Tutti i nodi all'interno rete per comunicare tra loro fanno uso di **protocolli**, ovvero definizioni del formato e l'ordine dei messaggi scambiati. Questi garantiscono interoperabilità, e cioè che un terminale (o un software) possa comunicare efficacemente con un altro terminale senza conoscerne le caratteristiche implementative.

Gli **standard Internet** sono sviluppati dall' Engineering Task Force (IETF) e consistono in specifiche normative di una tecnologia o metodologia applicabile su Internet. Uno standard nasce come **Internet Draft**, pubblicato da IETF, può essere promosso a **Request for Comments (RFC)** cioè un documento che propone uno standard e richiede simultaneamente proposte di miglioramento a tutti gli esperti. Questo, una volta supervisionato ed approvato diventa un vero e proprio standard. Si parla di **standard de facto** per riferirsi ad architetture, protocolli che sono effettivamente di ampia diffusione e di dominio pubblico, ma che non sono state mai normate. Mentre invece sono **standard de iure** quelli ufficializzati da enti come ISO.

STRUTTURA DI INTERNET

L'Accesso ad internet avviene per mezzo di un fornitore di servizi o ISP (Internet Service Provider). Gli ISP sono collegati tra loro secondo una struttura gerarchica (**ISP Locali-> ISP Nazionali**). Gli ISP Nazionali si collegano a fornitori di connettività internazionali (NBP, **Network Backbone Provider**), che sono a loro volta collegati in punti di interscambio detti NAP (**Network Access Point**)

SERVIZI INTERNET DISTRIBUITI

Internet può essere vista come una piattaforma che fornisce servizi alle applicazioni. Queste ultime sono dette applicazioni distribuite perché coinvolgono più sistemi periferici che si scambiano reciprocamente dati. I moduli software sono eseguiti sulle periferiche, e quindi i nodi della rete non hanno a che fare propriamente con l'elaborazione, però necessitano della rete per comunicare con altri sistemi.

L'applicazione deve fornire delle **API (Application Programming Interface)** che specificano come un modulo

software possa servirsi di Internet per recapitare dati ad un altro modulo software residente su un altro terminale collegato alla rete stessa.

SISTEMI PERIFERICI

Sono tutti i dispositivi connessi alla rete che si trovano ai confini della stessa (calcolatori, smartphone, server, ecc.). Questi vengono anche detti host in quanto ospitano (ed eseguono) i programmi applicativi. Talvolta questi vengono suddivisi in **client** che richiedono i servizi e **server** che erogano i servizi.

RETI DI ACCESSO

Le **Access Network** sono reti che connettono fisicamente un sistema al suo **edge router (router di bordo)** cioè il primo router sul percorso verso la destinazione.

Accesso Residenziale:

- un accesso di **tipo DSL** (Digital Subscriber Line) viene fornito da una compagnia telefonica che assume il ruolo di ISP. Questa tecnologia usufruisce della linea telefonica (doppino telefonico in rame) per scambiare dati con una DSLAM (Digital Subscriber Line Access Multiplex) che si trova nella centrale locale. Il model DLS converte i dati digitali in toni ad alta frequenza per poterli trasmettere alla centrale locale sul cavo telefonico. Analogamente tutti i segnali analogici in arrivo al modem vengono riconvertiti in formato digitale.

Vengono utilizzate tecniche di Multiplazione a divisione di frequenza (FDM) per separare il segnale vocale (0-4KHz) dal traffico dati in downstream (50kHz-1MHz) e upstream (4-50kHz).

ADSL: l'accesso è **asimmetrico**, perché la velocità di trasmissione e di ricezione sono diverse.

- Un accesso di **tipo DIAL-UP** consiste in una connessione di tipo non permanente, dove la banda utilizzata è quella fonica a bassa frequenza, che viene occupata tramite una composizione di una numerazione telefonica utilizzando programmi Dialer. (velocità di accesso 56 kbps).
- L'accesso di **tipo FTTH** (fiber to the home) fornisce un collegamento in fibra ottica dalla centrale locale direttamente alle abitazioni. (velocità di accesso potenziale dell'ordine dei gigabit).

Accesso aziendale (e residenziale):

- **Rete locale (LAN, Local Area Network)** è una rete di collegamento che copre un'area limitata (abitazioni, aziende, università) e che utilizza tecnologia Ethernet (LAN) o Wireless (WLAN).

La tecnologia **Ethernet** utilizza un doppino in rame intrecciato per collegare numerosi end-system tra loro, connetterli ad uno switch Ethernet e portarli verso l'ISP tramite un router. L'accesso ha generalmente velocità intorno ai 100 Mbps.

La tecnologia **Wireless (Wi-Fi)** consiste in una comunicazione che non fa uso di cavi e che si oppone quindi a sistemi **wired** (cablati).

Accesso wireless su scala geografica:

- **3G Wireless di terza generazione** consentono accesso wireless a Internet con commutazione a pacchetto anche a decine di chilometri dalla stazione base, a velocità che superano 1Mbps.
- **4G Wireless di quarta generazione** può potenzialmente raggiungere velocità superiori a 10 Mbps.

NUCLEO DELLA RETE

Comutatori di pacchetto

Ciascun flusso di dati che si scambiano le applicazioni distribuite viene diviso in parti piccole detti **pacchetti**. Per raggiungere il destinatario questi pacchetti viaggiano attraverso diversi **comutatori di pacchetto** posti sul path (che possono essere di tecnologia e velocità trasmissiva differente).

Ogni pacchetto è composto da un

- **Header** utilizzato ai fini dell'identificazione e gestione;
- **Payload** che contiene i dati veri e propri.

Store and Forward

La maggior parte dei comutatori utilizza la **trasmissione store-and-forward**. Ciò significa che il commutatore deve ricevere l'intero pacchetto e storarlo in un buffer prima di poter cominciare la trasmissione in uscita.

Trascurando ritardi e tempi di elaborazione, se T è il tempo di trasmissione tra un host e il commutatore (numero bit/velocità mezzo), allora il tempo totale di trasmissione tra due host che vogliono scambiare un pacchetto attraverso un router è pari a $2T$. Il tempo di trasmissione di P pacchetti attraverso un router è ($P+1$) T . Il ritardo totale per una trasmissione di P pacchetti su un percorso di N router è pari a **($P+N$) T** .

Ritardi di accodamento e perdita di pacchetti

Ogni comutatore, per ciascuno dei collegamenti a cui è connesso, mantiene un **buffer di output** (o coda) dove conserva i pacchetti che sta per inviare. Un pacchetto in arrivo che richiede l'invio attraverso un determinato collegamento, ma lo trova occupato dalla trasmissione di un altro, deve accodarsi nel buffer. Questo comporta dei **ritardi di accodamento** che vanno ad incrementare i tempi di trasmissione di un pacchetto. Tali ritardi ovviamente sono variabili e dipendono dal livello di traffico sulla rete.

Dato che la dimensione del buffer è finita può accadere che un pacchetto in arrivo trovi il buffer completamente pieno. Si verificherà una **perdita di pacchetto (packet loss)**.

Un problema di interesse ingegneristico è il **controllo di congestione**: adeguare la velocità di invio pacchetti da parte dell'end-system alla velocità di inoltro del router, in modo da evitare code di bufferizzazione piene.

Tabelle di inoltro e protocolli di instradamento

Un router prende un pacchetto in arrivo da un suo collegamento e lo inoltra su un altro dei suoi collegamenti. Esistono diversi modi, a seconda del tipo di rete, tramite i quali un router può determinare su quale collegamento inoltrare il pacchetto.

In Internet ogni end-system ha un **indirizzo IP**. Ogni pacchetto che percorre la rete contiene nella sua intestazione (**header**) l'indirizzo ip della sua destinazione. Ogni router ha una **tavella di inoltro (forwarding table)** che mette in relazione gli indirizzi di destinazione con i collegamenti in uscita del router stesso. Quando il pacchetto giunge ad un router, quest'ultimo esamina l'indirizzo della destinazione e consulta la

propria tabella per determinare il collegamento uscente appropriato verso cui dirigere il pacchetto in uscita.

Internet implementa parecchi **protocolli di instradamento (routing protocol)** per impostare automaticamente queste tabelle di inoltro.

COMMUTAZIONE A CIRCUITO

Nelle reti a commutazione di pacchetto i messaggi utilizzano le risorse di rete (il canale) solo quando necessario. Nelle reti a **commutazione di circuito**, al contrario, le risorse richieste per la comunicazione tra due end-system sono riservate per l'intera durata della sessione di comunicazione ed inoltre prima che la comunicazione avvenga, la rete deve stabilire una connessione (circuito) **end-to-end** tra mittente e destinatario, con capacità trasmissiva assegnata.

Rispetto ad alla commutazione a pacchetti usata in Internet offre una maggiore garanzia sulla ricezione dei pacchetti, che viaggiano su canali dedicati. Non esiste dunque il problema della congestione.

Lo svantaggio è rappresentato dal fatto che i circuiti dedicati sono inutilizzati durante i **periodi di silenzio** e non possono essere sfruttati da altre connessioni.

Multiplexing

Un circuito all'interno di un unico collegamento è realizzato tramite **multiplexing a divisione di frequenza (FDM)** o **multiplexing a divisione di tempo (TDM)**. Con il primo, lo spettro di frequenza di un collegamento viene suddiviso tra le diverse connessioni. Nello specifico viene dedicata un' **ampiezza di banda (bandwidth)** a ciascuna connessione. Con il secondo invece, il tempo viene suddiviso in frame (intervalli) di durata fissa, 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.

PROBLEMATICHE COMMUTAZIONE DI PACCHETTO

L'obiettivo di qualunque rete, idealmente, sarebbe spostare una quantità di dati tra due host, instantaneamente e senza perdita di dati. Purtroppo l'architettura della rete stessa e i mezzi materiali costitutivi limitano necessariamente il throughput (quantità di bit al secondo) e introducono ritardi e perdite.

1. Ritardo di elaborazione

Il tempo richiesto per esaminare l'header del pacchetto e per determinare dove dirigerlo fa parte del **processing delay**. Questo può includere altri fattori, tra i quali il tempo richiesto per controllare errori a livello di bit nel pacchetto. Dopo l'elaborazione il router dirige il pacchetto verso la coda in uscita.

2. Ritardo di accodamento

Una volta in coda, il pacchetto subisce un **queuing delay** mentre attende la trasmissione sul collegamento. Tale ritardo dipende è variabile e cambia dinamicamente in funzione della dimensione della coda. Può essere caratterizzato staticamente tramite distribuzioni di probabilità e i modelli della teoria delle code.

3. Ritardo di trasmissione

Sia L la lunghezza del pacchetto in bit ed R la velocità di trasmissione del collegamento tra router A e router B; Il **trasmission delay** è dato da L/R e misura il tempo effettivamente impiegato per trasmettere tutti i bit del pacchetto sul collegamento.

4. Ritardo di propagazione

Il **propagation delay** misura il tempo impiegato da un bit immesso sul collegamento per propagarsi fino al router B. Il bit viaggia alla velocità di propagazione del collegamento, che dipende dunque dal mezzo fisico. È dato da d/v , dove d è la distanza e v la velocità di propagazione.

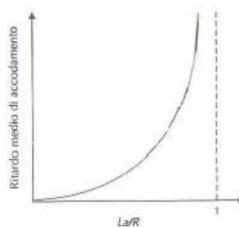
Ritardo di accodamento

La componente più complessa del ritardo totale di nodo è il ritardo di accodamento. Essendo una grandezza variabile in funzione della velocità di arrivo sulla coda, della velocità di trasmissione del collegamento e della natura del traffico (periodico, a raffiche, ecc.) per caratterizzarla si fa uso di misure statistiche quali il ritardo di accodamento medio, la varianza del ritardo di accodamento e la probabilità che il ritardo di accodamento superi un valore fissato.

λ è la velocità media di arrivo dei pacchetti alla coda (packet/s), R la velocità di trasmissione (bit/s). Supponiamo che tutti i pacchetti siano di L bit. La velocità media di arrivo dei bit è λR (bit/s); Il rapporto $\lambda R / R$ è detto **intensità di traffico**.

$\lambda R / R > 1$ allora la velocità media di arrivo dei bit supera la velocità alla quale vengono ritrasmessi in uscita; In questa situazione la coda tenderà a crescere all'infinito e con essa il ritardo di accodamento.

$\lambda R / R \leq 1$ la velocità in uscita è più alta di quella in entrata e sul ritardo di accodamento influisce la natura del traffico. Se l'intensità di traffico è vicina a zero, allora gli arrivi di pacchetti sono pochi e piuttosto distanziati (poco ritardo di accodamento) mentre se è vicino ad 1 il tasso di arrivo dei pacchetti è molto alto e si forma facilmente una coda (ritardo di accodamento aumenta).



Perdita di pacchetti

Nei casi reali i buffer di accodamento nei nodi della rete non sono illimitati, ma hanno capacità finita. Un pacchetto in arrivo dunque può trovare la coda piena (**buffer over flow**). In questi casi il router lo eliminerà e il pacchetto andrà perduto. Possono essere poi applicate diversi protocolli per garantire o meno l'arrivo di tutti i pacchetti (ritrasmessione, ricostruzione, ecc.).

Ritardo End-To-End

I tipi di ritardo finora trattati concernono i ritardi di nodo, ovvero ritardi sul singolo router. Il ritardo end-to-end (punto-punto) invece è il ritardo misurato per inviare un pacchetto dalla sorgente alla destinazione. Su una rete di N router il ritardo end-to-end tra A e B sarà

$$D = (d_{trasm} + d_{prop})_A + \sum_{i=1}^N (d_{elab} + d_{coda} + d_{trasm} + d_{prop})_i$$

Dove i è l'i-esimo nodo di commutazione tra A e B.

Nell'ipotesi che i collegamenti siano tutti non congestionati (d_{coda} trascurabile), che i router abbiano tutti la stessa velocità di elaborazione, i link abbiano tutti la stessa velocità di trasmissione e ritardo di propagazione allora il ritardo approssimato sarà $D = (N+1) (d_{elab} + d_{trasm} + d_{prop})$

Throughput

Il throughput istantaneo in ogni istante di tempo è la velocità in bps alla quale un host di destinazione sta ricevendo un file da una sorgente. Il throughput medio del trasferimento è pari a $F(\text{numero bit})/T(\text{tempo trasferimento})$. Non è da confondersi con la capacità del *link*, infatti mentre quest'ultima esprime la frequenza trasmisiva massima alla quale i dati possono viaggiare, il throughput è indice dell'effettivo utilizzo della capacità del link.

Goodput

È definita come la quantità di dati *utili* trasmessi nell'unità di tempo del throughput, e cioè scartando le informazioni di overhead associate ai protocolli e gli eventuali pacchetti rinviati.

ARCHITETTURA A LIVELLI

Un'architettura a livelli consente di definire singole parti specifiche e ben definite di un sistema articolato e complesso e rende inoltre possibile cambiare l'implementazione del servizio fornito da un determinato livello (mantenendo però l'interfaccia invariata).

Ciascun protocollo di rete è dunque organizzato in **livelli o strati** che offrono dei **servizi** ai livelli superiori.

Nel caso di sistemi distribuiti, un protocollo di livello n si trova appunto distribuito tra i diversi end-system.

L'insieme dei protocolli dei vari livelli è detto **pila di protocolli (protocol stack)** e per Internet consiste di cinque livelli, anche se il modello di riferimento proposto da OSI ne consta sette.



Livello Applicazione

L'*application layer* è la sede delle applicazioni di rete e include molti protocolli quali HTTP (richiesta e trasferimento dei documenti web), SMTP (trasferimento messaggi posta elettronica) e FTP (trasferimento di file tra due sistemi remoti), DNS (traduzione di nomi di host in indirizzi di rete). È distribuito su più sistemi periferici e ciò significa che un'applicazione di un end-system scambia pacchetti di informazione, tramite il protocollo, con l'applicazione di un altro end-system. E' come se si scambiassero **MESSAGGI** direttamente, anche se il pacchetto passa attraverso tutti i livelli.

Livello di trasporto

Il *transport layer* fornisce un canale logico-affidabile di comunicazione end-to-end per pacchetti chiamati in questo livello **SEGMENTI**. Tra le funzionalità offerte possiamo trovare:

- un servizio che stabilisce una connessione persistente all'host, e la chiude quando non necessaria;
- verifica dell'ordine di consegna ed eventuale riordinamento;
- verifica delle perdite ed eventuale ritrasferimento dei pacchetti;
- controllo di flusso (sincronizzazione delle trasmissioni in caso di velocità di trasmissioni diverse);
- controllo di congestione che riconosce lo stato di congestione e adatta di conseguenza la velocità;
- multiplazione che permette di stabilire diverse connessioni contemporaneamente tra due stessi host (astrazione delle porte).

In Internet i protocolli di trasporto diffusi sono

- **TCP (Transmission Control Protocol)** che fornisce un servizio orientato alla connessione, che include la consegna garantita dei messaggi e il controllo di flusso; inoltre segmenta i messaggi lunghi e fornisce un meccanismo di controllo della congestione. Rende con queste caratteristiche la rete *affidabile* ma non garantisce prestazioni in termini di tempi.
- **UDP (User Datagram Protocol)** fornisce un servizio connectionless (senza connessione), senza affidabilità (non gestisce riordinamento e ritrasmissione), né controllo di flusso e della congestione. In compenso è molto rapido (non c'è latenza) ed efficiente per applicazioni leggere real-time. (es in una trasmissione Voip si richiede un bit-rate alto e qualche perdita di dati è tollerabile).

Livello di Rete

Il *network layer* si occupa di trasmettere logicamente i pacchetti, detti **DATAGRAMMI**, tra due host arbitrari, che in generale non sono direttamente connessi; in sostanza si occupa di *instradamento e indirizzamento* verso la giusta destinazione attraverso il path di rete più appropriato. Tra le funzionalità di questo livello troviamo:

- Inoltro (*forwarding*) ovvero ricevere un pacchetto su una porta, immagazzinarlo e ritrasmetterlo;
- Frammentazione e riassemblaggio dei pacchetti

- Instradamento (*routing*) verso il percorso ideale tramite algoritmi dinamici che contengono informazioni sulle condizioni della rete, le tabelle di instradamento e le priorità del servizio;

Livello di collegamento

Il *link layer* riceve datagrammi dal livello di rete e forma i **FRAME** che vengono passati al livello sottostante. Deve svolgere diverse funzioni:

- In trasmissione raggruppare i bit provenienti dal livello rete in frame e mandarli al livello fisico;
- In ricezione controllare e gestire gli errori di trasmissione
- Controllo di flusso
- Operare una moltiplicazione per l'accesso condiviso dello stesso canale fisico

Questo livello in effetti fa apparire in ricezione al livello fisico un flusso di pacchetti esenti da errori. Esempi di protocolli che fanno parte di questo livello sono Ethernet, Wi-Fi e il protocollo di accesso alla rete DOCSIS. Dato che un pacchetto (datagramma) in genere attraversa diversi collegamenti tra la sorgente e la destinazione, questo potrebbe essere gestito da protocolli differenti lungo il suo tragitto.

Livello fisico

Il physical layer ha il ruolo di trasferire i singoli bit pacchettizzati nel frame da un nodo a quello successivo, occupandosi dunque della conversione adatta al mezzo trasmisivo. Il protocollo dunque è fortemente dipendente dal mezzo. Gli standard (più che protocolli) che fanno parte di questo livello definiscono:

- Le caratteristiche fisiche del mezzo trasmisivo (forma, dimensioni, numero pin)
- Le caratteristiche funzionali (significato dei pin);
- Le caratteristiche elettriche (valori tensione per i livelli logici, la codifica e la durata di ogni bit);
- La codifica del segnale digitale su un mezzo trasmisivo che è analogico.

MODELLO OSI

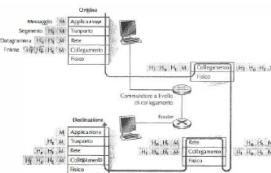
Il modello OSI (Open System Interconnection) proposto da ISO negli anni '70 prevedeva sette livelli ed aggiungeva allo stack di livelli già visti per Internet, il livello di presentazione e quello di sessione.

- Il livello di **presentazione** fornisce servizi che permettono ad applicazioni che vogliono comunicare di interpretare il significato dei dati scambiati. Comprende compressione e cifratura dati.
- Il livello di **sessione** fornisce la delimitazione e la sincronizzazione dello scambio dei dati, compresi i mezzi per costruire uno schema di controllo e di recupero degli stessi.

In internet questi due livelli sono incorporati nel livello applicazione e nel livello di trasporto per ridurre la complessità.

Implementazione livelli

Solitamente i router e i commutatori non implementano tutti i livelli della pila di protocolli ma solo quelli inferiori. I commutatori a livello di collegamento implementano i livelli 1 e 2, i router implementano i livelli da 1 a 3. Gli host invece ragionevolmente implementano tutti e cinque i livelli. Ciò è coerente con l'idea che l'architettura Internet ponga la maggior parte della sua complessità alla periferia della rete.



INCAPSULAMENTO (imbustamento)

Presso un host mittente, un **messaggio** a livello di applicazione M viene passato al livello di trasporto. Questo gli concatena informazioni aggiuntive (informazioni di intestazione di livello) H che saranno utilizzate e scorporate nel livello di trasporto dell'host ricevente. Il messaggio del livello di applicazione concatenato con le informazioni di intestazione del trasporto costituiscono il **segmento** del livello di trasporto, che di fatto incapsula il messaggio del livello di applicazione. Questo meccanismo di passaggio e **arricchimento** si intera per ogni livello, andando così a formare un **datagramma** a livello di rete, un **frame** a livello di collegamento.

Finita la fase di arricchimento ed arrivati al livello fisico, questo si collega con il livello fisico dell'host ricevente che inizia la fase di **impoverimento**, scorporando ad ogni livello l'header relativo e passando il payload al livello superiore.

Dunque a ciascun livello il pacchetto ha due tipi di campi: un **header**, aggiunto dal livello stesso, e un **payload** proveniente dal livello superiore.



CENNI SULLA SICUREZZA

- **Malware**: indica qualsiasi programma usato per disturbare le operazioni svolte dal computer, rubare informazioni sensibili, accedere a sistemi privati, mostrare pubblicità indesiderata. Il modo di propagazione consiste di frammenti di software parassiti che si inseriscono in codice eseguibile già esistente;
- **Botnet**: i malware possono essere anche auto replicanti, nel senso che una volta che hanno infettato un host, cercano riferimenti ad altri host ed usano l'host iniziale per infettarne altri. Si possono diffondere a velocità esponenziali creando vere e proprie reti di Botnet;
- **Virus**: sono malware che richiedono una forma di interazione con l'utente per infettarne il dispositivo;
- **Worm**: sono malware che possono entrare in un dispositivo senza alcuna interazione esplicita con l'utente;
- **Negazione del servizio (DoS)**: è un attacco che rende inutilizzabile dagli utenti legittimi una rete, un host o un'altra parte di infrastruttura. Possono essere di tipo **Bandwidth flooding** (l'attaccante invia un'enorme quantità di pacchetti), **Connection flooding** (l'attaccante stabilisce un gran numero di connessioni TCP), oppure rivolte alle **vulnerabilità dei sistemi**.

- **Analisi del traffico:** è detto packet sniffer un ricevitore passivo che memorizza una copia di ciascun pacchetto che transita su una rete, che sia LAN o WLAN. Il lavoro oneroso non è tanto sniffare, quanto estrapolare o ricostruire i dati a livello applicativo. Una delle migliori difese dunque è la crittografia.
- **Mascheramento:** IP spoofing è la capacità di immettere in Internet pacchetti con un indirizzo sorgente falso "fatti a mano". Per prevenire questo tipo di attacco sono applicati meccanismi di end-point authentication.

CAPITOLO 2 – LIVELLO DI APPLICAZIONE

Il livello applicativo di un applicazione di rete consiste nella codifica dei programmi distribuiti sui sistemi periferici che comunicano tra loro. Lo sviluppatore non deve predisporre un'applicazione per i dispositivi del nucleo della rete, quali router e commutatori, perché questi ultimi presentano solo i livelli inferiori.

Lo sviluppatore prima di procedere alla codifica vera e propria delle applicazioni deve progettare l'**architettura dell'applicazione** e stabilire l'organizzazione sui vari sistemi periferici (es. Client-Server o P2P).

CLIENT SERVER

In un'architettura **Client-Server** gli host di rete possono essere Client, se fruiscono di un determinato servizio dalla rete o Server se forniscono tale servizio. Il server resta in ascolto delle richieste di servizi da parte del cliente. La comunicazione è avviata dunque dal client. Sia client che server dispongono di un indirizzo fisico, detto indirizzo IP che identifica, nella comunicazione, la sorgente e la destinazione.

Quando un client e un server iniziano a comunicare si possono scambiare pacchetti di controllo prima di spedire i dati effettivi (**handshaking**).

Spesso nelle applicazioni Client-Server un singolo host server non è in grado di rispondere a tutte le richieste dei client. Per questo si usano i **data center** che ospitano molti host, che insieme creano un potente server virtuale.

P2P

In un'architettura **P2P** i nodi non sono gerarchizzati sotto forma di client e di server ma sotto forma di nodi paritari (peer) che possono quindi fungere sia da cliente che da servente verso gli altri host della rete. Qualsiasi nodo è in grado di avviare o completare una transazione.

Tra i vantaggi dell'utilizzo di quest'architettura:

- Non è necessario un server con potenza elevata; la capacità di servizio è distribuita;
- Elevata scalabilità, ovvero possibilità di aggiungere capacità di servizio al sistema (più peer);
- La velocità media di trasmissione è più elevata dal momento che l'informazione richiesta può essere reperita da diversi client connessi (peer) anziché un unico server.

PROCESSI COMUNICANTI

Processi di comunicazione in esecuzione sullo stesso sistema utilizzano un approccio inter processo, magari con condivisione di risorse. Per quanto riguarda le applicazioni di rete si tratta invece di comunicare tra processi in esecuzione su diversi host.

E' utilizzata dunque una comunicazione a scambio di messaggi, che si contrappone ad una comunicazione a risorse condivise (**shared nothing**). Il processo mittente crea e invia **messaggi** nella rete e il processo destinatario li riceve, e quando previsto invia messaggi di risposta.

Nel contesto di una sessione di comunicazione tra una coppia di processi quello che avvia la comunicazione è indicato come **client** mentre quello che per iniziare la sessione è detto **server**.

Interfaccia tra il processo e la rete

Ogni messaggi inviato da un processo a un altro deve passare attraverso la rete sottostante. Il passaggio è effettuato tramite un'interfaccia software detta **socket**. Il processo presuppone l'esistenza di un'infrastruttura esterna che trasporterà il messaggio fino al processo dell'altro host (TCP, UDP).

Una **socket** dunque è l'interfaccia tra il livello di applicazione e il livello di trasporto. Si può parlare anche di **API** (application programming interface) tra l'applicazione e la rete.

Indirizzamento

In Internet i processi riceventi e quelli destinatari devono avere un indirizzo per identificare il path di rete che un messaggio deve percorrere ed arrivare a destinazione. Gli host sono identificati tramite **indirizzi IP**, numeri di 32 bit. Oltre a specificare *l'indirizzo dell'host* a cui è destinato il messaggio, il mittente deve identificare anche il *processo destinatario* e più in specifico la **socket** che deve ricevere il dato.

L'applicazione di rete (processo) ricevente è identificata da un **numero di porta**.

Le porte sono suddivise in 3 gruppi:

- **Well Known Ports** (0-1023): sono le porte dei servizi di sistema;
- **Registered Ports** (1024- 49151): sono assegnate da ICANN per usi specifici;
- **User Ports (Dynamic and/or Private Ports)** (49152 – 65525): non sono legate ad usi specifici.

SCEGLIERE IL PROTOCOLLO

Quando si sviluppa un'applicazione occorre scegliere il protocollo a livello di trasporto da usare, scegliendo tra quelli che offrono i servizi più confacenti all'applicazione. I servizi possono essere classificati in grandi linee secondo quattro dimensioni: trasferimento affidabile, throughput, temporizzazione e sicurezza.

Trasferimento dati affidabile

In alcune applicazioni la perdita di informazioni potrebbe causare gravi conseguenze. Quindi occorre garantire che i dati inviati siano consegnati corretti e completi. Se un protocollo fornisce questo tipo di servizio si dice che fornisce un trasferimento affidabile (**reliable data transfer**). Questo requisito può essere tollerato per applicazioni **loss-tolerant**.

Throughput

E' il tasso al quale il processo mittente può inviare i bit al processo ricevente. Dato che altre sessioni condividono la banda, e che queste vengono istituite e rilasciate dinamicamente, il throughput è variabile nel tempo. Alcune applicazioni possono richiedere che sia garantito un throughput minimo disponibile e sono dette sensibili alla banda (**bandwidth-sensitive**); le applicazioni **elastiche** invece possono far uso di tanto o poco throughput a seconda di quanto ce ne sia disponibile.

Temporizzazione

Un protocollo a livello di trasporto può fornire garanzie di timing. Per esempio potrebbe garantire che ogni bit che il mittente invia sulla socket venga ricevuto sulla socket di destinazione entro un tempo massimo.

Sicurezza

Un protocollo a livello di trasporto può fornire inoltre uno o più servizi di sicurezza, tra i quali cifratura di tutti i dati trasmessi e relativa decifratura prima di consegnarli; l'integrità dei dati; l'autenticazione end-to-end.

Servizi che offre il TCP

- Servizio orientato alla connessione: client e server devono scambiarsi informazioni di controllo a livello di trasporto prima che i messaggi a livello di applicazione comincino a fluire. Questa procedura è detta **handshaking**. Dopo questa fase si dice che è stabilita una **connessione TCP** tra le socket di due processi.
- Servizio di trasferimento affidabile: i processi comunicanti si affidano su TCP per trasportare i dati senza errori e nel giusto ordine.
- Servizio di controllo della congestione: permette di limitare la quantità di dati trasmessi, adattando il flusso dati inviato all'eventuale stato di congestione della rete.

Servizi che offre l'UDP

- Senza connessione: non necessita di handshaking ;
- Servizio di trasferimento non affidabile : possibili perdite, errori e ordine non corretto;
- Non esiste un controllo di congestione : un processo può inviare a qualunque velocità;

OSS Né il protocollo TCP né quello UDP hanno alcuna garanzia sui throughput e sulla temporizzazione.

PROTOCOLLI A LIVELLO DI APPLICAZIONE

Un protocollo a livello di applicazione definisce come i processi di un'applicazione, in esecuzione su sistemi periferici diversi, si scambiano i messaggi. In particolare definisce

- I tipi di messaggi scambiati
- La sintassi dei vari tipi di messaggi
- La semantica dei campi
- Le regole per determinare quando e come un processo invia e risponde ai messaggi

Alcuni protocolli vengono specificati negli RFC e sono pertanto di dominio pubblico (HTTP), altri sono privati e volutamente non disponibili (Skype).

WEB

Nei primi anni '90 comparve il World Wide Web ed è uno dei principali servizi di Internet che permette di navigare e usufruire di contenuti amatoriali e professionali. Una delle caratteristiche principali della rete è che i nodi che la compongono sono tra loro collegati tramite i **link** (collegamenti ipertestuali) formando un enorme **ipertesto**.

PROTOCOLLO HTTP

HTTP (hypertext transfer protocol) è un protocollo a livello applicativo usato come principale sistema per la trasmissione d'informazioni sul Web e definito nell'RFC 1945.

Una **pagina web**, detta anche documento è costituita da oggetti, che a loro volta sono file (file HTML, immagine, video, ecc.) indirizzabile tramite URL. Solitamente una pagina web consiste in un file HTML principale e diversi oggetti referenziati (che possono essere anche altre pagine HTML).

Ogni **URL (Uniform Resource Locator)** ha due componenti: il nome dell'host del server che ospita l'oggetto e il percorso dell'oggetto (<http://www.nomeHost.it/pathOggetto>).

Un **browser web** implementa il lato client di HTTP. Un **web server** implementa il lato server di HTTP.

Il protocollo HTTP definisce formalmente in che modo i client web devono richiedere le pagine ai web server e come questi ultimi le trasferiscono ai clienti.

Caratteristiche principali:

- HTTP utilizza come protocollo di trasporto TCP. Quindi il client HTTP per prima cosa inizia una connessione TCP attraverso le proprie socket. Il client invia le richieste e riceve risposte HTTP tramite la propria interfaccia socket, analogamente il server riceve richieste e invia messaggi di risposta attraverso la propria interfaccia di socket.
- HTTP non si deve preoccupare di eventuali dati smarriti nelle richieste, in quanto TCP mette a disposizione un servizio di trasferimento dati affidabile.
- HTTP differisce da altri protocolli di livello 7 (come FTP) per il fatto che le connessioni vengono generalmente chiuse una volta che una richiesta è stata soddisfatta.
- HTTP è classificato come protocollo senza memoria di stato (**stateless**), ciò significa che il server invia i file richiesti al client senza memorizzare alcuna informazione di stato a proposito del client. I browser cercano di garantire una user experience statefull implementando il meccanismo dei **cookies**.

Connessioni persistenti e non persistenti

Quando utilizzato un protocollo TCP per la comunicazione server/client, come nel caso del HTTP, lo sviluppatore può decidere se la coppia richiesta/risposta deve essere inviata sulla stessa connessione TCP (**connessione persistente**) o su connessioni TCP separate (**connessione non persistente**).

Step (non persistente)

1. Il processo client HTTP inizializza una connessione TCP con il server sulla porta 80.
2. Il client HTTP, tramite la socket, invia al server un messaggio di richiesta HTTP che include il path

3. Il processo server HTTP riceve il messaggio di richiesta attraverso la propria socket associata alla connessione, recupera la risorsa chiesta dalla memoria, la incapsula in un messaggio di risposta HTTP che viene inviato al client attraverso la socket
4. il processo server HTTP comunica a TCP di chiudere la connessione al momento della completa ricezione integra del messaggio
5. Il client HTTP riceve il messaggio di risposta e la connessione TCP termina

Per la maggior parte delle applicazioni web si sceglie però di implementare una connessione persistente. Infatti nel caso in cui il client debba richiedere più risorse, nella stessa sessione, allo stesso server, allora conviene non chiudere la connessione TCP aperta e continuare ad inviare richieste su questa.

Round-Trip-Time (RTT) rappresenta il tempo impiegato da un pacchetto di dimensione trascurabile per viaggiare dal client al server e poi tornare al client.

In un protocollo HTTP quindi l'inizializzazione di una connessione TCP comporta un RTT (dovuto all'**handshaking**). L'invio della richiesta HTTP da parte del client e l'inoltro della risorsa da parte del client comportano un altro RTT. Il tempo totale di risposta è di 2RTT+tempo di trasmissione della risorsa.

Il un protocollo HTTP con persistenza dunque il RTT "pagato" per la connessione è sfruttato anche per altri oggetti. I tempi medi di risposta totali si riducono notevolmente.

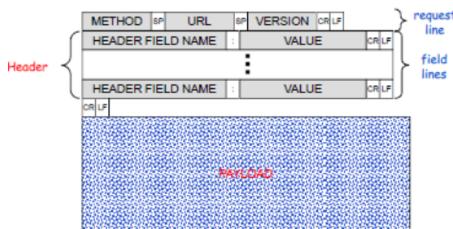
MESSAGGI HTTP

I formati dei messaggi di richiesta e risposta sono definiti negli RFC. Essi sono scritti in formato ASCII su diverse righe, ognuna delle quali è seguita da un carattere di ritorno a capo (carriage return) e un carattere di nuova linea (line feed).

Messaggio di richiesta

Riga di richiesta è composta dal metodo (get, post, head, put, delete, trace, options, connect), l'URI (uniform resource identifier) che indica l'oggetto della richiesta e la versione del protocollo usato.

Gli header possono essere Host, che contiene il nome del server a cui si riferisce l'URL o User-Agent che identifica il tipo di client.



Messaggio di risposta

La riga di stato comprende la versione http e un codice di stato:

- 1xx: messaggi informativi
- 2xx: successfull
- 3xx: redirection
- 4xx: client error (richiesta sbagliata)
- 5xx: server error (problema interno al server)

I più comuni sono 200 (OK), 301 (Moved Permanently), 302 (Found), 400 (Bad Request), 404 (Not Found), 500 (Internal Server Error), 505 (HTTP Version Not Supported).

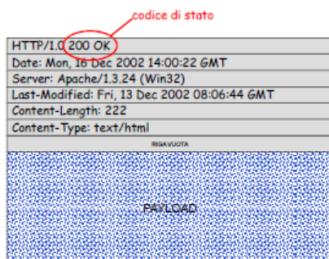
La riga Date indica l'ora e la data di creazione e invio da parte del server della risposta HTTP;

La riga Server indica la tipologia del server dal quale è stato generato il messaggio.

Last-Modified indica l'istante e la data in cui l'oggetto è stato creato o modificato per l'ultima volta. È importante per la gestione dell'oggetto nella cache.

Content-Length contiene il numero di byte dell'oggetto inviato;

Content-Type indica il tipo dell'oggetto inviato.



OSS In un messaggi di risposta è il browser a decidere quali righe di intestazione includere a seconda del tipo e della versione, della configurazione da parte dell'utente e a seconda che possieda in cache una versione dell'oggetto.

Metodi

-GET: server per richiedere una risorsa ad una server (solo header). Può richiedere una risorsa statica o dinamica (tramite query).

-HEAD : E' usata principalmente per scopi di controllo e debugging. Il server risponde soltanto con i metadati associati alla risorsa richiesta (header), senza inviare il corpo della risorsa. Usato per verificare la validità di un URI, l'accessibilità di un URI, la coerenza di cache di un URI.

-POST: Permette di trasmettere delle informazioni dal client al server ad esempio per sottomettere dati ad un forum. Quando le informazioni da inviare sono poche si può utilizzare la GET, passando i dati direttamente sulle URI tramite query.

-PUT: Serve per trasmettere delle informazioni dal client al server, creando o sostituendo la risorsa specificata. Identifica la risorsa che gestirà nel corpo della richiesta.

Header HTTP

Sono righe testuali free-format che specificano caratteristiche generali della trasmissione, dell'entità trasmessa, della richiesta effettuata, della risposta generata.

-Header generali: si applicano sia alla richiesta che alla risposta

- DATE : data e ora della trasmissione
- MIME-Version : Multipurpose Internet Mail Extension è uno standard che estende la definizione del formato dei messaggi (originariamente implementato sulla posta elettronica)
- Connection : Close/Keep Alive (+ tempo di connessione aperta)
- Via : è aggiunta dal proxy ed usata per tracciare un messaggio
- Cache-control : tipo di controllo per la caching
- Expires : data dopo la quale la risorsa non è più valida per la caching

Cookie

I server HTTP sono stateless (senza memoria), scelta che si è resa necessaria per garantire velocità nei salti attraverso i link ipertestuali. I cookie HTTP sono una sorta di gettono identificativo, usato dai server web per poter riconoscere i browser durante comunicazioni con il protocollo HTTP. Tale riconoscimento permette di realizzare meccanismi di autenticazione (login), di memorizzare dati utili alla sessione di navigazione, di associare dati memorizzati dal server, di tracciare la navigazione dell'utente.

Sono composti da una riga di intestazione HTTP, un file mantenuto sul sistema dell'utente e gestito dal browser e un database sul sito.

Nonostante i cookie semplificano molte attività via internet, sono fonte i controversie, in quanto possono essere considerati una violazione della privacy e dell'utente. Inoltre potrebbero essere intercettati tramite diverse tecniche (cross-site scripting, cross-site request) ed usati per ottenere l'accesso (credenziali) al sito web a cui il cookie appartiene.

Web Caching

Un web cache, anche nota come **Proxy Server**, è un'entità di rete che soddisfa richieste HTTP al posto del web server effettivo. Il proxy ha una propria memoria (cache) in cui conserva copie di oggetti recentemente richiesti. Il browser può essere configurato in modo che tutte le richieste HTTP vengano dirette innanzitutto al proxy. Se il proxy dispone della risorsa localmente la fornirà direttamente al cliente, mentre la richiederà al server e poi la inoltrerà (dopo averne salvato una copia locale) al client se non disponibile localmente.

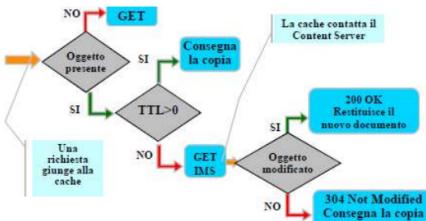
Le motivazioni alla base dell'utilizzo di questo meccanismo sono prima di tutto la riduzione massiccia dei tempi di risposta alle richieste e in secondo luogo la riduzione del traffico.

HTTP fornisce due meccanismi per la **gestione della coerenza** nel caching. Nel TTL (Time To Live) il server quando fornisce un oggetto dice anche quando scade (header Expired), oppure tramite GET condizionali.

Get Condizionale

L'oggetto ospitato nel web server potrebbe essere stato modificato rispetto alla copia nel client (proxy o browser che sia). Il meccanismo di **GET condizionale** permette di affiancare al metodo GET, all'interno di una richiesta HTTP, una riga di intestazione *If-modified-since*.

Il web server invia un messaggio in risposta a un get condizionale che non include l'oggetto richiesto se la riga di stato è 304 Not Modified e questo permette al client di recuperare la risorsa sul proxy.



PROTOCOLLO HTTPS

Il traffico effettuato attraverso il browser con il protocollo HTTP è completamente in chiaro senza alcun genere di sicurezza. Qualunque informazione personale trasmessa tramite questo protocollo può essere intercettata sulla rete tramite un semplice programma di sniffing. Il protocollo HTTPS introduce invece un canale di comunicazione criptato attraverso lo scambio di certificati in modo da garantire l'identità delle parti e la riservatezza dei dati. L'HTTPS si appoggia generalmente sulla porta 443. Sintatticamente è identico all'HTTP, al quale è sovrapposto un livello di SSL (**Secure Sockets Layer**).

PROTOCOLLO FTP

E' il protocollo utilizzato a livello applicativo per trasferire i file (File Transfer Protocol), basato su TCP e definito nel RFC 959.

A differenza di altri protocolli, FTP per trasportare un file utilizza due canali TCP separati che agiscono in parallelo:

- Una **connessione di controllo** usata per trasmettere informazioni di controllo tra client e server come l'identificazione dell'utente, password, ecc.
- Una connessione per il **trasferimento dati**.

Dato che l'FTP usa un connessione di controllo separata, si dice che invia le sue informazioni di controllo **fuori banda (out-of-band)**.

FTP è un protocollo **statefull** perché mantiene lo stato sulle directory e sui dati di autenticazione durante le trasmissioni.

Client e Server FTP

Il protocollo FTP si riferisce ad un modello client/server dove il client da luogo al trasferimento in entrambi i versi e il server resta in attesa delle connessioni (in ascolto sulla **porta 21**).

FTP Attivo

Quando un utente avvia una sessione FTP con un server remoto, il lato client apre due porte con numero random superiore a 1023, una per la trasmissione e ricezione dei dati e l'altra per il controllo.

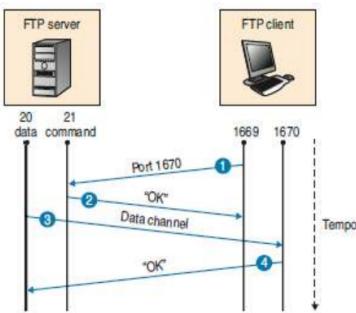
Il client instaura una connessione TCP di controllo con il server sulla porta 21 del server specificando il numero di porta dati del client.

Il client ottiene dal server l'autorizzazione alla connessione sulla linea di controllo.

Il client invia l'identificazione dell'utente e la password sulla connessione di controllo.

Il server apre il data channel verso la porta indicata dal client per iniziare il trasferimento.

Al termine del trasferimento il server chiude la connessione.



Una problematica legata a questa modalità è il fatto che i porti si stabiliscono dinamicamente in fase di controllo, quindi i firewall potrebbero bloccare le connessioni in ingresso sulle porte stabilite.

FTP Passivo

Questa modalità consente al client di aprire sia la connessione di controllo che quella per i dati.

Il client alloca due porte con numero maggiore di 1023, ma non viene indicato al server il numero di porta riservata ai dati: è il server a comunicare al client il numero di porta che lui riserva per il trasferimento dei dati, e quindi non è più la porta numero 21 di default.

La risposta del server comprende anche il suo numero di porta data in modo che il client attivi la connessione anche su di essa, inviando un comando di richiesta di apertura per il canale data.

Comandi FTP

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

CWD <directory> cambia directory corrente

Sia comandi che risposte FTP sono codificate in testo ASCII

PROTOCOLLO SMTP

La posta elettronica, come la posta tradizionale, rappresenta un mezzo di comunicazione asincrono, dato che i destinatari possono leggere i messaggi nel momento per loro più opportuno. Serve dunque un intermediario per consentire la comunicazione asincrona.

L'applicazione prevede tre entità principali:

- gli **user agents** che consentono di leggere, rispondere, inoltrare, salvare e comporre i messaggi;
- i **mail servers** costituiscono la parte centrale dell'infrastruttura.
 - Ciascun destinatario ha una **mail box** contenente i messaggi in entrata non letti;
 - Il mail server contiene una **coda di messaggi in uscita** che contiene i messaggi non ancora recapitati;
- il **protocollo SMTP**.

Le fasi operate per inviare un messaggio da un mittente A ad un destinatario B sono le seguenti.

1. lo user agent di A invia il messaggio al suo mail server, dove è collocato in una coda di messaggi;
2. il mail server di A apre una connessione TCP verso il mail server di B;
3. dopo un handshaking si invia il messaggio;
4. il mail server di B riceve il messaggio e viene posizionato nella mail box del destinatario;
5. quando il destinatario vuole leggere il messaggio, tramite autenticazione può prelevarlo dalla mail box;

OSS: un mail server funge in momenti diversi da client o da server a seconda del ruolo che ricopre nello scambio del messaggio.

Queste fasi descrivono in pratica il protocollo SMTP.

SMTP sta per Simple Mail Transfer Protocol ed è un protocollo definito in RFC821.

Il protocollo SMTP prevede l'utilizzo di TCP (sul porto 25) per un trasferimento dati *affidabile*. La connessione TCP è persistente, nel senso che resta aperta fino a quando il client ha messaggi da scambiare. Presenta un lato *client* in esecuzione sul mail server del mittente e un lato *server* in esecuzione sul mail server del destinatario.

Non prevede mail server intermediari, quindi opera un trasferimento diretto dal server mittente al server destinatario.

Durante l'handshaking

È un protocollo molto vecchio e come tale presenta caratteristiche abbastanza ‘arcaiche’ ad esempio quella di codificare tutto il corpo dei messaggi in ASCII a 7 bit (no caratteri speciali e no multimedia).

Per effettuare un dialogo diretto (senza user agent) con un server SMTP è possibile usare **Telnet**

```
telnet serverName 25
```

Una volta aperta una connessione è possibile usare i comandi HELO, MAIL FROM, RCPT TO, DATA e QUIT per inviare il messaggio; il punto (.) è usato per codificare la fine di un messaggio. Ad ogni comando il server invierà una risposta.

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Motivazioni del SMTP

Si potrebbe pensare ad un protocollo che trasferisca il messaggio direttamente da un user agent ad un user agent. I mail server operano però da intermediario. Se il trasferimento fosse diretto il server che riceve dovrebbe restare sempre in ascolto (quindi acceso) e nel caso di server destinazione non raggiungibile, il client che invia dovrebbe pushare in continuazione finché il messaggio non viene inviato. La funzione da intermediario che ricoprono i due mail server è indispensabile per operare una comunicazione asincrona.

Confronto con HTTP

Durante il trasferimento, sia HTTP che SMTP utilizzano connessioni persistenti. HTTP è però principalmente un **protocollo pull**, nel senso che gli utenti usano HTTP per richiedere risorse (**pull**) dal web server; la connessione TCP viene iniziata da qualcuno che vuole ricevere il file. SMTP è al contrario un **protocollo push**, perché il mail server di *invio* spinge (**push**) il file al mail server in *ricezione*; la connessione TCP viene iniziata dall'host che vuole spedire il file.

Una seconda differenza è che mentre l'STMP deve comporre l'intero messaggio con codifica ASCII a 7 bit, HTTP non impone questo vincolo.

Per quest'ultimo motivo è stata introdotta l'estensione **MIME** (Multipurpose Internet Mail Extensions) con l'RFC 2045. L'estensione prevede righe aggiuntive nell'intestazione che informano di un body MIME, e che quindi può contenere file multimediali, quali video, immagini, audio ed eseguibili.

versione MIME
 metodo utilizzato per codificare i dati
 tipo, sottotipo e parametri del contenuto
 dati codificati

```

From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.....base64 encoded data

--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain

Dear Bob,
Please find a picture of a crepe.
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.....base64 encoded data
--98766789-
  
```

Questa estensione prevede la possibilità di scrivere messaggi *multiparte*, che contengono al loro interno testo con diversa codifica. Viene stabilito un *boundary* (es `boundary=98766789`) che permette di dividere le diverse parti di testo che verranno codificate secondo il *content-type* definito.

PROTOCOLLO POP3

POP3 Post Office Protocol – versione 3 è il protocollo preposto all’accesso alla posta definito in RFC 1939. Entra in azione quando un user agent apre una connessione TCP verso il mail server sulla porta 110.

Quando la connessione è stabilita

1. Durante una fase di autorizzazione lo user agent invia username e password (in chiaro) per autenticare l’utente;
2. Durante una fase di transazione lo user agent recupera i messaggi da leggere e può marcare i messaggi da eliminare;
3. Dopo il comando `quit`, in una fase di aggiornamento, il mail server procede ad eliminare i messaggi marcati dalla mail box.

Il mail server può rispondere ai comandi inviati dall’user con `+OK` o `-ERR`.

- `user` : specifica la username
- `pass` : specifica la password
- `list` : visualizza la lista dei messaggi
- `retr` : preleva il messaggio per numero
- `dele` : elimina il messaggio dal server
- `quit` : chiude la sessione

Uno user agent che usa POP3 può essere configurato in una modalità ‘scarica e cancella’ dove l’utente scarica il messaggio da leggere e il server lo elimina direttamente dalla mail box; ed una modalità ‘scarica e mantieni’ dove il mail server mantiene anche i messaggi letti. Per la prima modalità se un user agent è configurato su più macchine, non è possibile leggere un messaggio già scaricato da una macchina con un’altra macchina.

PROTOCOLLO IMAP

il protocollo IMAP (**I**nteractive **M**ail **A**ccess **P**rotocol) si contrappone al POP3 che permette di scaricare e mantenere i messaggi in locale sull’host. IMAP invece è pensato per consentire direttamente l’accesso ai messaggi memorizzati su un server remoto. Le email sono storate in cache temporanea sui server remoti e mantengono lo stato. Se l’utente legge, modifica, cancella o segna come letto un messaggio, tali modifiche vengono riprodotto sul server e quindi visibili al prossimo accesso.

Con questo protocollo vengono scaricate solo le intestazioni dei messaggi, fin quando non è richiesto il contenuto, così da ottenere una panoramica veloce delle mail.

ACCESSO ALLA POSTA VIA WEB

Grazie a tale servizio, lo user agent è un semplice browser web e l'utente comunica con la propria casella remota via HTTP. Quando l'utente richiederà i servizi di posta elettronica il web browser userà comunque protocolli di mailing SMTP, POP3, IMAP per comunicare con gli altri user agent. Quello che cambia è la comunicazione tra user agent e il primo mail server che è attuata tramite protocollo HTTP e non più SMTP.

DNS

DNS sta per Domain Name System; è un sistema utilizzato per la risoluzione di nomi dei nodi della rete in indirizzi IP e viceversa. Il nome DNS denota, oltre che il servizio, il protocollo che ne regola il funzionamento, i programmi che lo implementano e i server che cooperano per fornire il servizio.

Un IP consiste di 4 byte. Ha una forma del tipo 192.168.1.1 in cui ogni punto separa un numero decimale compreso tra 0 e 255. L'indirizzo IP è gerarchico perché leggendolo da sinistra verso destra otteniamo informazioni sempre più specifiche sulla collocazione dell'host nella rete.

Motivazioni e Servizi

Uno dei servizi principali del DNS è quello di **risoluzione dei nomi**. Attribuendo un nome simbolico ad un server si migliora l'user experience. Oltre che attribuire un nome ad un IP, il DNS fornisce anche il servizio di **Host aliasing** che permette di associare un sinonimo semplice ad un nome complicato. Questo servizio è fornito anche per contattare i **mail server** (che potrebbero avere nomi complicati)

Nel caso in cui si debba sostituire un server che ospita un servizio, o quando è necessario utilizzare nomi diversi per riferirsi ai diversi servizi erogati da uno stesso host, grazie all'utilizzo del DNS il client non deve preoccuparsi in queste situazioni.

Un altro servizio estremamente importante che fornisce il DNS è la **distribuzione del carico di rete (load distribution)**. Con la stessa richiesta di nome, il client può essere reindirizzato verso diversi server replicati. Quando un client effettua una query DNS per un nome associato ad un insieme di indirizzi, il server risponde con l'intero insieme di indirizzi ma ne varia l'ordinamento ad ogni risposta. Dato che il client invia la richiesta HTTP al primo indirizzo, di fatto si distribuisce il traffico sui diversi server.

Implementazione

Il DNS è implementato su diverse macchine DNS server da alcuni software, come ad esempio BIND (Berkeley Internet Name Domain) usato nei sistemi UNIX o DDNS (Dynamic Domain Name System) incluso nei server Windows.

Il DNS utilizza il protocollo di trasporto UDP e la porta 53 per soddisfare le richieste di risoluzione provenienti dagli host. È utilizzato UDP perché il servizio deve essere veloce (TCP non ha garanzie sulla velocità) e l'IP ritornato dal DNS è grande solo 4 byte (il TCP aggiungerebbe molti byte di controllo e di handshaking). Essendo però l'UDP un protocollo non affidabile, in caso di perdita di pacchetti, questi dovranno essere rispediti.

DNS Centralizzato

Un DNS centralizzato prevede un singolo server che contiene un database con le corrispondenze IP/nomi. Seppur di semplice implementazione, un sistema del genere implica una serie di problemi tra i quali:

- un **single point of failure**: se il server si guasta ne soffre l'intera rete;
- **volume di traffico** enorme per un solo server;
- **database centralizzato distante** dai singoli client DNS;

DNS Distribuito e Gerarchico

Per trattare il problema della scalabilità, il DNS utilizza un grande numero di server, organizzati in maniera gerarchica e distribuito nel mondo.

In prima approssimazione esistono tre classi di DNS server:

- **Root Server**: sono i server DNS responsabili del dominio di radice (punto . alla fine dei domini), possiedono l'elenco dei server autoritativi di tutti i domini di primo livello (TLD) conosciuti e lo forniscono in risposta a ciascuna richiesta. I root server sono 13 in tutto il mondo, 7 dei quali collocati negli Stati Uniti; Gli altri sono dislocati sul globo tramite indirizzamento *anycast* (permette di assegnare a più computer lo stesso indirizzo IP);



- **TLD o Top Level Domain**: si occupano dei domini di primo livello quali com, org, net, edu, gov (generici) e di tutti i domini di primo livello relativi ai vari paesi come uk, it, fr, ca (geografici); i server TLD forniscono gli indirizzi IP dei server autoritativi;
- **DNS server autoritativi**: ogni organizzazione dotata di host pubblicamente accessibili tramite Internet (web server e mail server) deve fornire record DNS pubblicamente accessibili che associno i nomi di tali host a indirizzi IP. L'organizzazione può decidere di pagare terzi o di implementare i proprio server autoritativo per ospitare tali record.

Con questo primo approccio se un Client richiede l'IP di www.amazon.com:

1. Il client contatta un root server per avere la lista degli indirizzi IP dei TLD per il dominio .com;
2. Il client contatta uno dei TLD server che gli restituisce l'indirizzo IP del server autorevole per amazon.com;

3. Il client contatta il server autorevole per amazon.com che gli restituisce l'indirizzo IP di [www.amazon.com](http://amazon.com);

DNS Server Locali

In realtà esiste un'altra classe di DNS server che è quella dei **DNS Server Locali**. Questi non appartengono strettamente alla gerarchia dei server, infatti non sono autoritativo per nessun dominio, ma sono centrali per l'architettura DNS. Ciascun ente (università, società, ISP, etc.) ne installa uno o più nel proprio dominio. Tutti gli host nel dominio richiedono a questo server il servizio di risoluzione.

Quando un host effettua una richiesta DNS, la query viene inviata al DNS locale, che opera da proxy e inoltra la richiesta alla gerarchia dei DNS server.

Ciascun host deve essere configurato con l'indirizzo del DNS server locale. Questa configurazione avviene in manuale o in automatico tramite l'utilizzo del DHCP.

Caching dei nomi

Il DNS sfrutta in modo estensivo il caching per migliorare le prestazioni di ritardo e per ridurre il numero di messaggi DNS sulla rete. Il caching si basa sul fatto che a seguito di una richiesta, un DNS server possa mettere in cache le informazioni di risposta. Ogni qual volta è richiesta la stessa risoluzione, questa verrà fornita direttamente. Quindi in questi casi il DNS server fornisce l'IP richiesto anche se non è autoritativo per tale indirizzo. Essendo le associazioni in alcun modo permanenti, i DNS server invalidano la cache dopo un TTL (time to live) in genere di 2 giorni.

Record e messaggi DNS

I server che implementano il database distribuito DNS memorizzano i cosiddetti **record di risorsa(RR)** che hanno il seguente formato (*Nome, Valore, Tipo, TTL*).

Il **TTL** è il tempo di vita del record, dopo il quale viene eliminato dalla cache. Il significato del campo Nome e Valore dipende dal Tipo:

- Tipo A: un record A fornisce la corrispondenza hostname/indirizzo IP
 - nome=hostname;
 - valore = indirizzo IP;
- Tipo NS: usato per instradare le richieste DNS successive alla prima concatenazione delle query;
 - nome=dominio;
 - valore=indirizzo IP del NameServer Autoritativo;
- Tipo CNAME:
 - nome=alias per il nome canonico,
 - valore=nome canonico;
- Tipo MX:
 - nome=dominio di posta;
 - valore = nome dell'host associato al nome;

```

> nslookup
> www.cisco.com
Server:      143.225.229.3
Address:     143.225.229.3#63

Non-authoritative answer:
Name:  www.cisco.com
Address: 198.133.219.26
> ns1.ty=NS
> cisco.com
Server:      143.225.229.3
Address:     143.225.229.3#63

Non-authoritative answer:
cisco.com    nameserver = ns1.cisco.com.
cisco.com    nameserver = ns2.cisco.com.

Authoritative answers can be found from:
ns1.cisco.com  internet address = 128.107.241.186
ns2.cisco.com  internet address = 64.102.266.44
> server ns1.cisco.com
Default server: ns1.cisco.com
Address: 128.107.241.186#63
> ns1.ty=A
> www.cisco.com
Server:      ns1.cisco.com
Address:     128.107.241.186#63

Name:  www.cisco.com
Address: 198.133.219.26

```

Chiedo di risolvere l'host www.cisco.com

Indirizzo del local DNS che serve la richiesta

Ecco la risposta, che non proviene da un server DNS autoritativo

Imposto nslookup per l'invio di query di tipo NS: restituirà i name server autoritativi di un dominio specificato

Chiedo il Name Server autoritativo per il dominio cisco.com

Eccoli: sono due

E questi sono i loro indirizzi IP

Imposto nslookup per l'invio delle successive query al NS ns1.cisco.com

Reimpiego nslookup per l'invio di query di tipo A (risoluzione di nomi di host) e richiedo la risoluzione del nome di host www.cisco.com

Questa volta la risposta è autoritativa. La entry non-autoritativa memorizzata in cache era valida.

Utilizzi impropri

Il meccanismo del DNS può essere facilmente sfruttato per utilizzi impropri. Quando un client inoltra una query DNS ad un server ricorsivo potrebbe ricevere risposte manipolate (e quindi IP diversi). Questa tecnica può essere usata per diversi scopi: protezione di abusi (filtraggio dei siti pericolosi), censura, man-in-the-middle;

TELNET

E' un protocollo di rete definito in RFC 854 solitamente utilizzato per fornire all'utente sessioni di login remoto di tipo riga di comando tra host su Internet. E' un protocollo client- server basato su TCP. I client si connettono alla porta 23 sul server. A causa della sua progettazione è possibile utilizzare un programma Telnet per stabilire una connessione interattiva ad un qualche altro servizio su un server, ad esempio collegandosi alla porta 25 si può effettuare un debugging di un server di posta.

SSH Secure Shell

E' un protocollo che permette di stabilire una sessione remota cifrata tramite interfaccia a riga di comando con un altro host di una rete. Ha sostituito l'analogo, ma insicuro, Telnet. La porta 22 TCP e UDP è assegnata al servizio SSH.

SNMP Simple Network Management

E' un protocollo di rete senza connessione che utilizza come protocollo del livello di trasporto UDP sulle porte 161 e 162, e consente di semplificare configurazione, gestione e supervisione di apparati collegati in una rete, riguardo a tutti quegli aspetti che richiedono azioni di tipo amministrativo.

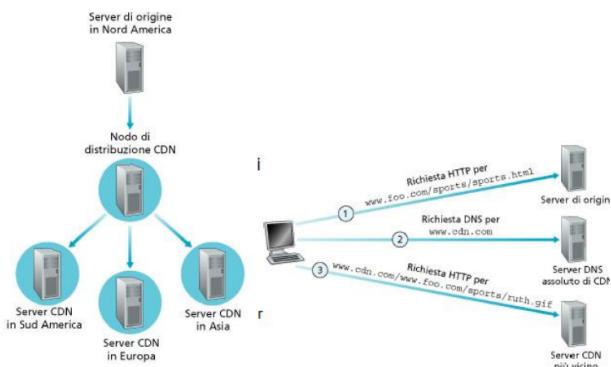
IRC Internet Relay Chat

E' un protocollo di messaggistica istantanea su Internet che consente sia la comunicazione diretta fra due utenti, che il dialogo contemporaneo di gruppi di persone raggruppati in stanze di discussione dette canali. Utilizza il protocollo di trasmissione TCP. Diversi server IRC sono in grado di connettersi tra di loro, formando una rete di comunicazione alla quale gli utenti accedono mediante un client specificando un *nickname*.

CDN Content Delivery Network

Nella distribuzione dei contenuti vengono introdotte le problematiche del delay e del throughput dato che le singole risorse devono essere trasferite da un singolo provider verso una molteplicità di utenti.

La soluzione è rappresentata dal Content Delivery Network, un sistema di server collegati in rete che collaborano per distribuire contenuti agli utenti finali. La tendenza è quella di replicare il contenuto su quanti più nodi possibili e soprattutto vicino agli utenti finali.



L'utente quindi richiede la risorsa al server CDN installato nell'ISP di provenienza, e non il server originario. L'user experience risulta essere migliore se il contenuto è replicato su molti server, dato che l'accesso avviene su server geograficamente più vicini.

La CDN crea una **mappa** che indica la distanza tra i vari ISP e i nodi CDN. Quando arriva una query al DNS autoritativo si determina l'ISP che ha originato la query, si usa la mappa per la scelta del server CDN più vicino e ruta la richiesta a quest'ultimo.

Le CDN solitamente adottano due politiche di dislocazione dei server:

- **Enter deep:** filosofia introdotta da Akamai che prevede di entrare in profondità nelle reti di accesso degli ISP installando gruppi di server, detti anche cluster. L'obiettivo è quello di essere vicini agli utenti finali in modo da limitare il ritardo percepito e il throughput;
- **Bring home:** filosofia di Limelight che prevede di portarsi in casa l'ISP costruendo grandi cluster in pochi punti chiave e interconnetterli usando una rete privata ad alata velocità; Invece di entrare negli ISP pongono un cluster vicino ai PoP di molti ISP di livello 1;

ARCHITETTURA P2P

È un modello di architettura logica in cui i nodi non sono gerarchizzati sotto forma di client e di server, ma sotto forma di nodi equivalenti o **paritari (peer)**. Qualsiasi nodo dunque è in grado di avviare o completare una transazione. Due host della rete sono connessi in modo intermittente e comunicano *direttamente* tra di loro.

Tempo distribuzione Client-Server

u_s banda di upload del collegamento del server;

u_i banda di upload del collegamento dell'i-esimo peer

d_i banda di download del collegamento dell'i-esimo peer

F è la dimensione del file (in bit) e N il numero di peer che vuole scaricare la copia.

Il tempo di distribuzione è il tempo richiesto perché tutti gli N peer ottengano una copia, sotto l'ipotesi che la rete abbia banda elevata (ciò implica che i colli di bottiglia siano le reti di accesso).

Il server trasmette NF bit (N copie di F bit). Il tempo richiesto per inviare le copie è **NF/u_s**,

Il client i impiega un tempo pari a **F/d_i** per scaricare una copia

Il tempo di distribuzione sarà pari al **max {NF/u_s, F/min(d_i)}**

Con questa architettura il tempo cresce linearmente con il crescere dei client N che scaricano il file.

Tempo di distribuzione P2P

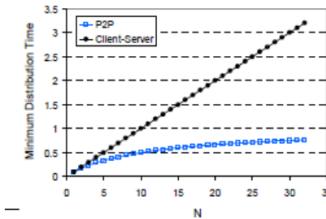
In questa architettura ciascun client che abbia ricevuto alcuni dati del file concorre nella distribuzione del file stesso con la propria banda di upload.

Il server deve inviare almeno una copia ad un peer. Tempo richiesto **F/u_s**

Il client i impiega un tempo pari a **F/d_i** per il download

NF bit totali devono essere scaricati. Velocità massima di upload $u_s + \sum u_i$

Il tempo di distribuzione sarà pari al $\max \{F/u_s, F/\min(d_i), NF/(u_s + \sum u_i)\}$



Con il crescere dei peer N che scaricano il file, il tempo di distribuzione diminuisce. È stato dimostrato quindi che l'architettura P2P (al contrario di Client-Server) può essere scalabile e questo è la diretta conseguenza del fatto che i peer re-distribuiscono i bit oltre che a scaricarli.

FILE SHARING : BITTORRENT

BitTorrent è un protocollo P2P per la distribuzione di file. L'insieme di tutti i peer che partecipano alla distribuzione di un file è chiamata **torrent**. I peer in un torrent si scambiano porzioni dello stesso file di uguale dimensione (256Kb) chiamate **chunk**.

Quando un peer entra a far parte di un torrent inizialmente non ha chunk del file. Man mano che inizia ad accumularli inizia anche ad inviarli agli altri peer del torrent.

Ciascun torrent ha un nodo di infrastruttura chiamato **tracker**. Quando il peer si aggiunge ad un torrent esso si registra presso il tracker per avere la lista dei peer e si connette ad un sottoinsieme di tali peer (**neighbors**) direttamente con connessioni TCP.

Periodicamente, un peer chiede a tutti i neighbor la lista dei chunk in loro possesso ed invia richieste per i chunk mancanti.

Prelievo Chunk

Nel decidere quali chunk richiedere si adotta la tecnica **rarest first**, cioè si determinano i chunk più rari tra quelli che mancano (con il minor numero di copie tra i neighbors). In questo modo i chunk più rari vengono distribuiti più velocemente.

Invio Chunk

Per determinare a quali richieste di chunk il peer debba rispondere si utilizza un algoritmo di **tit-for-tat**, cioè si dà priorità ai peer che forniscono dati al rate più alto. In particolare vengono scelti i top 4, chiamati **unchocked** e che sono ricalcolati ogni 10 secondi. Ogni 30 secondi si seleziona un peer nuovo in maniera casuale (**optimistically unchoke**) e si inizia ad inviargli chunk. Il peer scelto può essere aggiunto ai top 4 se più veloce di uno dei 4.

Questo meccanismo limita la pratica del free-riding, ovvero di scaricare senza contribuire al torrent, in quanto incentiva scambi tra peer con velocità compatibili.

P2P con directory centralizzata (Napster)

Il peer si connette inizialmente ad un server centralizzato fornendo il proprio indirizzo IP e il nome degli oggetti resi disponibili per la condivisione. In questo modo il server raccoglie le info sui peer attivi e le aggiorna dinamicamente.

Gli svantaggi di questo tipo di architettura sono: la presenza di un singolo punto di fallimento, un collo di bottiglia per le prestazioni, e soprattutto il server potrebbe contenere informazioni che violano il copyright.

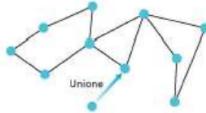
Il trasferimento dei file è di fatto decentralizzato, ma la locazione dei contenuti è pesantemente centralizzata.



P2P Decentralizzato (Gnutella)

Questa architettura è completamente decentralizzata (non esistono server). Si realizza con un'architettura di rete sovrapposta (**overlay network**) fatta da connessioni TCP in corso. L'overlay network ha una struttura paritetica dove ogni peer è connesso al massimo con 10 altri peer. Le problematiche da risolvere sono come costruire e gestire la rete di peer e come localizzare i contenuti all'interno della rete.

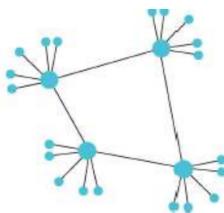
Un peer A inizialmente si connette ad un solo peer B della rete e con la tecnica del **flooding** (inondazione) propaga le query di richiesta (**messaggi di PING**) in uno scope con raggio limitato. Ciòè la richiesta si propaga ai peer collegati al peer B ma solo fino ad un certo punto (es 7 peer a partire dal primo). Il peer A quindi riceve tutti gli indirizzi (**messaggi di PONG**) IP intercettati nella rete con il flooding e a questo punto può connettersi direttamente tramite TCP.



P2P con directory decentralizzata

Sfrutta le caratteristiche positive dell'architettura centralizzata di Napster e quelle dell'architettura decentralizzata di Gnutella. Ogni peer è associato ad un gruppo leader (mini hub) che è esso stesso un peer. Un group leader memorizza le informazioni in condivisione dei "figli". Ogni group leader è in grado di interrogare altri group leader. Quindi il meccanismo delle query flooding è applicato alla rete dei group leader.

In una fase di bootstrap un peer che si connette deve essere associato ad un group leader o deve essere designato esso stesso group leader. L'overlay è costituita da connessioni TCP tra peer e group leader e tra coppie di group leader. Ogni file possiede un identificatore hash e un descrittore. I peer spediscono le query al proprio group leader; quest'ultimo risponde per ogni richiesta con l'indirizzo IP del detentore della risorsa, l'hash, e dei metadati associati alla risorsa. Il group leader inoltra sia le richieste sia le eventuali risposte da parte di altri group leader. Il peer seleziona la risorsa file per il download e invia una richiesta HTTP al detentore della risorsa usando l'hash come identificatore.



SKYPE : P2P VoIP

Skype è un'applicazione P2P VoIP sviluppata da KaZaa nel 2003 che supporta inoltre instant messaging e conferencing. È sviluppata con un protocollo completamente proprietario per cui le info conosciute sono attribuite unicamente a studi di reverse engineering.

Skype utilizza un rete di overlay con tre tipi di host:

- Host ordinari, Skype users
- Super nodi, Skype users con sufficiente hardware
- Server di login per l'autenticazione

Ciascun client Skype mantiene una lista di indirizzi IP di super nodi conosciuti. I super nodi sono responsabili della localizzazione degli utenti, del routing delle chiamate, del mantenimento delle informazioni circa gli host connessi alla rete Skype.

Alla prima esecuzione dopo l'installazione, un client Skype comunica con il server Skype (skype.com). Durante la comunicazione, la cache dell'host è popolata di 7 indirizzi IP di super nodi da usare per il bootstrap. A questo punto l'host può contattare uno di essi per il join. Selezionato il super nodo per il join, parte la fase di autenticazione con user name e password con il server Skype. L'host viene periodicamente aggiornato con indirizzi IP di nuovi super nodi. Per i login successivi un client sceglie uno degli indirizzi dei super nodi e stabilisce direttamente la connessione.

Inizialmente ciascun client Skype poteva essere un supernodo se rispettava dei requisiti minimi di banda e risorse di sistema. Con l'acquisizione nel 2011 da parte di Microsoft l'architettura decentralizzata è stata smantellata sostituendo i supernodi con dei datacenter hostati dalla stessa Microsoft.

STREAMING VIDEO

Generalmente i contenuti video messi a disposizione su Internet sono di due tipologie: in presa diretta (distribuiti contemporaneamente a più utenti) o di tipo Video-On-Demand (cioè pre-registrati). Nei primi vengono utilizzati protocolli **multicast** per la distribuzione simultanea e nei secondi connessioni **unicast** tra l'utente e la piattaforma che ospita il servizio.

Nello streaming HTTP il video viene memorizzato in un server HTTP come un file ordinario con un url specifico. Quando un utente vuole vedere un video, stabilisce una connessione TCP con il server e invia un messaggio GET per il suo URL. Il server invia il video all'interno del messaggio HTTP di risposta. Sul lato client i dati vengono bufferizzati. Lo svantaggio è che il client ricevono la stessa versione codificata del video, nonostante abbiano disponibile una larghezza di banda che può variare da un caso all'altro.

Per superare il problema è stato sviluppato lo **streaming dinamico adattivo su HTTP (DASH dynamic adaptive streaming over HTTP)**. In DASH i video vengono codificati in diverse versioni, ognuna avendo un bit rate differente e quindi differente livello di qualità. Il client seleziona automaticamente in modo dinamico la versione con bit rate adeguato alla banda disponibile.

CAPITOLO 4 – LIVELLO DI RETE: Piano dei Dati

Data la sua complessità, il livello di rete può essere in due parti interagenti:

- Il **piano dei dati**: racchiude tutte le funzioni che ogni router implementa singolarmente per determinare come un datagramma che arriva a uno dei collegamenti in entrata del router è inoltrato a uno dei collegamenti in uscita del router;
- Il **piano di controllo**: racchiude la logica globale di rete che controlla come i datagrammi sono instradati tra i router su un percorso end-to-end tra sorgente e destinazione.

Ad un livello di astrazione alto, il livello di rete prende dall'host mittente i segmenti del livello di trasporto, li incapsula in un datagramma e li trasmette ad un router vicino; nell' host destinatario estrae i segmenti e li consegna al livello di trasporto.

Le funzioni principali del livello rete sono quella di **forwarding** (inoltro) e quella di **routing** (instradamento).

- Con *inoltro* si fa riferimento all'azione locale del router che associa i pacchetti da un'interfaccia di ingresso ad una di uscita. Per inoltrare i pacchetti i router estraggono i valori (instradamento), da uno o più campi dell'header che utilizzano come indice nella **forwarding table**. Il risultato indica a quale interfaccia di uscita il pacchetto debba essere diretto. È implementato generalmente in hardware perché avviene su scala temporale molto piccola.
- Con *instradamento* si indica invece il processo globale di rete che determina i percorsi dei pacchetti nel loro viaggio dalla sorgente alla destinazione. Coincide con l'operazione di "riempimento" delle forwarding table. È implementato in software perché avviene su scale temporali più grandi.

Il livello di rete mette a disposizione un servizio **best-effort** cioè “col massimo impegno possibile”, per cui non c’è garanzia che i pacchetti vengano ricevuti nell’ordine in cui sono stati inviati, così come non è garantita la loro consegna. Non c’è garanzia sul ritardo end-to-end, né su una larghezza di banda minima. Nonostante ciò il livello rete è best-effort by design, perché si dimostra essere efficiente e soprattutto più veloce se combinato con determinate politiche di controllo implementate ai livelli superiori.

Servizi Datagram e a circuito virtuale

Esistono due tipologie di servizio a livello rete: **Datagram** e a **circuito virtuale**.

Con il servizio datagram, i pacchetti entrano nella rete e seguono ognuno un percorso diverso in base a diversi fattori. Tutti arrivano a destinazione con un ordine di arrivo non definito. Ogni nodo intermedio deve avere un modo per fare routing, ossia ogni pacchetto deve essere identificato con gli indirizzi IP mittente e destinazione.

Il servizio a circuito virtuale utilizza invece un canale virtuale logico attraverso cui viaggiano i pacchetti. La difficoltà iniziale sta nel creare, mediante algoritmi di routing, il canale virtuale. Una volta creato può essere numerato con un id, un **identificatore** di flusso, ed utilizzato per instradare i pacchetti. Il router deve solo verificare l’id ed instradare il pacchetto nel relativo canale.

	Datagram	Circuito Virtuale
Creazione circuito	No	Sì (porta del ritardo iniziale)
Informazione di Stato	Ogni pacchetto deve avere indirizzo mittente e destinario	Solo indirizzo circuito (molto veloce e leggero)
Instradamento	Nessuna informazione – pacchetti non fanno mai la stessa strada	Ogni circuito virtuale deve essere identificato
Effetti di guasti nei router	Nessuno, a parte i pacchetti persi	Il percorso legato ai router guasti crollano – recupero complesso
Controllo della congestione	Complesso – demandato ai livelli superiori	Semplice

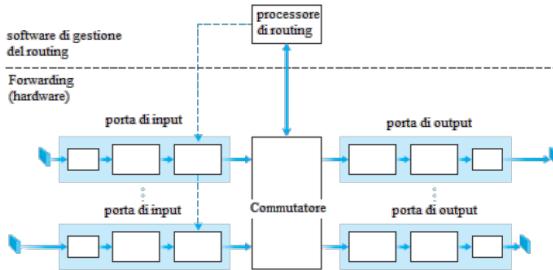
Approccio tradizionale e approccio SDN

In un approccio tradizionale, le funzioni di routing e di forwarding, quindi il piano dei dati e quello di controllo, sono implementate internamente ad ogni router. Il router quindi riempie le proprie tabelle con algoritmi di routing poi sceglie con algoritmi di forwarding il link di uscita per l’inoltro. La gestione di rete è statica e affidata ai singoli operatori che configurano switch e router.

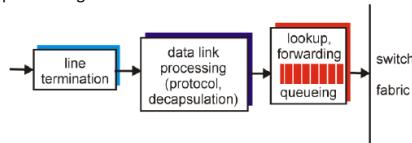
In un approccio alternativo, un controller remoto, separato fisicamente dai router, calcola e distribuisce le tabelle di inoltro a tutti i router, permettendo di fatto la separazione tra il piano dei dati e quello di controllo. Il controller potrebbe essere implementato in un data center. L’architettura **SND Software-Defined-Network** permette questo approccio. L’infrastruttura di rete fisica resta astratta e direttamente programmabile tramite un’interfaccia di controllo software.

ROUTER

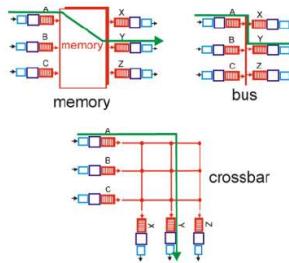
Il router è un dispositivo a livello di rete quindi deputato alla commutazione di livello 3 nello stack OSI. Si possono identificare quattro componenti fisici principali:



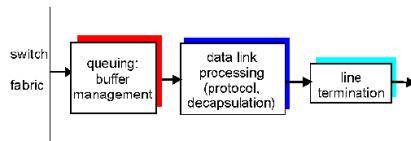
- **Porte di ingresso**: svolgono le funzioni di terminazione (elettrica) di un collegamento in ingresso al router e di elaborazione a livello di collegamento. Queste implementano rispettivamente il livello fisico e di collegamento associati a un singolo collegamento in ingresso al router. Durante l'elaborazione alla porta di ingresso, utilizzando le informazioni della tabella di inoltro, viene determinata la porta di uscita a cui dirigere un pacchetto attraverso la struttura di commutazione. La tabella di inoltro viene elaborata e aggiornata dal processore. Una sua copia conforme è memorizzata su ogni porta di ingresso.



- **Struttura di commutazione (switching fabric)**: connette fisicamente le porte di ingresso a quelle di uscita. Lo switching può essere effettuato in memoria, via bus o attraverso una rete di interconnessione.



- **Porte di uscita**: che memorizzano i pacchetti che provengono dalla struttura di commutazione e li trasmettono sul collegamento in uscita, operando le funzionalità necessarie del livello di collegamento e fisico.



- **Processore di instradamento (routing processor):** esegue le funzioni del piano di controllo, cioè di routing. Nei router tradizionali esegue i protocolli di instradamento, gestisce le tabelle di inoltro e le informazioni sui collegamenti attivi, ed elabora la tabella di inoltro per il router. Nei router SDN, il processore è responsabile della comunicazione con il controller remoto

Le funzioni del **piano dei dati** (quindi routing), quindi svolte da porte di ingresso, porte di uscita e struttura di commutazione sono implementate quasi sempre in hardware perché operano su scala temporale dei nanosecondi. Le funzioni del **piano di controllo** (forwarding) sono invece implementate via software ed eseguite dal processore di instradamento, perché operano sulla scala dei millisecondi o secondi.

STRUTTURA DI COMMUTAZIONE

ACCODAMENTO DEI PACCHETTI

SCHEDULAZIONE DEI PACCHETTI

IP INTERNET PROTOCOL

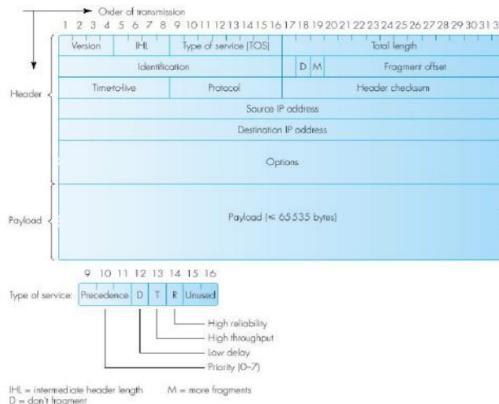
È un protocollo di interconnessione classificato al livello di rete (3) del modello ISO/OSI. È un protocollo a pacchetti senza connessione e di tipo best effort, cioè che non garantisce alcuna forma di affidabilità della comunicazione in termini di controllo di errore, controllo di flusso e di congestione, che viene realizzata al livello superiore (TCP). Attualmente sono in uso due versioni del protocollo, IPv4 e IPv6

Il pacchetto di rete è noto come **datagramma**. È costituito da un header e da un'area dati. I principali campi dei datagrammi IPv4 sono:

- **Numeri di versione [4 bit]:** specifica la versione del protocollo
- **Lunghezza dell'header (IHL) [4 bit]:** indica la lunghezza dell'header del pacchetto IP, ovvero l'offset del campo dati. È necessario perché un datagramma può contenere un numero variabile di opzioni (incluse nell'intestazione);
- **Tipo di servizio [8 bit]:** servono all'host mittente per specificare il modo in cui il destinatario deve trattare il datagramma (precedenza, latenza, throughput, affidabilità).
- **Lunghezza del datagramma [16 bit]:** rappresenta la dimensione in byte del datagramma IP (header + dati). Considerato che questo campo è di 16 bit, la massima dimensione dei datagrammi è di

65535 byte, anche se raramente superano i 1500 byte in modo da non superare la lunghezza massima del campo dei dati dei *frame* Ethernet.

- **Identificatore [16 bit]**: questo campo è un identificativo del datagramma
- **Flag [3 bit]**: il bit DF (Don't Fragment) se settato ad 1 indica che il pacchetto non può essere frammentato; MF (More Fragment) se settato a 0 indica che il pacchetto è l'ultimo frammento;
- **Fragment offset [13 bit]**: indica l'offset di un particolare frammento di un pacchetto IP;
- **Time to live (TTL) [8 bit]**: un contatore di 8 bit incluso per assicurare che i datagrammi non restino in loop per sempre nella rete. Questo campo viene decrementato (o dovrebbe) di un'unità ogni volta che il datagramma è elaborato da un router. Quando raggiunge 0, il datagramma deve essere scartato. (OSS: con 8 bit, un pacchetto può fare al massimo 255 hops; in Internet il routing assicura che un pacchetto arrivi a destinazione al massimo dopo 30-40 hops)
- **Protocollo [8 bit]**: indica il codice associato al protocollo utilizzato nel campo dati del pacchetto IP. Ad esempio il numero 6 indica che i dati sono destinati al protocollo TCP mentre il 17 designa UDP.
- **Checksum dell'intestazione [16 bit]**: consente ai router di rilevare gli errori dell'header. Ad ogni hop il checksum viene ricalcolato e confrontato con il valore di questo campo; se non c'è corrispondenza il datagramma viene scartato. Non viene effettuato nessun controllo sulla presenza di errori del campo Dati, che viene deputato ai livelli superiori.
- **Source/ Destination address [32 bit]**: indirizzi IP che identificano il nodo mittente e quello ricevente.
- **Opzioni**: (multipli di 4 byte) campo che consente di estendere l'intestazione IP con un numero variabile di opzioni che comprendono la security, source routing, router recording, stream identification, timestamping. Se l'opzione non occupa 4 byte, vengono inseriti dei bit di riempimento (zero).
- **Dati (payload)**: contiene il segmento a livello di trasporto (TCP o UDP) da consegnare alla destinazione e in alcuni casi anche altri tipi di dati, quali messaggi ICMP.



Frammentazione dei datagrammi IPv4

La massima quantità di dati che un frame di livello collegamento può trasportare è detta **maximum transmission unit (MTU)**. Non tutti i protocolli a livello di collegamento possono trasportare pacchetti della stessa dimensione a livello di rete (per esempio i frame Ethernet possono trasportare fino a 1500 byte) e sicuramente tra il mittente e il destinatario di un pacchetto verranno utilizzati differenti protocolli di collegamento con differenti MTU.

Il protocollo di rete, e in particolare IP deve frammentare i datagrammi in **frammenti** adatti all'MTU del protocollo di collegamento ricevente e trasferirli in uscita.

I frammenti dovranno essere riassemblati prima di raggiungere il livello di trasporto alla destinazione. Tenendo fede al principio di mantenere semplice il nucleo della rete si prevede di operare il riassemblaggio dei datagrammi sui sistemi periferici, anziché nei router interni.

Quando un host riceve una serie di datagrammi alla stessa origine, deve individuare i frammenti, determinare quando ha ricevuto l'ultimo e stabilire l'ordine di riassemblaggio. I campi di identificazione, flag e offset contenuti nei datagrammi IP assolvono a questo compito.

Original IP Datagram						
Sequence	Identifier	Total Length	DF	May/Don't	MF	Fragment Offset
0	345	5140	0	0	0	0

IP Fragments (Ethernet)						
Sequence	Identifier	Total Length	DF	May/Don't	MF	Fragment Offset
0-0	345	1500	0	1	0	
0-1	345	1500	0	1	105	
0-2	345	1500	0	1	370	
0-3	345	700	0	0	566	

```
C:\Documents and Settings\dis>ping 143.225.229.3 -l 1500
Esecuzione di Ping 143.225.229.3 con 1500 byte di dati:
Risposta da 143.225.229.3: byte=1500 durata=98ms TTL=242
Statistiche Ping per 143.225.229.3:
Pacchetti: Trasmessi = 4, Ricevuti = 4, Persi = 0 (0% persi),
Tempo approssimativo percorsi andata/ritorno in millisecondi:
    Minimo = 98ms, Massimo = 98ms, Medio = 98ms
C:\Documents and Settings\dis>ping 143.225.229.3 -l 1500 -f
Esecuzione di Ping 143.225.229.3 con 1500 byte di dati:
Sarà necessario frammentare il pacchetto ma DF è attivo.
Sarà necessario frammentare il pacchetto ma DF è attivo.
Sarà necessario frammentare il pacchetto ma DF è attivo.
Sarà necessario frammentare il pacchetto ma DF è attivo.
Statistiche Ping per 143.225.229.3:
Pacchetti: Trasmessi = 4, Ricevuti = 0, Persi = 4 (<100% persi)
```

Indirizzamento IPv4

Generalmente un host ha un solo collegamento con la rete e lo utilizza per inviare e ricevere datagrammi. Il confine tra host è collegamento è detto **interfaccia**. Un router invece ha il compito di ricevere datagrammi da un collegamento ed inviarli su un altro, dunque presenta almeno 2 interfacce. Il protocollo IP sia assegnato un indirizzo IP ad ognuna di queste interfacce, e non al singolo host.

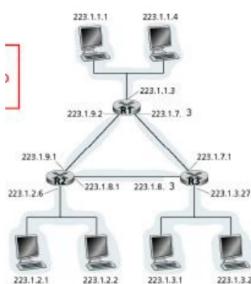
L'**indirizzo IPv4** è formato da 32 bit (4 byte) per tanto esistono circa 4 miliardi di indirizzi possibili. Tali indirizzi sono scritti in **dotted-decimal notation** cioè ciascun byte è indicato in forma decimale ed è separato con un punto dagli altri (es. 192.128.1.1).

Gli indirizzi IP **pubblici** sono univoci e assegnati da **ICANN** Internet Corporation for Assigned Names and Numbers.

Concettualmente l'indirizzo IP si compone di due parte:

- **Identificatore di rete** e precisamente della **sottorete** (Net_ID)
- **Identificatore di host** (Host_ID)

Per determinare le sottoreti di una rete complessa si sganciano le interfacce da host e router in maniera tale da creare isole di reti isolate delimitate dalle interfacce.



Nella rete d'esempio sono presenti 3 router connessi da collegamenti punto a punto. Ciascuno ha 3 interfacce, due per i collegamenti con gli altri router e uno per il collegamento broadcast che connette una coppia di host.

Ogni collegamento tra due reti isolate formano una sottorete. Quindi nell'esempio sono presenti 6 sottoreti.

CLASSFULL ADDRESSING

La strategia di assegnazione degli indirizzi nota come **classfull addressing** prevede che la suddivisione dei bit tra le due componenti di un indirizzo è effettuata in base alla classe di appartenenza. Esistono dunque tre classi, i cui indirizzi hanno rispettivamente 8, 16, 26 bit assegnati alla parte della rete.

La classe di appartenenza è determinata dalla **subnet mask**, che se in notazione /x determina il numero di bit che ricadono nel **prefisso** e che individuano la sottorete. Il restante numero (32-x) costituisce il **suffisso** e identifica l'host specifico all'interno della rete.

La subnet mask però può essere scritta anche in dot notation su 4 byte (X.X.X.X). Se un byte è 1 la porzione corrispondente dell'indirizzo fa parte del prefisso, se è 0 al contrario fa parte del suffisso.

Il **network address** si ottiene poi mediante un AND bit a bit tra l'indirizzo IP e la subnet mask.

<i>Indirizzo IP</i>	192.168.1.1
<i>Subnet mask</i>	255.255.255.0
<i>Subnet Address</i>	192.168.1.0

<i>Indirizzo IP</i>	192.168.5.130	(130 -> 10000010)
<i>Subnet Mask</i>	255.255.255.192	(192-> 11000000)
<i>Subnet Address</i>	192.168.5.128	(10000000 = 128)

Utilizzo Subnet

Al momento dell'invio di un pacchetto, l'host confronta l'indirizzo IP della destinazione con il proprio. Se la destinazione è sulla sua stessa sottorete invia i pacchetti sulla LAN, altrimenti li invia ad un Gateway, che sarà connesso alle altre reti e si occuperà di inoltrare i pacchetti ricevuti.

A seconda della subnet mask utilizzata l'indirizzo ricade in una delle seguenti classi:

CLASSE	UTILIZZO BIT	SUBNET MASK	RETI	HOST
A	0NNNNNNN . HHHHHHHH . HHHHHHHH . HHHHHHHH	255.0.0.0 / 8	128	16M
B	10NNNNNN . NNNNNNNN . HHHHHHHH . HHHHHHHH	255.255.0.0 / 16	16K	65K
C	110NNNNN . NNNNNNNN . NNNNNNNN . HHHHHHHH	255.255.255.0 / 24	2M	256
D	1110XXXX . XXXXXXXX . XXXXXX . XXXXXXXX	MULTICAST		

Classe A:

- Il primo byte rappresenta la rete (va da 0-128), gli altri tre gli host per ogni rete
- La maschera di rete è 255.0.0.0 oppure /8 perché i primi 8 bit sono dedicati alla rete.
- Questi indirizzi iniziano con i bit 0

Classe B:

- I primi due byte rappresentano la rete (da 128-191), gli altri due gli host per ogni rete
- La maschera di rete è 255.255.0.0 oppure /16
- Gli indirizzi iniziano con i bit 10

Classe C:

- I primi tre byte rappresentano la rete (192- 223), l'ultimo gli host per ogni rete
- La maschera di rete è 255.255.255.0 oppure /24)
- Gli indirizzi iniziano con i bit 110

Classe D:

- È riservata agli indirizzi Multicast.
- Non è definita una maschera di rete, essendo tutti i 32 bit utilizzati per indicare un gruppo, non un singolo host.
- Questi indirizzi in binario iniziano con i bit 1110.

I seguenti sono indirizzi speciali *riservati* ad usi specifici.

Network address: indirizzi di qualsiasi classe, il cui suffisso è costituito da tutti '0' e dunque non sono assegnati a nessun host ma identificano una rete specifica. Es. x.0.0.0/8

Direct Broadcast Address: indirizzi che consentono l'invio di un messaggio a tutti gli host sulla stessa sottorete. Il suffisso dell'indirizzo è costituito da tutti '1'. Es x.x.1.1/16

Limited Broadcast Address: 1.1.1.1 (255.255.255.255) viene utilizzato da un host per inviare un messaggio a tutti gli host della stessa rete (il router blocca questi messaggi verso reti esterne).

This Computer Address: 0.0.0.0 utilizzato da un host durante la procedura di avvio del suo stack IP in quanto non è ancora a conoscenza del proprio indirizzo IP; il destinatario di tale pacchetto è un server di bootstrap presente sulla rete

Zero Address: è un indirizzo che contiene tutti 0 nel prefisso e indica un particolare host nella rete locale con i bit del suffisso. Es 0.0.0.46/24

Loopback Address (o Localhost): è un indirizzo ip con il primo byte pari a 127 utilizzati per il test dello stack IP. Individuano la stessa macchina e non abbandonano mai l'host che li ha generati ES 127.0.0.1

L'indirizzamento a classi presenta diversi limiti dovuti soprattutto al numero di host gestibili dalle diverse classi. In pratica se si esauriscono gli indirizzi univoci resi disponibili da una classe, occorre fare ricorso ad un indirizzo di classe superiore.

CLASSLESS INTERDOMAIN ROUTING (CIDR)

A partire dal 1993 si abbandonò il concetto di classful routing in favore di una nuova strategia detta **classless interdomain routing (CIDR)** che generalizza la nozione di indirizzamento di sottorete. L'indirizzo IP è ancora diviso in **prefisso e suffisso** e mantiene la forma decimale puntata a.b.c.d/x. La novità essenziale consiste nell'utilizzare subnet mask di lunghezza arbitraria x, quando l'indirizzamento classfull ammette solo tre lunghezze (/8, /16, /24).

All'interno di una tradizionale Classe C (/24) sono possibili le seguenti sottoreti:

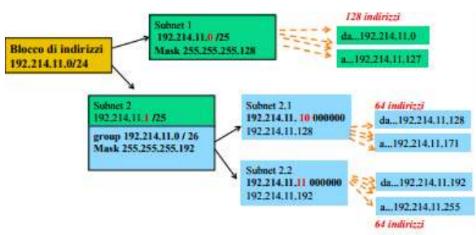
Notazione CIDR	Maschera di Rete	Sottorete disponibile	Hosts di rete disponibili	Totale Hosts utilizzabili
/24	255.255.255.0	1	256	254
/25	255.255.255.128	2	128	126
/26	255.255.255.192	4	64	62
/27	255.255.255.224	8	32	30
/28	255.255.255.240	16	16	14
/29	255.255.255.248	32	8	6
/30	255.255.255.252	64	4	2
/31	255.255.255.254	128	2	2 *

Con il vecchio approccio **classfull** in una sottorete di classe C, 24 bit individuano la singola sottorete, e i restanti 8 bit sono per identificare 256 host. Per cui se sono necessari ad esempio 300 indirizzi una rete di classe C è insufficiente ed una rete di classe B è sovabbondante.

Con l'approccio **classless** possono essere forniti blocchi di indirizzi con 31 netmask diverse. Ciò rende più flessibile la suddivisione di reti in più sottoreti mantenendo un adeguato numero di host in ogni sottorete.

Esempio:

Su un blocco di 256 indirizzi (classe C) è possibile usare una netmask /25 (cioè 255.255.255.128) per ottenere 2 sottoreti da 128 indirizzi. È possibile suddividere la seconda in ulteriori 2 sottoreti da 64 indirizzi con una mask 255.255.255.192 (/26).



INDIRIZZI PRIVATI

Tra gli indirizzi speciali riservati, alcuni blocchi sono riservati per **uso privato**.

- | | |
|---------------------------------|----------|
| • 10.0.0.0 – 10.255.255.255 | CLASSE A |
| • 172.16.0.0 – 172.31.255.255 | CLASSE B |
| • 192.168.0.0 – 192.168.255.255 | CLASSE C |

Sono indirizzi riservati alle reti locali. I pacchetti con tali indirizzi non vengono utilizzati per l'indirizzamento e l'instradamento tramite protocollo IP dai router Internet verso la rete di trasporto. Risolvono dunque il problema della limitatezza degli indirizzi IP pubblici perché possono essere replicati su reti differenti senza avere conflitti. Nel caso occorra connettere ad Internet una rete locale che utilizza indirizzi privati si deve ricorrere al NAT il quale multiplica (o mappa) più indirizzi IP privati su un solo indirizzo IP pubblico, visibile all'esterno della sottorete ed utilizzabile per l'instradamento.

NAT – Network Address Translation

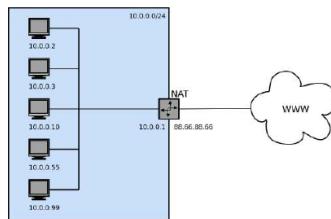
Il NAT è una tecnica che funziona al livello di rete e in particolare è implementato nei router e nei firewall interposti tra le reti private e Internet.

All'interno di una rete privata, sono utilizzati solitamente indirizzi IP privati, che non permettono il routing verso l'esterno. Quando il router che implementa il NAT riceve un pacchetto dalla rete privata, scambia l'indirizzo IP privato (e la porta) con uno pubblico valido per il routing assegnato al router stesso. In pratica un insieme di macchine interne alla rete è vista dall'esterno con un unico ip pubblico, che è quello del router.

Il NAT mantiene una tabella di routing per gestire le connessioni e le diverse comunicazioni fra host. Quando riceve un pacchetto dall'esterno, verifica con la tabella di routing se c'è un host interno che aspetta un pacchetto e lo inoltra.

Il NAT può essere di tipo:

- **Statico:** il NAT associa un indirizzo IP pubblico ad un indirizzo IP privato interno alla rete. Permette di connettere terminali della rete interna ad internet in maniera trasparente ma non risolve il problema della scarsità di indirizzi.
- **Dinamico (Network Address and Port Translation):** il NAT permette di condividere un indirizzo IP routabile fra più terminali in indirizzamento privato. Tutti i terminali della rete interna hanno virtualmente, lo stesso indirizzo IP, se visti dall'esterno. Questa meccanismo può essere infatti chiamato anche **IP masquerading** (maschera IP). Per multiplexare i diversi indirizzi IP su un indirizzo pubblico il NAT dinamico usa il meccanismo di traslazione della porta (PAT Port Address Translation), cioè l'attribuzione di una porta sorgente diversa ad ogni nuova richiesta in maniera tale da poter mantenere una corrispondenza tra le richieste che provengono dalla rete interna e le risposte dei terminali su Internet, tutte indirizzate all'indirizzo IP del router.



PORT FORWARDING

La traslazione di indirizzo, tramite NAT, permette di collegare solo delle richieste che provengono dalla rete interna verso quella esterna, il che significa che è impossibile per un terminale esterno inviare un pacchetto verso un terminale della rete interna.

Esiste un'estensione del NAT detta **port forwarding** che consiste nel configurare in gateway per trasmettere ad un terminale specifico della rete interna, tutti i pacchetti ricevuti su una particolare porta. Così, ad esempio, se si vuole accedere ad un server web (porta 80) funzionante su un terminale 192.168.1.2 (privato) dall'esterno, sarà necessario definire una regola di forwarding della porta sul gateway, ridirigendo tutti i pacchetti TCP ricevuti sulla porta 80 verso il terminale 192.168.1.2.

DHCP - DYNAMIC HOST CONFIGURATION PROTOCOL

DHCP è un protocollo di rete di livello applicativo che permette agli host di una rete locale di ricevere automaticamente ad ogni richiesta di accesso ad una rete, la configurazione IP necessaria per stabilire una connessione e inter operare con tutte le altre sottoreti.

Oltre che all'indirizzo IP da assegnare all'host, DHCP fornisce dinamicamente, al client richiedete:

- Maschera di sottoretore
- Gateway di default
- Indirizzi del server DNS
- Indirizzi del server WINS
- Indirizzi del server NTP
- Altro

DHCP è implementato come protocollo Client-Server. Un **client DHCP** di solito è un host che, connesso alla rete, richiede informazioni sulla configurazione di rete. Di solito ogni sottoretore dispone di un **server DHCP** che assegna gli indirizzi, ma in caso contrario è necessario un **agente di Relay DHCP**, che si occupa di inoltrare le richieste ad un server.

Sessione tipica DHCP:

1. **Individuazione del server DHCP:** il client collegato sulla rete invia un pacchetto UDP chiamato **DHCPDISCOVER** in broadcast sulla porta 67, con indirizzo IP sorgente 0.0.0.0 (this computer address) e destinazione 255.255.255.255 (indirizzo di broadcast);
2. **Offerta del server DHCP:** il pacchetto è ricevuto da tutto il dominio di broadcast e in particolare da tutti i server DHCP presenti, i quali possono rispondere (o meno) ciascuno con un pacchetto di **DHCPOFFER** in cui propongono un indirizzo IP e gli altri parametri di configurazione al client. Questo pacchetto di ritorno è indirizzato in broadcast all'indirizzo di livello datalink del client (che non ha ancora un indirizzo IP) cioè in unicast, per cui può essere inviato solo da un server che si trovi sullo stesso dominio di broadcast. Il pacchetto contiene oltre che a tutte le configurazioni di rete che DHCP offre, anche un **lease time**, cioè il lasso di tempo durante il quale l'indirizzo IP è valido.
3. **Richiesta DHCP:** Il client sceglie tra le offerte dei server dopodiché invia un pacchetto di **DHCPREQUEST** in broadcast, indicando all'interno del pacchetto quale server ha selezionato. Anche questo pacchetto raggiunge tutti i server DHCP presenti sulla rete.
4. **Conferma DHCP:** Il server che è stato selezionato conferma l'assegnazione dell'indirizzo con un pacchetto di **DHCPACK** (nuovamente indirizzato in broadcast all'indirizzo di livello datalink del client); gli altri server vengono automaticamente informati che la loro offerta non è stata scelta dal client, e che sulla sottoretore è presente un altro server DHCP.

Sicurezza DHCP

Il client si identifica verso il server attraverso un campo client-id dei pacchetti DHCP. Questo campo normalmente ha come valore il mac address della scheda di rete per cui si richiede l'indirizzo. Questa è l'unica forma di autenticazione disponibile ed è piuttosto debole, in quanto utilizza un dato che viene inviato in broadcast sulla rete locale, e quindi può essere facilmente trovato da qualunque host connesso alla stessa rete.

Un host malevolo, connesso alla rete, e configurato come server DHCP potrebbe rispondere ad una richiesta DHCP e fornire come gateway di default il proprio mac address. A questo punto l'host malevolo potrà sniffare tutto il traffico generato dal client, e tramite IP masquerading può ridirigere le connessioni verso il gateway ufficiale (MITM). Per prevenire questi attacchi, alcuni switch offrono la funzionalità **DHCP snooping** che permette di fermare i pacchetti che non sono originati da server autorizzati.

PROTOCOLLO ICMP

Il protocollo ICMP (Internet Control Message Protocol) viene usato da host e router per scambiarsi informazioni a livello di rete, tipicamente riguardanti errori, malfunzionamenti o informazioni di controllo.

ICMP è incapsulato direttamente in IP e non è quindi garantita la consegna a destinazione di tali pacchetti (best effort).

ICMP può essere usato per veicolare diversi tipi di messaggi di gestione, identificati dal tipo e dal relativo codice.

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

bit del messaggio:

	1	2	3
0	1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1		
1			
2			
3			
	Tipò Codice Checksum dell'Header ICMP		
	Dati....		

PROTOCOLLO IPv6

Nei primi anni '90 l'IETF diede inizio alla versione successiva del protocollo IPv4. La prima problematica affrontata fu quella dello spazio di indirizzamento. I 4 miliardi di indirizzi possibili con IPv4 stavano per terminare. Inoltre sulla base dell'esperienza accumulata, i progettisti colsero l'occasione per apportare altre migliorie e risolvere diverse problematiche:

- spazio di indirizzamento
- routing
- sicurezza (confidenzialità, integrità, autenticazione)
- configurazione automatica
- servizi di tipo real-time (IPv6 cerca di garantire un servizio efficiente a differenza di IPv4)

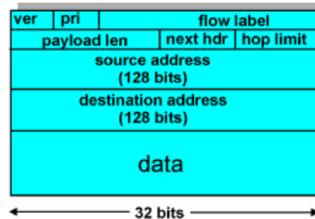
L'indirizzo di rete è lungo 268 bit, cioè 32 cifre esadecimale. Questo porta il numero di indirizzi esprimibili dall'IPv6 a $2^{128} = 16^{32} = 3,4 \times 10^{38}$.

Viene definito un nuovo tipo di indirizzo, **anycast**, che si riferisce ad un insieme di interfacce. Un pacchetto inviato ad un indirizzo anycast, viene recapitato ad una delle interfacce che fanno parte dell'insieme da esso individuato (tipicamente quella più vicina secondo la metrica utilizzata dal protocollo di routing).

L'header è stato semplificato. Alcuni campo sono stati eliminati o resi opzionali. Ciò ha consentito che, malgrado gli indirizzi IPv6 siano 4 volte più lunghi di quelli di IPv4, l'header del primo è soltanto il doppio di quello del secondo.

Viene introdotto il supporto per la *Quality of Service*, cioè una funzionalità che permette di etichettare (**flow label**) i pacchetti appartenenti a flussi di dati particolari per i quali si richiede un trattamento di tipo non default.

L'header consiste di due parti: **header principale** ed **extension headers**. Gli extension headers sono introdotti per ospitare opzioni aggiuntive tra le quali opzioni per la gestione del routing, della frammentazione, dell'autenticazione e della sicurezza.



- **vers**: indica il numero della versione del protocollo IP
- **prio**: livello di priorità del datagramma
- **flow label**: associato alla QoS richiesta
- **payload lenght**: lunghezza del payload
- **next Hdr**: tipo di header che segue quello IPv6 (es. 6 per pacchetti TCP e 16 per UDP)
- **hop limit**: contatore del numero di hops (analogo a TTL)
- **source address**: indirizzo del mittente
- **destination address**: indirizzo del destinatario

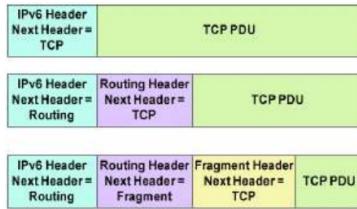
Differenze con l'header IPv4

- **checksum:** rimossa completamente per ridurre il tempo di processamento nei router ad ogni hop;
- **options:** previste, ma non nell'header. È possibile inserirle nel payload utilizzando il next header;
- **ICMPv6:** nuova versione di ICMP.

Le priorità dei pacchetti possono essere:

0	traffico non noto
1	traffico di riempimento (es. news)
2	traffico batch (es. e-mail)
3	riservato
4	traffico interattivo a bassa priorità (es. ftp, NFS)
5	riservato
6	traffico interattivo ad alta priorità (es. telnet, X)
7	traffico di controllo di internet (es. OSPF, SNMP)

Il meccanismo degli header, incapsulati nel payload funziona come segue:



Frammentazione

Un'altra differenza sostanziale con IPv4 è la modalità di **frammentazione** dei pacchetti. In IPv4 la frammentazione avviene all'occorrenza sui router intermedi, quando il pacchetto è più grande del MTU del router. In IPv6 la frammentazione viene effettuata invece in modalità **end-to-end**.

Se un IPv6 datagram ha dimensioni maggiori della MTU, questo viene scomposto (dalla sorgente) in datagrammi più piccoli, detti **frammenti**, ognuno dei quali contiene una parte del payload relativo all'IP datagram originale. Il livello Network prima di inviare il datagramma richiede all'interfaccia di rete di destinazione la sua MTU per poter eventualmente effettuare la frammentazione. L'header non è frammentabile ed è inviato con ciascun frammento.

Questa frammentazione avviene con l'ausilio di un particolare *extension header*, detto **Fragment Header**, identificato dal valore 44 nel campo Next Header. L'header in questione contiene i campi relativi alla frammentazione presenti anche nell'header dell'IPv4 (fragment offset, M, identification).

In questo modo si riduce l'overhead dei router, in modo che essi possano gestire più pacchetti per unità di tempo.

Indirizzi IPv6

Gli indirizzi sono rappresentati come 8 gruppi di 4 cifre esadecimale separati dal carattere :
Se uno dei gruppi è composto da una sequenza di quattro zeri può essere contratto ad un solo zero;
Una sequenza di zeri contigui (e una soltanto) composta da 2 o più gruppi può essere contratta con ::

Di seguito sono riportate varie rappresentazioni dello stesso indirizzo

```
2001:0db8:0000:0000:0000:0000:1428:57ab  
2001:0db8:0000:0000::1428:57ab  
2001:0db8:0::0:0:1428:57ab  
2001:0db8:0::0:1428:57ab  
2001:0db8::1428:57ab
```

Inoltre possono essere omessi gli zeri iniziali di ogni gruppo: 2001:db8:2de::e13

Gli ultimi 32 bit possono essere scritti in decimale, nella notazione dotted decimal, rendendo così la sintassi IPv6 retro compatibile con quella IPv4. Esistono due rappresentazioni per gli IPv4 utilizzando indirizzi IPv6.

IPv4-mapped address

I primi 80 bit sono posti a 0, i successivi 16 sono posti a 1 (ffff) e gli ultimi 32 rappresentano l'IPv4.

```
0:0:0:0:0:ffff:192.168.0.1  
::ffff:192.168.0.1
```

IPv4-compatible address

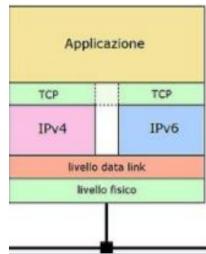
I primi 96 bit sono posti a 0 e gli ultimi 32 rappresentano l'indirizzo IPv4 (deprecati in favore di mapped)

```
0:0:0:0:0:0:192.168.0.1  
::192.168.0.1
```

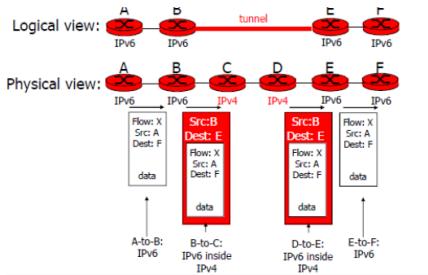
Transizione IPv4/IPv6

Il passaggio nella rete Internet del protocollo di livello rete IP dalla versione 4 alla versione 6 ha portato ingenti problemi di compatibilità. La politica naturalmente adoperabile consisterebbe nel costruire router e switch in grado di interpretare entrambi i protocolli. Questo risulta di difficile applicazione per la vastità dei router da sostituire nel mondo. Si adottano dunque soluzioni software quali *dual stack o tunneling*.

La tecnica del **dual stack** prevede l'utilizzo del doppio stack IP, nella pila protocolare. Questo è di semplice implementazione ma aumenta la complessità della rete, nel senso che i router e gli switch devono interpretare più istanze dello stesso protocollo. Inoltre non risolve il problema dell'indirizzamento, perché ogni interfaccia deve comunque essere dotata di un indirizzo IPv4 e uno IPv6.



La tecnica del **tunneling** è quella più utilizzata. Il principio del tunneling stabilisce un collegamento point to point tra due host. I pacchetti IPv6 vengono incapsulati dall'host sorgente in pacchetti IPv4 (nel payload), inviati nel tunnel, e una volta giunti a destinazione, l'host li decapsula e li tratta come se fossero comuni pacchetti IP.



Indirizzi Speciali

È stato definito un certo numero di indirizzi speciali riservati ad usi particolari. Nella notazione CIDR:

- ::/128 – indirizzo composto da tutti zeri utilizzato per inizializzare l'host prima che esso conosca il proprio indirizzo.
- ::1/128 – indirizzo di **loopback** utilizzato per identificare la macchina locale (localhost) su cui i programmi sono in esecuzione
- ::/96 – utilizzato per interconnettere le due tecnologie IPv4/IPv6 nelle reti ibride (deprecato)
- ::ffff:0:0/96 – indirizzo IPv4 mapped address utilizzato nei dispositivi dual stack
- fe80::/10 – il prefisso link-local specifica che l'indirizzo è valido solo sullo specifico link fisico
- ff00::/8 – indirizzo multicast

PROTOCOLLO ARP

Una volta noto l'indirizzo IP di destinazione, la scheda di rete per indirizzare il frame alla destinazione deve conoscere l'indirizzo fisico della scheda di rete dell'host di destinazione.

La mappatura tra indirizzo fisico MAC e indirizzo IPv4 è fornita dal protocollo **ARP, address resolution protocol**. Il suo analogo in IPv6 è NDP (Neighbor Discovery Protocol).

ARP è un protocollo di servizio, utilizzato in una rete che utilizzi il protocollo di rete IP sopra un livello di datalink che supporti il servizio **broadcast**.

Per inviare un pacchetto IP ad un host della stessa sottorete, è necessario incapsularlo in un pacchetto a livello di datalink, che dovrà avere come indirizzo di destinazione il MAC Address dell'host destinatario. ARP è utilizzato per ottenere questo indirizzo. Se il pacchetto deve essere inviato ad un host di un'altra sottorete, ARP è utilizzato per recuperare il MAC Address del gateway o del router che interfaccia le due reti.

In ogni host il protocollo ARP tiene traccia delle risposte precedentemente ottenute in una ARP Cache per evitare di dover utilizzare continuamente ARP prima di inviare ciascun pacchetto al destinatario (overhead). Le informazioni contenute nella cache vengono cancellate tipicamente dopo 5 minuti.

1. L'host che vuole conoscere il MAC Address di un altro host sulla stessa rete di cui conosce l'IP (ad esempio tramite DNS) invia in *broadcast* una **ARP Request** contenente il proprio indirizzo MAC e l'indirizzo IP del destinatario.
2. Tutti gli host collegati alla rete ricevono la richiesta
3. In ciascun host, il protocollo ARP verifica, confrontando l'IP proprio con quello inviato, se viene richiesta il proprio indirizzo MAC;
4. L'host di destinazione che riconosce il proprio indirizzo IP, provvede ad inviare una **risposta ARP Reply** contenente il proprio MAC direttamente all'host mittente (*in unicast*)

Questo protocollo è applicato in maniera identica quando un host deve recuperare un MAC Address di un host su un'altra rete. Il protocollo in questo caso è utilizzato per recuperare il MAC Address del gateway (il cui indirizzo IP è settato nella configurazione della scheda di rete) che inoltrerà poi la richiesta all'host destinatario.

Formato del pacchetto ARP

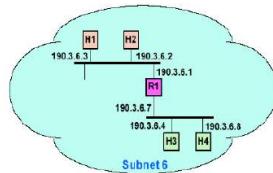
Ogni scambio di pacchetti ARP avviene tramite incapsulamento di questi all'interno di frame di livello datalink (ad esempio frame Ethernet), così come sono incapsulati i pacchetti IP. Il campo *protocol type* nell'header discrimina i pacchetti. Il pacchetto ARP è costituito dai seguenti campi:

- **Hardware type:** specifica il tipo di interfaccia hw su cui l'utente cerca una risposta, per Ethernet si setta il campo 1;
- **Protocol type:** indica il tipo di indirizzo ad alto livello che il mittente ha fornito, per IP si setta a 0x0800
- **Hardware len e Protocol len:** consentono di usare ARP su reti arbitrarie perché specificano la lunghezza dell'indirizzo hardware (MAX) e dell'indirizzo del protocollo di alto livello (IP);
- **ARP operation:** ARP Request (valore 1), ARP Reply (valore 2), RARP Request (3), RARP Reply (4).

0	8	16	31
Hardware Type		Protocol Type	
Hardware length	Protocol length	Operation Richiesta:1, Risposta:2	
Source hardware address			
Source protocol address			
Destination hardware address (vuoto nelle richieste)			
Destination protocol address			

Proxy ARP

È una tecnica di utilizzo del protocollo ARP per fornire un meccanismo *ad hoc* di routing, che non richiede la configurazione dell'indirizzo IP del default gateway sugli host. Un router che implementa il Proxy Arp risponde alle richieste ARP per tutti gli host che appartengono a reti che sa come raggiungere, fingendo che il proprio indirizzo MAC corrisponda a quello degli host di ciascun indirizzo IP richiesto. In questo modo gli host non avranno bisogno di configurare un router di default.



ARP Spoofing

La costruzione di un pacchetto ARP ingannevole è semplice ed infatti questa è una tra le maggiori vulnerabilità delle reti locali. Inviano ad un host un ARP REPLY opportunamente contraffatto possiamo modificare la sua cache ARP, ottenendo ad esempio la possibilità di intercettare dati destinati ad altri host (MITM). Questa tecnica è detta ARP Spoofing o ARP Cache Poisoning.

PROTOCOLLO RARP

Il Reverse Resolution Address Protocol è un protocollo usato per risalire all'indirizzo IP conoscendo l'indirizzo fisico MAC. Opera come ARP a livello di collegamento e in pratica svolge l'operazione inversa rispetto al protocollo di risoluzione degli indirizzi ARP.

La traduzione è necessaria in quanto la maggior parte delle reti locali collegate a Interne impiega un metodo diverso per indirizzare un messaggio al computer di destinazione.

Situazioni tipiche in cui si rende necessario l'utilizzo di questo protocollo riguardano per lo più sistemi diskless. Questi al momento dell'accensione caricano un'immagine del sistema operativo da un server remoto, per cui l'unica informazione di cui dispongono è l'indirizzo fisico della scheda di rete. RARP consente di mandare una richiesta di IP in broadcast, agli altri host connessi in rete. In generale la richiesta arriva ad un server RARP che contiene l'indirizzo di risposta nei propri file di configurazione.

Il fatto che RARP utilizzi il meccanismo di broadcast significa che l'utilizzo è limitato all'interno di una subnet, e che quindi i messaggi RARP non possono essere inoltrati dai router. Questo protocollo è infatti diventato obsoleto e sostituito dal BOOTP (BOOTstrap Protocol) o DHCP.

CAPITOLO 5 – LIVELLO DI RETE: Piano di controllo

Il piano di controllo del livello di collegamento, come abbiamo visto, racchiude la logica globale di rete che controlla come i datagrammi sono instradati tra i router su un percorso end-to-end tra sorgente e destinazione.

Nel definire la tecnica con la quale le tabelle di inoltro e dei flussi vengano calcolate possono essere adottati due approcci:

- **Controllo locale:** l'algoritmo di instradamento viene eseguito su ogni singolo router, all'interno del quale vengono effettuate sia le funzioni di inoltro (piano dei dati) che quelle di instradamento (piano di controllo). Ogni router ha un componente di instradamento che comunica con le componenti di instradamento degli altri router per calcolare la propria tabella di inoltro. I protocolli **OSPF E BGP** sono basati su tale approccio.
- **Controllo logicamente centralizzato:** il controller centralizzato calcola e distribuisce le tabelle di inoltro che devono essere utilizzate da ogni router. Il controller interagisce i **control agent** di ogni router i quali non partecipano attivamente all'elaborazione della tabella di inoltro. Le architetture SDN utilizzano tale approccio.

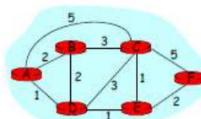
ALGORITMI DI INSTRADAMENTO

Gli algoritmi di instradamento hanno lo scopo di determinare i percorsi, o **cammini**, tra le sorgenti e le destinazioni attraverso la rete di router.

Tipicamente il percorso migliore è quello che ha un **costo minimo**, anche se nella pratica giocano un ruolo importante anche problematiche di interesse concreto, quali possono essere le policy adottate dalle parti interagenti (es. un utente della rete potrebbe non voler far passare i pacchetti per determinati router.).

Per formulare i problemi di instradamento, le reti sono modellate attraverso dei **grafi**:

- I **nodi** rappresentano i router della rete;
- Gli **archi** rappresentano i link fisici che connettono i vari router della rete.



Ad un arco è associato un valore che ne indica il **costo**. Nel grafo d'esempio, gli archi non sono orientati (sono bidirezionali), ragion per cui si presume un costo uguale per entrambi i versi di percorrenza. In generale un link può non essere simmetrico, e presentare un costo diverso per ognuno dei versi.

Dati due nodi qualsiasi, esistono più **percorsi** che li congiungono, ciascuno con il proprio costo (somma dei costi di tutti gli archi attraversati). L'obiettivo di un algoritmo di instradamento è l'individuazione dei percorsi a **costo minimo** tra la sorgente e la destinazione.

In generale si distinguono, in base al tipo di controllo adottato, due macro classi di algoritmi:

- **Algoritmi centralizzati:** sono algoritmi globali che ricevono in ingresso tutti i collegamenti tra i nodi e i relativi costi. Quindi “consci” del grafo della rete possono determinare il percorso a costo minimo;
- **Algoritmi decentralizzati:** il percorso a costo minimo è calcolato in modo distribuito e iterativo. Nessun nodo possiede informazioni complete sul costo di tutti i collegamenti di rete. Inizialmente i nodi conoscono solo i costi dei collegamenti a loro incidenti. Attraverso un processo iterativo, e tramite lo scambio di informazioni con i nodi adiacenti un nodo gradualmente calcola il percorso a costo minimo. Si dice che un singolo nodo deve prendere una decisione a livello globale sulla base di informazioni locali.

Un secondo criterio per classificare gli algoritmi di instradamento li suddivide in:

- **Algoritmi statici** (non adattivi): non basano le loro decisione di routing su misurazioni o stime del traffico corrente e della tipologia. La scelta dei percorsi da usare è calcolata in anticipo e caricata nei router.
- **Algoritmi dinamici** (adattivi): i percorsi sono individuati e aggiornati automaticamente in funzione delle modifiche di topologia o di traffico.

Parametri del processo decisionale

Una **metrica** è la misura di quanto buono è un percorso, ricavata dalla trasformazione di una grandezza fisica (o di una combinazione di essere), in forma numerica (**costo**), al fine di scegliere il percorso a costo inferiore come percorso migliore.

La scelta della metrica va fatta in funzione del tipo di traffico (campo TOS del pacchetto IP) e inoltre prendendo in considerazione alcuni fattori rilevanti:

- **non-ottimizzazione:** il compito del router è inoltrare il traffico degli utenti, non passare il tempo a fare routing. Conviene sempre prediligere soluzioni che, pur non essendo pienamente ottimizzate, non compromettano la funzionalità primaria della rete;
- **complessità:** più criteri vengono combinati, più l'algoritmo diventa complesso e richiede risorse computazionali in fase di esecuzione;
- **stabilità:** una metrica basata sulla banda disponibile, ad esempio, è troppo instabile, poiché dipende dal carico istantaneo di traffico, che è molto variabile nel tempo;
- **inconsistenza:** le metriche adottate dai nodi nella rete devono essere coerenti per evitare il rischio di cicli di pacchetti tra due router che utilizzano metriche differenti in conflitto.

Parametri correntemente usati per definire una metrica, e quindi confluiscono nel **costo del link** sono:

- **Bandwidth:** capacità di un link, tipicamente definita in bit per secondo;
- **Delay:** il tempo necessario per spedire un pacchetto da una sorgente ad una destinazione;
- **Load:** una misura del carico di un link;
- **Reliability:** o affidabilità riferita, ad esempio, all'error rate di un link;
- **Hop count:** numero di salti effettuati, cioè il numero di router da attraversare lungo il cammino.

Dal punto di vista dell'instradamento è comodo intendere la rete come un insieme di **sistemi autonomi (AS)** ognuno dei quali si occupa di gestire autonomamente e uniformemente il routing interno alla propria interrete. Per cui è possibile, ed è in pratica quello che succede, che i diversi AS adottino metriche e algoritmi di routing diversi tra di loro. L'importante che la scelta sia omogenea all'interno della rete.

TECNICHE DI ROUTING

La funzione principale di un router è quella di determinare i percorsi che i pacchetti devono seguire per arrivare a destinazione, partendo da una data sorgente. Ogni router si occupa quindi del processo di ricerca di un percorso per l'instradamento di pacchetti tra due nodi. Le principali tecniche utilizzate sono:

- **Routing by Network Address:** ogni pacchetto contiene l'indirizzo del nodo destinatario, il quale viene usato come chiave di accesso alla tabella di routing. Questa tecnica è molto semplice da implementare ed è utilizzata tipicamente nei protocolli non connessi (es IP);
- **Label Swapping:** consiste nell'inserimento di un'etichetta (label) all'interno di ogni pacchetto in transito. Quando un end-system ha necessità di instaurare una connessione con un altro end-system remoto, il primo invierà una richiesta di **path-setup** (instaurazione del percorso) alla rete, durante la quale, ogni router preparerà l'opportuna serie di etichette da utilizzare durante quella connessione. Dunque non è necessario indicare in ogni pacchetto il mittente. Il percorso è univocamente determinato in fase di preparazione del percorso.

UPDATE E TRANSITORI

Lo scambio delle informazioni tra i nodi (router) della rete che permette l'aggiornamento delle tabelle di routing può essere effettuato secondo due metodologie:

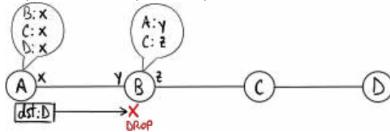
- **Broadcast periodico:** i router possono trasmettere agli altri router informazioni circa la raggiungibilità delle reti di propria competenza ad intervalli regolari di tempi. Risulta inefficiente, in quanto si spediscono informazioni anche quando non è cambiato nulla rispetto all'update precedente;
- **Event-driven:** in questo caso gli update sono inviati solo quando è cambiato qualcosa nella topologia oppure nello stato della rete (costo link). Questa tecnica garantisce un uso più efficiente della banda disponibile.

I cambiamenti di stato con i conseguenti *update* danno luogo a **fasi di transitorio**: non è possibile aggiornare contemporaneamente tutti i nodi di un sistema distribuito, poiché una variazione è propagata nella rete a velocità finita. Durante il transitorio le informazioni di stato nella rete possono essere inconsistenti (alcuni nodi hanno già le informazioni nuove, mentre altri hanno ancora le informazioni vecchie).

Il tempo utile perché il transitorio si estingua è chiamato **tempo di convergenza** e dipende dal particolare algoritmo di routing implementato.

Due problemi comuni affliggono gli algoritmi di instradamento durante il transitorio:

- **Black Hole:** i pacchetti verso una data destinazione sono inviati ad un router il quale, non disponendo di un percorso per la destinazione, li scarta;



- **Routing loop:** un router invia un pacchetto su un link ma, a causa di un'inconsistenza nelle tabelle di instradamento, il router all'altra estremità del link lo rispedisce indietro, creando path ciclici. Può provocare una saturazione del link se i pacchetti non sono scartati (eccessivo time to live).

ALGORITMI DISTANCE VECTOR

L'instradamento **distance vector**, noto anche come routing Bellman-Ford, è un tipo di algoritmo di **routing dinamico**, che tiene conto del carico istantaneo della rete. È un algoritmo di tipo:

- **distribuito** nel senso che ciascun nodo riceve parte dell'informazione da uno o più dei suoi neighbors (vicini) direttamente connessi, a cui, dopo aver effettuato il calcolo, restituisce i risultati.
- **Iterativo** nel senso che questo processo si ripete fino a quando non avviene ulteriore scambio informativo tra vicini. È dunque anche auto-terminante.
- **Asincrono**: nel senso che non richiede che tutti i nodi operino al passo con gli altri.

Concettualmente l'algoritmo misura la **distanza** di ogni singolo router dagli altri nodi della rete ricevendo i dati dai router vicini collegati. A partire da tali dati, utilizzando l'**algoritmo di Bellman-Ford**, il router costruisce una tabella che indica per ogni entry:

- Un nodo della rete raggiungibile
- Il next hop
- Il numero di hop necessari per raggiungere la destinazione (o altra metrica per il costo)

Periodicamente il router aggiorna il proprio vettore delle distanze e comunica la propria tabella ai vicini. I router che ricevono tale messaggio aggiornano la propria tabella modificando le informazioni sui cammini aggiunti, modificati o cambiati. Dopo sufficienti scambi di informazioni, ciascun router potrà avere una riga per ogni altro nodo della rete.

Esempio:

Destin.	Dist.	Route
net 1	0	direct
net 2	0	direct
net 4	8	router L
net 17	5	router M
net 24	6	router A
net 30	2	router Q
net 42	2	router A

→

Destin.	Dist.
net 1	2
net 4	3
net 17	6
net 21	4
net 24	5
net 30	10
net 42	3

Messaggio di aggiornamento del router A

Destin.	Dst.	Route
net 1	0	direct
net 2	0	direct
net 4	4	router A
net 17	5	router M
net 24	6	router A
net 30	2	router Q
net 42	4	router A
net 21	5	router A

Tabella aggiornata del router B

Il distance vector è un algoritmo di facile implementazione e richiede risorse computazionali limitate, quindi hardware nei router economico, però presenta alcuni svantaggi:

- Ogni messaggio deve contenere un'intera tabella di routing, il che occupa molta banda.
- La convergenza può essere piuttosto lenta, proporzionale al link e al router più lento della rete. Se lo stato della rete cambia velocemente, le rotte possono risultare inconsistenti.
- Può scatenare routing loop dovuti a particolari cambiamenti nella topologia.

Nella pratica, per il calcolo dei percorsi a costo minimo l'algoritmo si avvale della **Formula di Bellman-Ford**

$$d_x(y) = \min_v \{c(x,y) + d_v(y)\}$$

Con v sono denotati tutti i vicini di x . Se dopo aver viaggiato da x ad un generico vicino v , consideriamo il percorso a costo minimo da v a y , il costo del percorso sarà $c(x,y)+d_v(y)$. Dato che per raggiungere y , partendo da x , dobbiamo iniziare viaggiando verso qualche vicino v , il **costo minimo da x a y** è il minimo di $c(x,y)+d_v(y)$ calcolato su tutti i nodi.

Ad ogni nodo, x :

```

1 Inizializzazione:
2 per tutti i nodi adiacenti v:
3    $D^*(v,v) = \text{infinito}$       {il simbolo * significa "per ogni riga"}
4    $D^*(v,v) = c(x,v)$ 
5   per tutte le destinazioni, y
6     manda  $\min_w D^*(y,w)$  a ogni vicino

7 loop
8   aspetta (fino a quando vedo una modifica nel costo di un
9     collegamento oppure ricevo un messaggio da un vicino v)
10
11  if ( $c(x,v)$  cambia di d)
12    { cambia il costo a tutte le dest. via vicino v di d }
13    { nota: d può essere positivo o negativo }
14    per tutte le destinazioni y:  $D^*(y,v) = D^*(y,v) + d$ 
15
16  else if (nuovo mess. aggiornamento da v verso destinazione y)
17    { cammino minimo da v a y è cambiato }
18    { V ha mandato un nuovo valore per il suo  $\min_w D^*(y,w)$  }
19    { chiama questo valore "newval" }
20    per la singola destinazione y:  $D^*(y,v) = c(x,v) + newval$ 
21
22
23  if hai un nuovo  $\min_w D^*(y,w)$  per una qualunque destinazione y
24    manda il nuovo valore di  $\min_w D^*(y,w)$  a tutti i vicini
25
26 forever

```

Un nodo x quando vede il cambiamento di costo di uno dei collegamenti direttamente connessi o quando riceve da qualche vicino un vettore aggiornato (**linea 9-10**) aggiorna la propria stima del vettore delle distanze (**linea 15 e 21**) e, se si verifica un cambiamento nel costo del percorso a costo minimo (**linea 23**), trasmette ai propri vicini il proprio nuovo vettore delle distanze (**linea 24**).

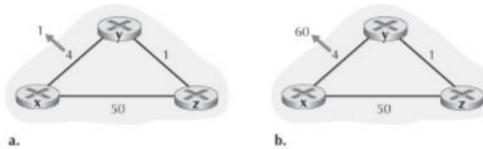
Problema del conteggio all'infinito

Il distance vector funziona in teoria, ma in alcuni casi, sebbene converga al risultato corretto, può farlo molto lentamente. In particolare reagisce rapidamente alle "buone notizie" e lentamente alle "cattive notizie".

Questo significa che, ad esempio, in uno scenario in cui il costo del collegamento da y a x passa da 4 ad 1, l'algoritmo provocherà i seguenti eventi:

- All'istante t_0 , y rileva il cambiamento nel costo del collegamento (da 4 a 1), aggiorna il proprio vettore delle distanze e informa i vicini del cambiamento.
- All'istante t_1 , z riceve l'aggiornamento da y e aggiorna la propria tabella, calcola un nuovo costo minimo verso x (che passa da 5 a 2) e invia il nuovo vettore delle distanze ai vicini
- All'istante t_2 , y riceve l'aggiornamento da z e aggiorna la propria tabella delle distanze. I costi minimi di y non cambiano e y non manda alcun messaggio a z .

Dopo due interazioni l'algoritmo raggiunge uno stato di quiete.



Prendiamo in considerazione lo scenario in cui il costo di un collegamento tra x e y aumenta da 4 a 60.

- All'istante t_0 , y rileva che il costo del collegamento è passato da 4 a 60 e calcola il suo nuovo percorso a costo minimo con la formula $D_y(x) = \min \{c(y,x)+D_x(x), c(y,z)+D_z(x)\} = \min \{60+0, 1+5\}=6$

Ovviamente se si avesse una visione globale della rete sarebbe ovvio che il nuovo costo attraverso z è errato. Ma l'unica informazione che il nodo y possiede è che il costo diretto verso x è 60 e che z ha ultimamente detto a y di essere in grado di raggiungere x con un costo di 5. All'istante t_1 , abbiamo un **istradamento ciclico**: al fine di giungere a x , y fa passare il percorso per z e lo fa passare per y .

- Dato che il nodo y ha calcolato un nuovo costo minimo verso x , informa z del suo nuovo vettore delle distanze al tempo t_1
- In un istante successivo, z riceve il nuovo vettore delle distanze di y , che indica che il costo minimo di y verso x è 6, sa che può giungere a y a costo 1 e quindi calcola un nuovo costo minimo verso x pari a $D_z(x) = \min \{50+0, 6+1\}=7$. Dato che il costo minimo di z verso x è aumentato, z informa y del suo nuovo vettore delle distanze al tempo t_2 .
- Y determina un altro costo minimo verso x che è 8 e invia a z il suo nuovo vettore.

Il ciclo prosegue per 44 interazioni, fino a quando z considera il costo del proprio percorso attraverso y maggiore di 50. A questo punto, z determina che il percorso a costo minimo verso x è quello con costo 50.

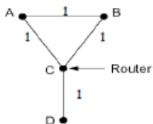
Il problema descritto viene talvolta detto **problema di conteggio all'infinito** e dimostra che le "cattive notizie" viaggiano senza dubbio assai lentamente.

Algoritmo Split Horizon

È una algoritmo che cerca di prevenire i routing loops (count to infinity). Esso impedisce ad un router di inviare informazioni di route al router da cui le ha apprese.

Un modello avanzato di questa tecnica è detto **Poison Reverse**. Il router invierà un'informazione allo stesso router da cui l'ha ricevuta, ma con una metrica “infinita”, comunicando di fatto a quel router che per lui la destinazione è irraggiungibile. Lo scopo è simile alla modalità semplice, ma con una maggiore efficacia contro gli anelli di routing.

La tecnica descritta può essere utile a volte, ma non rappresenta una soluzione generale al problema. Infatti se i cicli non riguardano semplicemente due nodi adiacenti, essi non verranno rilevati.



Nell'esempio raffigurato quando il link tra C e D si interrompe, C setterà la sua distanza da D ad infinito. Però, A userà B per andare a D e B userà A per andare a D. Dopo questi update, sia A che B riporteranno un nuovo percorso da C a D (diverso da infinito).

ALGORITMI LINK STATE

In un instradamento **link state** la topologia di rete e tutti i costi dei collegamenti sono noti, ossia disponibili in input all'algoritmo. Ciò si ottiene facendo inviare a ciascun nodo pacchetti sullo stato (**Link State Packet**) dei suoi collegamenti a **tutti** gli altri nodi della rete, ossia in broadcast.

Ciascun nodo esegue in maniere indipendente l'algoritmo per determinare il cammino minimo per raggiungere ogni nodo della rete ponendosi come radice dell'albero dei cammini minimi.

Questo tipo di approccio al routing è utilizzato in alcuni protocolli di instradamento, e soprattutto nel **protocollo OSPF**. Questi tipi di algoritmi presentano diversi **vantaggi**:

- Possono gestire reti composte da un gran numero di nodi;
- Convergono rapidamente al cammino minimo;
- Difficilmente generano cammini ciclici;
- È facile da comprendere poiché ogni nodo ha la mappa della rete.

Il principale **svantaggio** di un algoritmo Link State è la complessità di realizzazione, anche dovuta alla notevole capacità di memoria ed elaborazione richiesti dai router stessi.

Per il funzionamento del protocollo è fondamentale il pacchetto **LSP** che contiene:

- Informazioni relative allo stato di ogni link connesso al router
- L'identità di ogni vicino connesso all'altro estremo del link
- Il costo del link
- Numero di sequenza per l'LSP: trasmessi continuamente per aggiornare lo stato della rete.
- Una checksum per capire se l'LSP è stato corrotto nel viaggio;
- Lifetime: la validità di ogni LSP è limitata nel tempo.

LSP flooding

Ogni nodo della rete, come abbiamo detto, inoltra in broadcast, a tutti i suoi collegamenti, il suo LSP. Da un punto di vista globale gli LSP si diffondono su tutta la rete tramite un meccanismo di flooding.

La trasmissione avviene periodicamente ogni tot secondi, oppure quando viene rilevata una variazione nella topologia locale (adiacenze) ossia:

- Viene riconosciuto un nuovo vicino
- Il costo verso un vicino è cambiato
- Si è persa la connettività verso un vicino precedentemente raggiungibile.

Gestione degli LSP

Quando un nodo riceve un pacchetto LSP confronta il numero di sequenza del pacchetto con quello dell'ultimo pacchetto ricevuto da quel nodo:

- Se il numero di sequenza indica che il pacchetto è più recente di quello memorizzato, il nuovo pacchetto viene memorizzato e inoltrato a tutti i nodi collegati, eccetto quello da cui è stato ricevuto;
- Se il numero di sequenza è invariato il pacchetto viene scartato;
- Se il numero di sequenza indica che il pacchetto ricevuto è meno recente di quello memorizzato, quest'ultimo viene trasmesso al nodo mittente.

Ogni nodo memorizza i pacchetti ricevuti e costruisce una mappa completa e aggiornata della rete mantenuta in un **Link State Database** (uguale per tutti). Quest'ultimo può essere rappresentato con una tabella che mostra quali router è in grado di raggiungere un router con relativo costo del link o anche tramite una **matrice di raggiungibilità**, dove sulle righe ci sono tutte le destinazioni, sulle colonne tutti i mittenti e ogni casella rappresenta il costo del collegamento tra sorgente e destinazione.

LSP Database

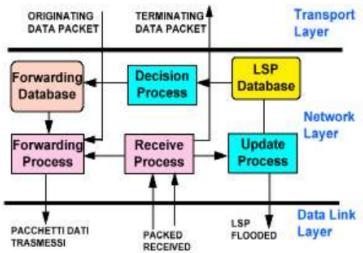
A	B/2
B	A/2 D/3 E/2
C	D/1
D	B/3 C/1 G/1
E	B/2 F/5 G/2
F	E/5 H/4
G	D/1 E/2 H/1
H	F/4 G/1

(replicato su ogni IS)

SORGENTE									
		A	B	C	D	E	F	G	H
DESTINAZIONE	A	0	2						
	B	2	0	3	2				
C		0	1						
D	3	1	0			1			
E	2		0	5	2				
F			5	0	4				
G		1	2		0	1			
H			4	1	0				

Routing

La conoscenza topologica non è sufficiente di per sé a determinare i percorsi, perché anche se sono noti i costi dei singoli collegamenti tra i router non si hanno conoscenze sufficienti per stabilire i percorsi globali tra i router. Sono necessari algoritmi di Forwarding, ad esempio quello di **Dijkstra**, per elaborare il Link State Database e produrre il **Forwarding Database**.



Sulla base di ogni singola destinazione, l'algoritmo decide qual è il next hop attraverso un calcolo nel quale ogni nodo si mette come radice di un albero e raggiunge con le sue foglie tutti gli altri nodi della rete.

Il forwarding database contiene per ogni nodo destinazione l'insieme delle coppie {path, collegamento in uscita} e la dimensione di tale insieme.

ALGORITMO DI DIJKSTRA

Ogni nodo ha a disposizione il grafo della rete ed utilizza l'algoritmo per costruire lo **Spanning Tree** del grafo, ovvero l'albero dei cammini di costo minimo. Ad ogni nodo si assegna un'etichetta che rappresenta il costo massimo per raggiungere quel nodo. L'algoritmo modifica le etichette cercando di minimizzarne il valore e di renderle permanenti.

L'algoritmo è di tipo iterativo: ad ogni interazione si determina il costo minimo verso un altro nodo, e se ne è il numero di nodi della rete, alla n-esima interazione il forwarding è completo per l'intera rete.

- **C(i,j)**: costo del collegamento tra i e j. Se non c'è un collegamento il costo è infinito. Assumiamo il costo simmetrico, anche se non è strettamente necessario.
- **D(v)**: il costo attualmente calcolato del percorso che va dalla sorgente al nodo v.
- **P(v)**: predecessore nel cammino di costo minimo dalla sorgente fino a v.
- **N**: insieme dei nodi per i quali la distanza minima è stata trovata. Alla fine dell'algoritmo, questo N deve coincidere con la totalità dei nodi sulla rete

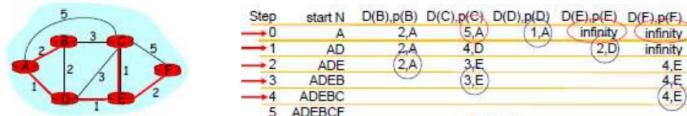
```

1 Inizializzazione:
2 N = {A}
3 per tutti i nodi v
4 if (v e' adiacente a A)
5   then D(v) = c(A,v)
6   else D(v) = infinity
7
8 Loop
9 sia w non in N tale che D(w) è minimo
10 aggiungi w a N
11 aggiorna D(v) per ogni v adiacente a w e non in N:
12   D(v) = min(D(v), D(w) + c(w,v))
13   {il nuovo costo fino a v è o il vecchio costo, oppure il costo del
      cammino più breve fino a w più il costo da w a v }
15 fino a quando tutti i nodi sono in N

```

L'algoritmo consiste in un passo di inizializzazione, più un ciclo di durata pari al numero di nodi della rete. Al termine avremo i percorsi più brevi dal nodo sorgente a tutti gli altri nodi.

Consideriamo una rete di esempio fatta di sei nodi. Si vuole trovare il minimo spanning tree che collega A con tutti gli altri nodi della rete. Si esegue l'algoritmo



Step 0 : Al passo di inizializzazione, nell'insieme N è incluso solo il nodo A. Tutti i nodi che non sono direttamente collegati ad A hanno distanza da A infinita, mentre i nodi direttamente collegati presentano come costo il costo del collegamento diretto.

Step 1 : Si entra nel ciclo. Viene scelto uno dei nodi che non stanno nello spanning tree e soprattutto quello che ha costo minimo (D) e viene aggiunto allo spanning tree. E' stato trovato in effetti il percorso a costo minimo che collega A a D. Vengono ricalcolate le distanze per i nodi raggiungibili da D (B,C ed E).

Step 2 : Viene scelto il nodo che ha costo minimo da D.

Step 3 : ...

Problematiche

In un algoritmo di questo tipo se il costo del link è proporzionale al traffico su quel link, cioè viene scelta una metrica che porta in conto livello di congestione del link, allora è possibile che si creino oscillazioni. L'algoritmo periodicamente, infatti, ricalcola le route instradando i pacchetti verso i link meno trafficati, rendendoli di fatto più trafficati rispetto a prima. Periodicamente il traffico oscilla tra diversi rami, mentre sarebbe preferibile che questo traffico si distribuisse uniformemente.

CONFRONTO LS E DV

Nel DV ciascun nodo dialoga solo con i vicini direttamente connessi, informandoli delle stime a costo minimo da sé stesso a tutti i nodi (che conosce) nella rete. Nel LS, ciascun nodo dialoga con tutti gli altri nodi via broadcast, ma comunica loro solo i costi dei collegamenti direttamente connessi.

- **Complessità dei messaggi:** LS richiede che ciascun nodo conosca il costo di ogni collegamento nella rete. Ciò implica l'invio di $O(|N| * |E|)$ messaggi. Inoltre ogni qualvolta cambia il costo di un collegamento, il nuovo costo deve essere comunicato a tutti i nodi. A ogni interazione l'instradamento DV richiede scambi di messaggi tra nodi adiacenti. Il tempo di convergenza può dipendere da molti fattori. Quando cambiano i costi dei collegamenti, l'instradamento DV propaga i risultati dei costi cambiati se il nuovo costo ha causato la variazione del percorso a costo minimo per uno o più nodi connessi a tale collegamento.
- **Velocità di convergenza:** l'implementazione di LS è un algoritmo $O(|N|^2)$ che richiede $O(|N| * |E|)$ messaggi. L'algoritmo del DV può convergere lentamente e può presentare cicli di instradamento e anche il problema del conteggio all'infinito.
- **Robustezza:** Con LS un router può comunicare via broadcast un costo sbagliato per uno dei suoi collegamenti connessi (ma non per altri). Un nodo può anche alterare o eliminare i pacchetti ricevuti in broadcast LS. Ma i nodi LS si occupano di calcolare soltanto le proprie tabelle di inoltro, e gli altri nodi effettuano calcoli simili per quanto li riguarda. Ciò significa che i calcoli di instradamento sono in qualche modo isolati, il che fornisce un certo grado di robustezza. Con DV, un nodo può comunicare percorsi a costo minimo errati a tutte le destinazioni.

ROUTING IN INTERNET

Il modello semplificato utilizzato per la trattazione degli algoritmi di instradamento era quello che vedeva la rete come un insieme di router interconnessi, ognuno di questi indistinguibile dagli altri, nel senso che eseguivano tutti lo stesso algoritmo per calcolare le route. Questo modello risulta un po' semplicistico e presenta diverse problematiche:

- **Scalabilità:** al crescere del numero di router, il tempo richiesto per calcolare, memorizzare e comunicare le informazioni di instradamento diventa proibitivo.
- **Autonomia amministrativa:** Internet è una rete di ISP che generalmente desiderano gestire i propri router liberamente e in modo diversificato all'interno della rete.

Questi problemi possono essere risolti organizzando la rete in **Autonomous System**, generalmente composti da gruppi di router amministrato da un'unica autorità. Ogni AS deve essere identificato con un AD number (16 bit) assegnato da ICANN. I router di uno stesso AS eseguono lo stesso algoritmo di instradamento.

Ogni AS è responsabile del **routing interno** delle sue reti ed inoltre tutti gli AS devono scambiarsi informazioni di raggiungibilità tramite un **routing esterno**.

Le tabelle di routing interne di un AS sono mantenute da algoritmi come :

- **RIP** (distance vector)
- **OSPF** (link state)
- **IGRP** (Interior Gateway Routing Protocol)

Le tabelle di routing esterne di un AS sono mantenute invece da :

- **EGP** (Exterior Gateway Protocol), ormai obsoleto
- **BGP** (Border Gateway Protocol): approccio path vector

CAPITOLO 6 – LIVELLO TRASPORTO

Un protocollo a livello di trasporto mette a disposizione una **comunicazione logica** tra processi applicativi di host differenti. Per comunicazione logica si intende, dal punto di vista dell'applicazione, che tutto proceda come se gli host che eseguono i processi fossero direttamente connessi. La differenza con il livello di rete è che se quest'ultimo si preoccupa del trasferimento e della consegna dei dati **tra end-system**, il livello di trasporto si occupa del trasferimento e smistamento dati **tra processi** residenti su determinati end-system.

I processi applicativi usano la comunicazione logica fornita dal livello di trasporto per scambiare messaggi, senza preoccuparsi dell'infrastruttura fisica utilizzata per trasportarli. I messaggi ricevuti dal livello applicativo sono incapsulati in pacchetti a livello di trasporto detti **segmenti**.

È un livello generalmente implementato nei sistemi periferici, ma non negli router della rete.

Uno dei servizi fondamentali del livello di trasporto è quello aumentare l'efficienza e l'affidabilità non garantite dai livelli superiori.

Una rete TCP/IP, e in particolare Internet, mette a disposizione del livello di applicazione due diversi protocolli: **UDP** (user datagram protocol), che fornisce alle applicazioni un servizio non affidabile e non orientato alla connessione, e **TPC** (transmission control protocol), che offre un servizio affidabile e orientato alla connessione. Come visto, lo sviluppatore dell'applicazione sceglie tra UDP e TCP durante la creazione delle socket.

I modelli di servizio offerti da UDP e TCP hanno i seguenti compiti fondamentali:

- Estendere il servizio di consegna IP tra sistemi periferici a quello di consegna tra processi in esecuzione sui sistemi periferici. Il passaggio da consegna **host-to-host** a consegna **process-to-process** viene detto **multiplexing e demultiplexing** a livello di trasporto.
- Fornire un controllo di integrità includendo campi per il riconoscimento di errori nelle intestazioni dei propri segmenti.

Questi due servizi minimi a livello di trasporto sono gli unici offerti da UDP. In particolare **UDP**, come IP, costituisce un **servizio inaffidabile** e cioè non garantisce che i dati inviati da un processo arrivino intatti (e neppure che arrivino) al processo destinatario.

TCP, d'altra parte, offre alle applicazioni diversi servizi aggiuntivi che permettono di fornire un **trasferimento dati affidabile**:

- Controllo di flusso
- Acknowledgement
- Timer
- Controllo di congestione

Multiplexing e Demultiplexing

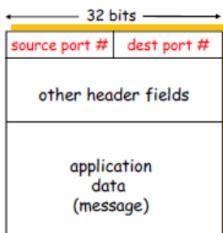
Per multiplexing e demultiplexing si intende il passaggio del servizio di trasporto da host a host (fornito con dal livello di rete) a un servizio di trasporto da processo a processo per le applicazioni in esecuzione sugli host. Nell'host destinatario il livello di trasporto riceve segmenti dal livello di rete immediatamente sottostante. Il livello di trasporto ha il compito di consegnare i dati di questi segmenti al processo applicativo appropriato in esecuzione nell'host.

Un processo può gestire una o più **socket**, attraverso le quali i dati fluiscono dalla rete al processo viceversa. Per cui il livello di trasporto nell'host di ricezione in realtà non trasferisce i dati direttamente a un processo, ma piuttosto ad una socket che fa da intermediario. Nello stesso istante saranno presenti più socket aperte in riferimento ad un unico processo (identificata univocamente).

Il compito di trasportare i dati dei segmenti a livello di trasporto verso la giusta socket viene detto **demultiplexing**. Il compito di radunare frammenti di dati da diverse socket sull'host di origine e incapsulare ognuno con intestazioni a livello di trasporto per creare segmenti e passarli al livello di rete, viene detto **multiplexing**.

Il multiplexing e il demultiplexing è realizzato inserendo all'interno dei segmenti UDP e TCP gli identificatori delle socket:

- Campo del numero di porta di origine
- Campo del numero di porta di destinazione



TCP/UDP segment format

Quando arriva un segmento all'host, il livello di trasporto esamina il numero della porta di destinazione e dirige il segmento verso la socket corrispondente. I dati del segmento passano, quindi, dalla socket al processo assegnato.

Mux e Demux Connectionless

Quando un processo in esecuzione crea una socket UDP, il livello di trasporto le assegna automaticamente un numero di porta compreso tra 1024 e 65535 che non sia ancora utilizzato. In alternativa le si può assegnare uno specifico numero di porta con il metodo **bind()**.

Quando un processo su un host A con porta **x** vuole inviare un blocco di dati a un processo con porta **y** nell'host B, il livello di trasporto di A crea un segmento che include i dati, il numero di porta di origine (x) e il numero di porta di destinazione (y). Il livello di trasporto passa il segmento risultante al livello di rete, che lo incapsula in un datagramma IP, ed effettua un tentativo best-effort di consegna del segmento all'host in ricezione. Se il segmento arriva all'host B, il suo livello di trasporto esamina il numero di porta di destinazione del segmento e lo consegna alla propria socket identificata da **y**.

È da ricordare che una socket UDP viene identificata completamente da una coppia che consiste di un indirizzo IP e di un numero di porta di destinazione. Di conseguenza, se due segmenti UDP se hanno stesso indirizzo IP e porta di destinazione, saranno diretti sulla stessa socket qualunque sia l'indirizzo e la porta di origine. Per quanto riguarda indirizzo e porto di origine, questi sono inclusi nei segmenti come **indirizzi di ritorno** nel caso l'host B voglia rispondere all'host A.

Mux e Demux Connection-oriented

La differenza tra una socket TCP e una UDP risiede nel fatto che la prima è identificata da quattro parametri: indirizzo e porta di origine, indirizzo e porta di destinazione. Pertanto quando un segmento TCP giunge alla rete di un host, quest'ultimo utilizza i quattro parametri per dirigere (fare demultiplexing) il segmento verso la socket appropriata.

Al contrario di UDP, due segmenti TCP in arrivo, aventi indirizzi IP di origine o numeri di porta di origine diversi, saranno sempre diretti a due socket differenti, anche a fronte di indirizzo IP e porta di destinazione uguali.

PROTOCOLLO UDP

UDP, User Datagram Protocol, definito in RFC 768, fa praticamente il minimo che un protocollo di trasporto debba fare. A parte la funzione di multiplexing/demultiplexing e una forma di controllo degli errori molto semplice, non aggiunge nulla a IP.

UDP prende i messaggi dal processo applicativo, aggiunge il numero di porta di origine e di destinazione per il mux/demux, aggiunge altri due piccoli campi e passa il segmento risultante al livello di rete. Questi incapsula il segmento in un datagramma IP ed effettua un tentativo di consegna all'host di destinazione in modalità best-effort.

In UDP non esiste handshaking tra le entità di invio e di ricezione a livello trasporto per questo motivo si dice **non orientato alla connessione**.

Nonostante la mancanza di garanzie, per molte applicazioni è preferibile UDP a TCP per i seguenti motivi:

- **Controllo più fine a livello applicazione** su quali dati sono inviati e quando (TCP dispone di un meccanismo di controllo della congestione che può ritardare l'invio). Utile per applicazioni in tempo reale.
- **Nessuna connessione stabilità** e nessun relativo ritardo.

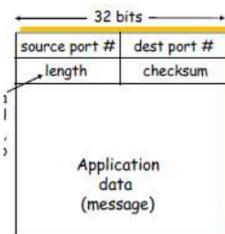
- **Nessuno stato di connessione** per cui non è necessario includere buffer di ricezione e invio, parametri per il controllo della congestione, parametri sul numero di sequenza e di acknowledgment. Un server UDP può supportare molti più client attivi rispetto ad un server TCP.
- **Minor spazio usato per l'intestazione** del pacchetto. L'intestazione dei pacchetti TCP aggiunge 20 byte, mentre UDP solo 8.

Applicazione	Protocollo a livello di applicazione	Protocollo di trasporto sottostante
Posta elettronica	SMTP	TCP
Accesso a terminali remoti	Telnet	TCP
Web	HTTP	TCP
Trasferimento file	FTP	TCP
Server di file remoti	NFS	generalmente UDP
Dati multimediali in streaming	generalmente proprietario	UDP o TCP
Telefonia su Internet	generalmente proprietario	UDP o TCP
Gestione di rete	SNMP	generalmente UDP
Protocolli di instradamento	RIP	generalmente UDP
Traduzione di nomi	DNS	generalmente UDP

Segmenti UDP

L'intestazione UDP presenta solo quattro campi di due byte ciascuno.

- I numeri di porta consentono all'host di destinazione di trasferire i dati applicativi al processo corretto (Demultiplexing)
- Il campo lunghezza specifica il numero di byte del segmento UDP (header + dati) ed è necessario perché la lunghezza del campo dati può essere diversa tra un segmento e quello successivo.
- La checksum è utilizzata dall'host ricevente per verificare se sono avvenuti errori nel segmento.



La **checksum UDP** serve per il rilevamento degli errori. Viene utilizzato per determinare se i bit del segmento UDP sono stati alterati durante il loro trasferimento da sorgente a destinazione.

Lato mittente UDP effettua il complemento a 1 della somma di tutte le parole da 16 bit nel segmento, e l'eventuale riporto finale viene sommato al primo bit. Tale risultato viene posto nel campo checksum del segmento UDP.

UDP mette a disposizione un checksum, anche se molti protocolli a livello di collegamento prevendono il controllo degli errori, perché non c'è garanzia che tutti i collegamenti tra sorgente e destinazione controllino gli errori. Si mette a disposizione così a livello di trasporto un meccanismo di verifica su base **end-to-end**.

TRASMISSIONE DATI AFFIDABILE SU UN CANALE CON ERRORI

Il compito dei **protocolli di trasferimento dati affidabile** è l'implementazione di un'astrazione di un canale affidabile tramite il quale si possono trasferire i dati. Con un canale affidabile vengono evitati:

- Presenza di errori
- Perdita di pacchetti
- Ordine dei pacchetti non garantito
- Duplicazione di pacchetti

E sono introdotti:

- Controllo di flusso
- Controllo di congestione

In un modello semplificato, in cui il protocollo di trasferimento presenti al livello sottostante un canale di trasmissione non affidabile in cui l'unico problema è che i bit di un pacchetto possono essere corrotti è necessario:

- **Rilevamento dell'errore:** è richiesto un meccanismo che consenta al destinatario di rilevare gli errori sui bit. Vengono utilizzate delle notifiche (**acknowledgement**) positive o negative per comunicare al mittente cosa sia stato ricevuto correttamente e cosa no, chiedendone quindi il rinvio.
- **Feedback del destinatario:** dato che mittente e destinatario sono generalmente in esecuzione su sistemi periferici diversi, l'unico modo che ha il mittente per conoscere la visione del destinatario consiste nel **feedback esplicito** del destinatario. Le risposte di notifica positiva **ACK** e negativa **NAK**.
- **Ritrasmissione:** un pacchetto ricevuto con errori sarà ritrasmesso dal mittente.

In un protocollo che utilizza questi strumenti, e cioè rilevamento dell'errore e notifiche positive e negative:

1. Il protocollo di trasporto lato mittente attende i dati da raccogliere dal livello superiore
2. Quando si verifica l'evento, il lato mittente crea un pacchetto contenente i dati da inviare, insieme al checksum e spedisce il pacchetto.
3. Il lato mittente si mette in attesa di un pacchetto ACK o NAK dal destinatario.
 - a. Se riceve un ACK sa che il pacchetto trasmesso più di recente è stato ricevuto correttamente e pertanto il protocollo ritorna allo stato di attesa dei dati provenienti dal livello superiore
 - b. Se riceve un NAK il protocollo ritrasmette l'ultimo pacchetto e attende una risposta alla ritrasmissione

È importante osservare che quando il mittente è nello stato di attesa di ACK o NAK non può recepire dati dal livello superiore. Quindi il mittente non invia nuovi dati finché non è certo che il destinatario abbia ricevuto correttamente il pacchetto corrente.

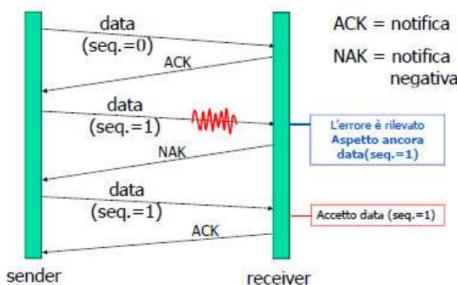
Problema ACK NAK corrotti

L'osservazione da fare è che gli stessi pacchetti di ACK e NAK possono essere corrotti. Se un ACK o un NAK è corrotto, il mittente non ha modo di sapere se il destinatario abbia ricevuto correttamente l'ultimo blocco di dati trasmessi.

La soluzione a questo problema potrebbe essere semplicemente quella che prevede, in ogni caso, il rinvio da parte del mittente qualora l'ACK o il NAK risulti alterato. Con questa soluzione ovviamente, se il pacchetto dati trasmesso arriva correttamente la prima volta, in seguito ad un ACK corrotto, con il relativo rinvio si introduggeranno dei **pacchetti duplicati**. Conseguentemente il destinatario, non sapendo se l'ultimo ACK inviato sia stato ricevuto correttamente dal mittente, non può sapere a priori se un pacchetto in arrivo è frutto di una ritrasmissione o è un pacchetto contenente dati nuovi.



La soluzione generale al problema consiste nell'aggiungere un campo al pacchetto dati, obbligando il mittente a numerare i propri pacchetti dati con un **numero di sequenza** nel nuovo campo. Al destinatario sarà sufficiente controllare questo numero per sapere se il pacchetto ricevuto rappresenti o meno una ritrasmissione. In accordo con ciò, i pacchetti ACK e NAK non devono indicare il numero di sequenza per non essere confusi con i pacchetti dati.



Protocollo senza NAK

In un'altra versione del protocollo, si possono avere le stesse funzionalità del protocollo descritto precedentemente, utilizzando però solo notifiche di ACK.

Possiamo ottenere lo stesso effetto di un NAK spedendo piuttosto un ACK per il più recente pacchetto ricevuto correttamente. Un mittente che riceve due ACK per lo stesso pacchetto (ossia riceve **ACK duplicati**) sa che il destinatario non ha ricevuto correttamente il pacchetto successivo a quello confermato due volte.

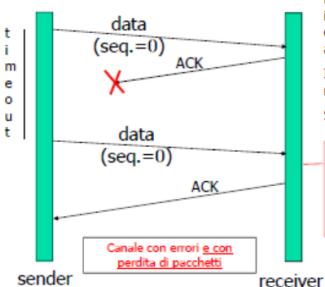
Il destinatario deve ora includere il numero di sequenza del pacchetto di cui invia l'acknowledgement all'interno del messaggio ACK, e il mittente deve ora controllare il numero di sequenza del pacchetto confermato da un messaggio ACK ricevuto.

TRASMISSIONE DATI AFFIDABILE SU UN CANALE CON PERDITE E ERRORI

Supponiamo ora che il canale di trasmissione, oltre a danneggiare i bit, possa anche **smarrire** i pacchetti. Il protocollo ora deve preoccuparsi di due aspetti aggiuntivi:

- Rilevare lo smarrimento di pacchetto
- Cosa fare quando avviene la perdita

Supposto che il mittente spedisca un pacchetto dati e che questo o l'ACK corrispondente del ricevente vada smarrito, se questo è disposto ad attendere un tempo sufficiente per essere certo dello smarrimento del pacchetto, allora può semplicemente ritrasmettere il pacchetto.



Il problema è quello di quantificare questo timeout. Certamente, il mittente deve attendere almeno per il minimo ritardo di andata e ritorno tra mittente e destinatario più il tempo richiesto per l'elaborazione di un pacchetto da parte del destinatario.

In molte reti questo ritardo relativo al caso peggiore è difficile perfino da stimare ed inoltre il protocollo dovrebbe ipoteticamente risolvere la perdita di dati prima possibile e quindi è inefficiente aspettare il ritardo peggiore calcolabile.

Nella pratica l'approccio adottato è scegliere in modo assennato un valore di tempo tale per cui la perdita di pacchetti risulta probabile, anche se non garantito. Se non si riceve un ACK in questo lasso di tempo, il pacchetto viene ritrasmesso.

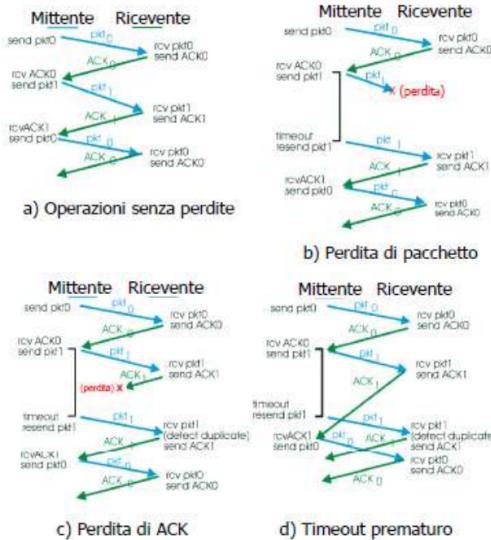
Ragionevolmente, se un pacchetto sperimenta un ritardo particolarmente largo, ma comunque arriva a destinazione, il mittente potrebbe ritrasmetterle introducendo **pacchetto duplicati** sulla canale. Il protocollo gestisce i duplicati con il meccanismo introdotto precedentemente dei numeri di sequenza.

Implementazione del timeout

Implementare un meccanismo di ritrasmissione basato sul tempo richiede un **contatore** in grado di segnalare al mittente l'avvenuta scadenza di un dato lasso di tempo. Il mittente dovrà essere in grado:

1. Inizializzare il contatore ogni volta che invia il pacchetto
2. Rispondere a un interrupt generato dal timer con l'azione appropriata
3. Fermare il contatore

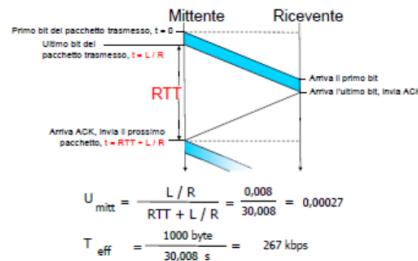
Un riepilogo del funzionamento del protocollo fin qui descritto è rappresentato di seguito. Il protocollo è detto Stop&Wait o anche **ad alternanza di bit** dato che i numeri di sequenza dei pacchetti si alternano tra 0 e 1.



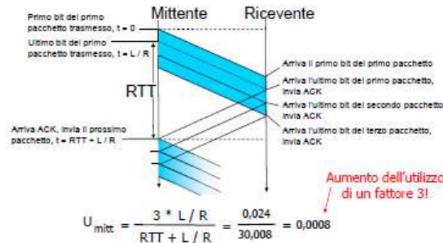
TRASFERIMENTO DATI AFFIDABILE CON PIPELINE

Il protocollo mostrato precedentemente è corretto dal punto di vista funzionale, ma non ottimo dal punto di vista prestazionale proprio per il fatto che si tratta di un protocollo **stop-and-wait**.

Se definiamo l'utilizzo del canale come la frazione di tempo in cui il mittente è stato effettivamente occupato nell'invio di bit sul canale, un'analisi delle prestazioni mostra che il protocollo presenta effettivamente un throughput molto basso.



Una soluzione a questo problema è quello di consentire al mittente di inviare più pacchetti senza attendere gli acknowledgement.



Se si consente al mittente di trasmettere, ad esempio, tre pacchetti senza aspettare gli ACK, l'utilizzo viene sostanzialmente triplicato. Dato che molti pacchetti in transito dal mittente al destinatario possono essere visualizzati come il riempimento di una tubatura, questa tecnica è nota come **pipelining**.

Le conseguenze su un protocollo di trasferimento dati affidabile sono le seguenti:

- L'intervallo di numeri di sequenza disponibili deve essere incrementato, dato che ogni pacchetto in transito deve presentare un numero di sequenza e univoco e che ci potrebbero essere più pacchetti in transito ancora in attesa di acknowledgement.
- I lati di invio e ricezione dei protocolli possono dover memorizzare in un buffer più di un pacchetto e in particolare quei pacchetti trasmessi, ma il cui acknowledgement non è ancora stato ricevuto.

Si possono identificare due approcci di base verso la risoluzione degli errori con pipeline: **Go-Back-N** e **Selective Repeat**.

Go-Back-N

In un protocollo di questo tipo il mittente può trasmettere più pacchetti senza dover attendere alcun acknowledgement, ma non può avere più di un dato numero massimo consentito N di pacchetti in attesa di acknowledgement nella pipeline.

Se definiamo **base** come il numero di sequenza del pacchetto più vecchio che non ha ancora ricevuto un acknowledgement e **nextseqnum** il più piccolo numero di sequenza inutilizzato, allora si possono identificare quattro intervalli di numeri di sequenza.



- Il primo gruppo corrisponde ai pacchetti già trasmessi e che hanno ricevuto acknowledgement.
- Il secondo intervallo corrisponde ai pacchetti inviati, ma che non hanno ancora ricevuto alcun acknowledgement.
- I numeri di sequenza del terzo intervallo possono essere per i pacchetti da inviare immediatamente, nel caso arrivassero dati dal livello superiore.
- I numeri di sequenza maggiori o uguale a $base+N$ non possono essere utilizzati finché il mittente non riceva un acknowledgement relativo a un pacchetto che si trova nella pipeline ed è ancora privo di riscontro.

L'intervallo di numeri di sequenza ammissibili per i pacchetti trasmessi, ma che non hanno ancora ricevuto alcun acknowledgement, può essere visto come una **finestra di dimensione N**.

Quando il protocollo è in funzione, questa finestra trasla lungo lo spazio dei numeri di sequenza.

Il principale motivo per il quale è necessario imporre un limite al mittente sul numero i pacchetti in sospeso N è il controllo di flusso.

Il mittente GBN deve rispondere a tre tipi di evento:

- **Invocazione dall'alto:** quando dall'alto si chiama `send()`, come prima cosa il mittente controlla se la finestra sia piena, ossia se vi siano N pacchetti in sospeso senza acknowledgement. Se la finestra non è piena, crea e invia un pacchetto e le variabili vengono aggiornate di conseguenza. Se la finestra è piena, il mittente restituisce i dati al livello superiore che, presumibilmente, ritenterà più tardi. In una reale implementazione è più probabile che il mittente mantenga questo dato nel buffer o implementi un meccanismo di sincronizzazione (semaforo o flag) che consenta al livello superiore di invocare `send()` solo quando la finestra non sia piena.
- **Ricezione di un ACK:** nel protocollo GBN, l'acknowledgement del pacchetto con il numero di sequenza n verrà considerato un **acknowledgement cumulativo**, che indica che tutti i pacchetti con un numero di sequenza minore o uguale a n sono stati correttamente ricevuti dal destinatario.
- **Evento del timeout:** In presenza di pacchetti persi o eccessivamente in ritardo, come nei protocolli stop-and-wait, si usa ancora un contatore per risolvere il problema. Quando si verifica un timeout, il mittente invia nuovamente tutti i pacchetti spediti che ancora non hanno ricevuto acknowledgement. Se si riceve un ACK, ma ci sono ancora pacchetti aggiuntivi trasmessi e non riscontrati, il timer viene fatto ripartire. Se, invece, non ci sono pacchetti in sospeso in attesa di acknowledgement, il contatore viene fermato.

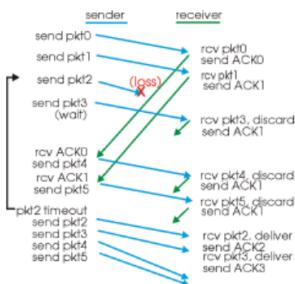
Le azioni del destinatario:

- Se un pacchetto con numero di sequenza n viene ricevuto correttamente ed è in ordine il destinatario manda un ACK per quel pacchetto e consegna i suoi dati al livello superiore.
 - In tutti gli altri casi, il destinatario scarta i pacchetti e rimanda un ACK per il pacchetto in ordine ricevuto più di recente.

Essendo i pacchetti consegnati uno alla volta al livello superiore, se il pacchetto k è stato ricevuto e consegnato, tutti i pacchetti con un numero di sequenza inferiore di k sono anch'essi consegnati.

Si potrebbe pensare di storare in un buffer i pacchetti arrivati fuori sequenza, per poi trasferirli riordinati al livello superiore. Il motivo per cui il destinatario scarta questi pacchetti, anche se questi sono stati ricevuti correttamente è che se al tempo t_0 è stato ricevuto il pacchetto $n+1$ ma il pacchetto n va perduto, sia quest'ultimo, sia il pacchetto $n+1$ verranno ritrasmessi.

La conseguenza è che se il mittente deve mantenere i limiti superiore e inferiore della propria finestra e la posizione nextseqnum all'interno di tale finestra, il destinatario deve mantenere solo il numero di sequenza del successivo pacchetto nell'ordine (salvato nella variabile **expectedseqnum**)



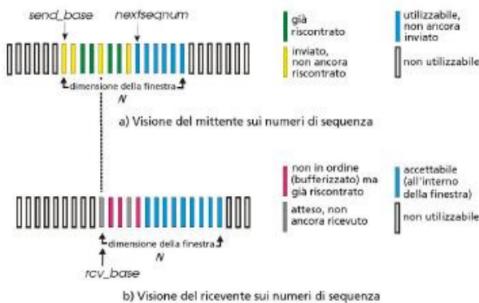
Esempio: la figura mostra come opera un protocollo GBN con una finestra di 4 pacchetti. Il mittente invia i pacchetti da 0 a 3, ma poi deve attendere la notifica di ricezione di uno di loro prima di poter procedere. Quando giungono i successivi ACK (ACK0 e ACK1), la finestra scorre in avanti e il mittente può trasmettere un nuovo pacchetto (pkt4 e pkt5). Dal lato ricevente il pacchetto 2 viene perso e pertanto i pacchetti 3,4 e 5 non rispettano l'ordine e sono scartati.

Discussione: potenzialmente il protocollo GBN consente al mittente di riempire il canale del trasporto con i pacchetti, evitando così i problemi di scarso utilizzo del canale riscontrati nei protocolli stop-and-wait. Esistono, tuttavia, scenari in cui lo stesso GBN ha problemi di prestazioni. In particolare quando l'ampiezza della finestra e il prodotto tra larghezza di banda e ritardo sono entrambi grandi, nella pipeline si possono trovare numerosi pacchetti. Un errore su un solo pacchetto può provocare pertanto un elevato numero di ritrasmissioni, in molti casi inutili, che potrebbe saturare la pipeline.

Selective Repeat

I protocolli a ripetizione selettiva (SR) evitano le ritrasmissioni non necessarie facendo ritrasmettere al mittente solo quei pacchetti su cui esistono sospetti di errore. Questa forma di ritrasmissione a richiesta è personalizzata costringe il destinatario a mandare acknowledgement specifici per i pacchetti ricevuti in modo corretto.

Si userà una nuova ampiezza di finestra pari a N per limitare il numero di pacchetti privi di acknowledgement nella pipeline. Tuttavia a differenza di GBN, il mittente avrà già ricevuto gli ACK di qualche pacchetto nella finestra.



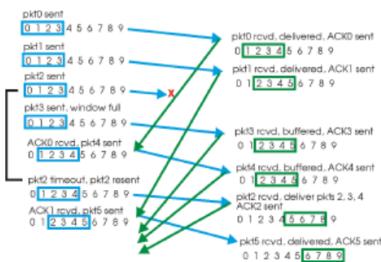
Eventi e azioni di un mittente SR:

- **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 vere un proprio timer logico, dato che al timeout sarà ritrasmesso un solo pacchetto. Si può realizzare un solo contatore hardware per simulare le operazioni di più timer logici.
- **ACK ricevuto:** se si riceve un ACK, il mittente SR etichetta tale pacchetto come ricevuto, ammesso che sia nella finestra. Se il numero di sequenza del pacchetto è uguale a **send_base**, la base della finestra si muove verso il pacchetto che non ha ricevuto acknowledgement con il più piccolo numero di sequenza. Se la finestra si sposta e ci sono pacchetti non trasmessi con il numero di sequenza che ora cade all'interno della finestra, questi vengono ritrasmessi.

Il destinatario SR invia un riscontro per i pacchetti correttamente ricevuti sia in ordine sia fuori sequenza. Questi vengono memorizzati in un buffer finché non sono stati ricevuti tutti i pacchetti mancanti (ossia quelli con i numeri di sequenza più bassi), momento in cui un blocco di pacchetti può essere trasportato in ordine al livello superiore.

Eventi e azioni di un destinatario SR:

- Il pacchetto con un 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 ACK selettivo. Se il pacchetto non era già stato ricevuto viene inserito nel buffer. Se presenta un numero di sequenza uguale alla base della finestra di ricezione (rcv_base) allora questo pacchetto e tutti i pacchetti nel buffer aventi numeri consecutivi 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 già ha riscontrato.
- Altrimenti si ignora il pacchetto.



Esempio: la figura mostra un esempio di comportamento SR in presenza di pacchetti persi. Inizialmente il destinatario memorizza i pacchetti 3,4,5 nel buffer e li consegna al livello superiore insieme al pacchetto 2 quando quest'ultimo viene finalmente ricevuto.

Discussione: è importante notare che il destinatario, come descritto sopra, spedisce un acknowledgement (anziché ignorare) per i pacchetti già ricevuti con certi numeri di sequenza al di sotto dell'attuale base della finestra. Questa nuova notifica è necessaria considerando l'ambito dei numeri di sequenza del mittente e del destinatario. Per esempio tra destinatario e mittente non si propaga un ACK per il pacchetto con numero di sequenza $send_base$, il mittente ritrasmetterlo anche se il destinatario lo ha già ricevuto. Se il destinatario non invia un acknowledgement per questo pacchetto, la finestra del mittente non può avanzare. Non sempre mittente e destinatario hanno la stessa visuale su che cosa sia stato ricevuto correttamente

Manca esempio pag 214-217, ultima slide LivelloTrasporto2

Riepilogo dei meccanismi di trasferimento dati affidabile e loro utilizzo

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 smarritimenti (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 di rilevare pacchetti duplicati.
Acknowledgement positivo (ACK)	Utilizzato dal destinatario per comunicare al mittente che un pacchetto o un insieme di pacchetti sono stati ricevuti correttamente. Gli acknowledgement trasporteranno generalmente i numeri di sequenza dei pacchetti da confermare. A seconda del protocollo, i riscontri possono essere individuali o cumulativi.
Acknowledgement negativo (NAK)	Usato dal destinatario per comunicare al mittente che un pacchetto non è stato ricevuto correttamente. Gli acknowledgement 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 e non aver ancora ricevuto acknowledgement si può migliorare l'utilizzo del canale rispetto alla modalità operativa stop-and-wait. L'ampiezza della finestra può essere impostata sulla base della capacità del destinatario di ricevere e memorizzare messaggi in un buffer, su quella del livello di congestione della rete, o su entrambe.

TRASPORTO ORIENTATO ALLA CONNESSIONE: PROTOCOLLO TCP

Le caratteristiche principali del protocollo TCP sono:

- **Connection-oriented:** TCP è un protocollo orientato alla connessione in quanto, prima di effettuare lo scambio dei dati, i processi devono effettuare un **three-way handshake**. È detto a tre vie perché i processi devono scambiarsi tre segmenti al fine di instaurare la connessione. Come parte dell'instaurazione della connessione TCP, entrambe le parti inizializzano molte variabili di stato associate alla connessione;
- **Full duplex data:** su una stessa connessione TCP tra due processi residenti su due host diversi, il flusso di dati è bidirezionale, cioè può fluire contemporaneamente nelle due direzioni.
- **End-to-end:** si tratta di una connessione end-to-end tra due host, in quanto lo stato della connessione risiede completamente nei due sistemi periferici, per cui gli elementi intermedi di rete sono completamente ignari della connessione;
- **Punto a punto:** ha luogo tra un singolo mittente e un singolo destinatario (non è realizzabile il multicast);
- **Senza errori, sequenza ordinata**
- **Pipelined:** controllo di lusso e di congestione impostano la TCP window
- **Controllo di flusso:** il mittente non invia più di quanto il ricevente non possa accettare;
- **Buffer su mittente e ricevente:** TCP dirige i dati al **buffer di invio** della connessione, uno dei buffer riservato durante l'handshake a tre via, da cui di tanto in tanto preleverà blocchi di dati e li passerà al livello di rete.



La massima quantità di dati prelevabili e posizionabili in un segmento viene limitata dalla **dimensione massima di segmento** (MSS, maximum segment size). Questo valore viene generalmente impostato determinando prima la lunghezza del frame più grande che può essere inviato a livello di collegamento dall'host mittente locale, la cosiddetta **unità trasmisiva massima** (MTU) e poi scegliendo un MSS tale che il segmento TCP (una volta incapsulato in un datagramma IP) stia all'interno di un singolo frame a livello di collegamento, considerando anche la lunghezza dell'intestazione TCP/IP normalmente pari a 40 byte.

OSS. L'MSS rappresenta la massima quantità di dati a livello applicazione nel segmento e non la massima dimensione del segmento TCP con intestazioni incluse.

OSS. I protocolli Ethernet e PPP hanno un MTU di 1500 byte per cui un valore tipico di MSS è di 1460 byte.

OSS. Sono stati proposti approcci per scoprire la MTU più grande lungo il percorso tra sorgente e destinazione per impostare MSS sulla base di quel valore.

Struttura dei segmenti TCP

La Protocol Data Unit (PDU) di TCP è detta **segmento**. Ciascun segmento viene normalmente imbustato in un pacchetto IP, ed è costituito da un header TCP e da un payload, ovvero dati a livello applicativo. Un segmento TCP è così strutturato:



- **numeri di porta di origine e destinazione**, 16 bit utilizzati per il multiplexing/demultiplexing
- il **campo numero di sequenza** di 32 bit che indica lo scostamento (in byte) dell'inizio del segmento TCP all'interno del flusso completo. È riferito allo stream che fluisce nella medesima direzione del segmento.
- Il **campo numero di acknowledgement**: di 32 bit ed ha significato solo se il flag ACK è impostato a 1 o se i segmenti utilizzano la tecnica trasmisiva **PiggyBacking**. Conferma la ricezione di una parte del flusso di dati nella direzione opposta, indicando il valore del prossimo numero di sequenza che l'host mittente del segmento TCP si aspetta di ricevere. Fa riferimento dunque allo stream dati che fluisce nella direzione opposta a tale segmento.
- Il **campo finestra di ricezione** di 16 bit viene utilizzato per il controllo di flusso e in particolare specifica la dimensione del buffer che il TCP ha a disposizione per la gestione dinamica della dimensione della finestra scorrevole.
- Il **campo lunghezza dell'intestazione**, di 4 bit, specifica la lunghezza dell'intestazione TCP in multipli di 32 bit. Generalmente il campo opzioni è vuoto e la lunghezza è quindi 20 byte.
- il **campo opzioni** facoltativo e di lunghezza variabile, utilizzato quando mittente e destinatario negoziano la dimensione massima del segmento o come fattore di scala per la finestra nelle reti ad alta velocità.
- Il **campo flag** è di 6 bit.
 - bit **ACK** viene usato per indicare che il valore trasportato nel campo di acknowledgement è valido; ossia il segmento in questione contiene un acknowledgement per un segmento che è stato ricevuto con successo;
 - il bit **RST** se impostato a 1 indica che la connessione non è valida; viene utilizzato in caso di grave errore, a volte insieme al flag ACK per la chiusura di una connessione;
 - il bit **SYN** se impostato a 1 indica che l'host mittente del segmento vuole aprire una connessione TCP con l'host destinatario; ha lo scopo di sincronizzare i numeri di sequenza

- di due host. L'host che ha inviato il SYN deve attendere dall'host remoto un pacchetto SYN/ACK.
- il bit **FIN**: se impostato 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. 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 ancora disponibile. Quando anche l'altro host invierà il pacchetto con FIN 1, la connessione (dopo il relativo FIN/ACK) sarà completamente chiusa.
 - il bit **PSH** se ha valore 1, indica che i dati in arrivo non devono essere bufferizzati ma passati subito ai livelli superiori
 - il bit **URG** indica che nel segmento sono presenti dati che il mittente a livello superiore ha marcato come "urgenti".
 - **Puntatore a dati urgenti**: indica lo scostamento in byte a partire dal Sequence number dei dati urgenti all'interno del flusso. Validò se il bit URG è settato a 1;
 - **Checksum**: campo di 16 di controllo utilizzato per la verifica della validità del segmento. Questa informazione è di essenziale importanza perché il protocollo IP non prevede nessun controllo di errore sulla parte dati del frame.

Checksum

Per il calcolo del valore checksum il TCP ha bisogno di aggiungere una pseudo intestazione al datagramma, per effettuare così un controllo anche sugli indirizzi IP di destinazione e provenienza. La pseudo intestazione viene creata e posta in testa al datagramma TCP. Viene inserito in essa un ulteriore byte di zeri per raggiungere un multiplo di 16. Successivamente viene calcolata la checksum su tutto il messaggio così formato, viene scartata la pseudo intestazione e passato il datagramma al livello IP.

In fase di ricezione, il livello TCP ricrea la pseudo intestazione interagendo con l'IP sottostante, calcola la checksum e verifica la correttezza del messaggio ricevuto. In caso di errore il datagramma verrà scartato (e quindi ritrasmesso dal mittente).

0	8	16	31
Indirizzo IP provenienza			
Indirizzo IP destinazione			
Zero	Protocollo	Lunghezza TCP	

Numeri di sequenza e numeri di acknowledgement

TCP vede i dati come un flusso di byte non strutturati, ma ordinati. I numeri di sequenza riflettono questa visione: il **numero di sequenza per un segmento** è quindi il numero nel flusso di byte del primo byte nel segmento.

Ad esempio se abbiamo un flusso lungo 500.000 byte, MSS uguale a 1000 byte:

- ➔ Il primo byte sarà numerato con 0
- ➔ TCP costruisce 500 segmenti, con numeri di sequenza 0, 1000, 2000...

Per i **numeri di acknowledgement**, vista la natura full-duplex della connessione TCP, si ha che ad esempio:

- ➔ A invia e contemporaneamente riceve da B
- ➔ I segmenti B->A contengono un numero di sequenza relativo ai dati B->A
- ➔ Il numero di acknowledgement che A scrive nei propri segmenti è il numero di sequenza del byte successivo che A attende da B (e lo può mandare anche inviando dati a B).

Dato che TCP effettua l'acknowledgement solo dei byte fino al primo byte mancante nel flusso, si dice che tale protocollo offre **acknowledgement cumulativi**. Si ha il vantaggio che la perdita di un riscontro non blocca la trasmissione se confermato da un acknowledgement successivo.

Numeri di sequenza:

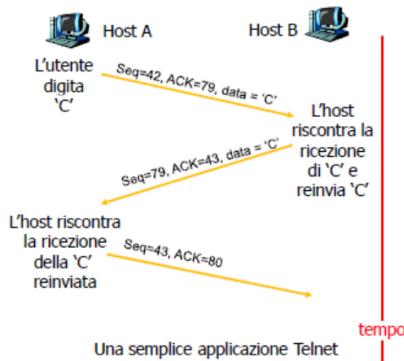
- "numero" del primo byte del segmento nel flusso di byte

ACK:

- numero di sequenza del prossimo byte atteso dall'altro lato
- ACK cumulativo

D: come gestisce il destinatario i segmenti fuori sequenza?

- R: la specifica TCP non lo dice – dipende dall'implementazione



Un'osservazione importante da fare è che quando un host riceve segmenti fuori sequenza nella connessione TCP, gli RFC non impongono regole su come si debba gestire la situazione. Si può scegliere di scartare immediatamente i segmenti non ordinati oppure bufferizzare.

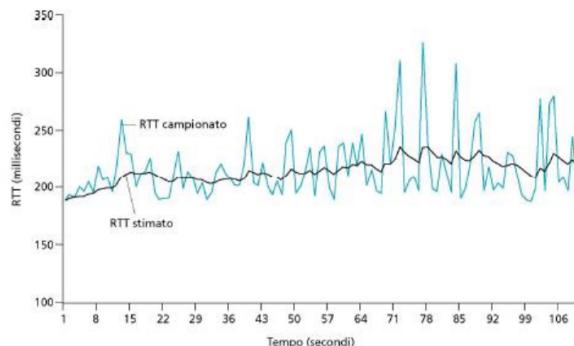
Principali opzioni di TCP

- **Maximum TCP payload:** durante la fase di connessione, ciascun end-point annuncia la massima dimensione di payload che desidera accettare; la minima tra le due dimensione annunciate viene selezionata per la trasmissione;
- **Window Scale:** per negoziare un fattore di scale per la finestra; utile per connessioni a larga banda e/o elevato ritardo di trasmissione;
- **Selective Repeat:** nel caso in cui un segmento corrotto sia stato seguito da segmenti corretti, introduce i NAK per permettere al receiver di richiedere la ritrasmissione di quello specifico segmento; è un'alternativa al "go back n", che prevede la ritrasmissione di tutti i segmenti

Round Trip Time e Timeout di Ritrasmissione

TCP come detto utilizza un meccanismo di timeout e ritrasmissione per recuperare i segmenti persi. La questione problematica da risolvere è la durata degli intervalli di timeout. Il timeout dovrebbe essere più grande del tempo di andata e ritorno sulla connessione (RTT).

Se il timeout scelto è troppo breve o troppo lungo c'è una scarsa efficienza nella gestione delle ritrasmissioni.



L'RTT misurato di un segmento, denotato come **SampleRTT**, è la quantità di tempo che intercorre tra l'istante di invio del segmento e quello di ricezione del relativo acknowledgement.

Anziché misurare un SampleRTT per ogni segmento, la maggior parte delle implementazioni TCP effettua una sola misurazione di SampleRTT alla volta. Ossia in ogni istante di tempo, SampleRTT viene valutato per uno solo dei segmenti trasmessi e per cui non si è ancora ricevuto acknowledgement, il che comporta approssimativamente la misurazione di un nuovo valore di SampleRTT a ogni RTT.

Inoltre, TCP non calcola mai il SampleRTT per i segmenti ritrasmessi, cioè lo calcola soltanto per i segmenti trasmessi una sola volta.

Ovviamente i campioni variano da segmento a segmento in base alla congestione nei router e al diverso carico sui sistemi periferici. Ogni valore di SampleRTT può essere quindi atipico.

Per effettuare una stima naturale viene calcolata una media dei valori di SampleRTT detta **EstimatedRTT**. Quando si ottiene un nuovo SampleRTT, TCP aggiorna EstimatedRTT secondo la formula:

$$\text{EstimatedRTT} = (1 - \alpha) * \text{EstimatedRTT} + \alpha * \text{SampleRTT}$$

EstimatedRTT corrisponde dunque ad una media esponenziale ponderata (EWMA: Exponential Weighted Moving Average) dei campioni. L'influenza di un singolo campione sul valore della stima decresce in maniera esponenziale, e si dà più importanza a campioni recenti. Un valore tipico per α è 0,125

Oltre ad avere una stima di RTT è anche importante possedere la misura della sua variabilità. Si definisce la **varianza di RTT** come una stima di quanto SampleRTT generalmente si discosta da EstimatedRTT:

$$\text{DevRTT} = (1 - \beta) * \text{DevRTT} + \beta * |\text{SampleRTT} - \text{EstimatedRTT}|$$

Si noti che DevRTT è una EWMA della differenza SampleRTT-EstimatedRTT. Se i valori di SampleRTT presentano fluttuazioni limitate, allora DevRTT sarà piccolo. Il valore raccomandato per β è 0,25.

Gestione del Timeout

Dati i valori di EstimatedRTT e DevRTT l'intervallo di timeout non può essere inferiore a quello di EstimatedRTT, altrimenti verrebbero inviate ritrasmissioni non necessarie. Ma l'intervallo stesso non dovrebbe essere molto maggiore di EstimatedRTT, altrimenti TCP non ritrasmetterebbe rapidamente il segmento perduto, il che comporterebbe gravi ritardi sul trasferimento.

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

E cioè il timeout prende in conto anche le fluttuazioni di valori. Se c'è molta fluttuazione deve essere considerato un margine più grande.

TRASFERIMENTO DATI AFFIDABILE

Con il servizio IP i datagrammi possono sovraffollare i buffer dei router e non raggiungere la destinazione o arrivare in ordine casuale e con bit alterati. Dato che sono inglobati nei datagrammi IP, anche i segmenti a livello di trasporto possono risentire di questi problemi. TCP **crea un servizio di trasporto affidabile** al di sopra del servizio inaffidabile e best-effort di IP, assicurando che il flusso di byte che i processi leggono dal buffer di ricezione TCP:

- Non sia alterato
- Non abbia buchi
- Non presenti duplicazioni
- Rispetti la sequenza originaria

Esistono tre eventi principali relativi alla trasmissione e ritrasmissione dei dati;

- **Dati ricevuto dall'applicazione:** TCP incapsula i dati che gli giungono dall'applicazione in un segmento e lo passa a IP. Notiamo che ciascun segmento include un numero di sequenza che rappresenta il numero del primo byte di dati del segmento nel flusso di byte. Inoltre se il timer non è già in funzione per qualche altro segmento TCP lo avvia quando il segmento viene passato a IP. E' utile pensare che il timer sia associato al più vecchio segmento che non ha ricevuto acknowledgement. L'intervallo di scadenza del timer è il **TimeoutInterval** calcolato come descritto in precedenza.
- **Timeout:** TCP risponde ritrasmettendo il segmento che lo ha causato e quindi riavviando il timer.
- **ACK ricevuti:** TCP deve gestire l'arrivo di segmenti di acknowledgement. Quando si verifica tale evento TCP confronta il valore y di ACK con la propria variabile **SendBase**. La variabile di stato TCP SendBase è il numero di sequenza del più vecchio byte che non ha ancora ricevuto un acknowledgement. Di conseguenza SendBase-1 è il numero di sequenza dell'ultimo byte che si sa essere stato ricevuto correttamente e nell'ordine giusto. TCP utilizza acknowledgement cumulativi pertanto y conferma la ricezione di tutti i byte precedenti al byte numero y . Se y è maggiore di SendBase, allora l'ACK si riferisce a uno o più segmenti che precedentemente non avevano ancora ricevuto conferma. Quindi il mittente aggiorna la propria variabile SendBase e poi se non ci sono segmenti che ancora necessitano di acknowledgement riavvia il timer.

Di seguito l'algoritmo di un sender TCP semplificato:

/ Si è assunto che il sender non sia limitato dal controllo di flusso o di congestione del TCP, che i dati da spedire siano di dimensioni inferiori all'MSS e che il trasferimento dei dati avvenga in una sola direzione. */*

```

NextSegNum=InitialSegNumber
SendBase=InitialSegNumber

loop (forever) {
    switch(event) {

        event: data received from application above
            create TCP segment with sequence number NextSegNum
            if (timer currently not running)
                start timer
            pass segment to IP
            NextSegNum=NextSegNum+length(data)
            break;

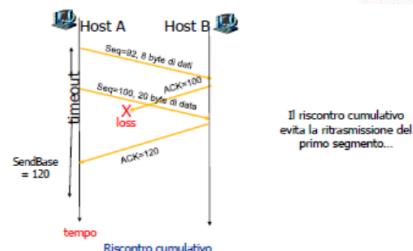
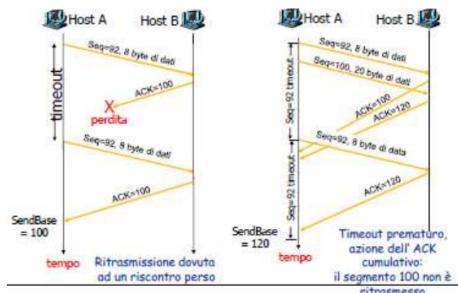
        event: timer timeout
            retransmit not-yet-acknowledged segment with
                smallest sequence number
            start timer
            break;

        event: ACK received, with ACK field value of y
            if (y > SendBase) {
                SendBase=y
                if (there are currently any not-yet-acknowledged
                    segments)
                    start timer
            }
            break;

    } /* end of loop forever */
}

```

Alcuni scenari interessanti sono mostrati qui di seguito:



Raddoppio dell'intervallo di timeout

Questa modifica nell'implementazione di TCP prevede che allo scadere di un timeout si imposti il prossimo intervallo al doppio del valore precedente (invece di usare la stima di RTT). Si ha dunque una crescita esponenziale degli intervalli dopo ogni ritrasmissione.

Quando il timer viene riavviato (alla ricezione di un ACK o di nuovi dati dall'applicazione) l'intervallo di timeout viene nuovamente configurato in funzione dei valori più recenti di EstimatedRTT e DevRTT.

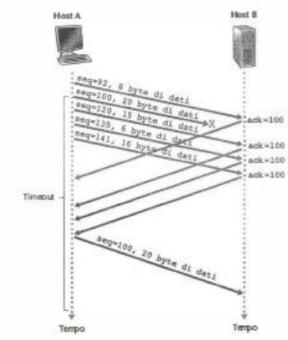
Questa modifica offre una forma limitata di controllo di congestione. La scadenza del timer viene probabilmente causata dalla congestione nella rete, che provoca la perdita di pacchetti. Nei periodi di congestione se le sorgenti continuano a ritrasmettere pacchetti, la congestione può peggiorare. Ciascun mittente quindi ritrasmette ad intervalli sempre più lunghi.

Ritrasmissione Rapida

Uno dei problemi legati alla ritrasmissione è che il periodo di timeout può rivelarsi lungo. Il mittente può in molti casi rilevare la perdita dei pacchetti ben prima che si verifichi l'evento di timeout grazie agli **ACK duplicati** relativi a un segmento il cui ACK è già stato ricevuto dal mittente.

Quando un destinatario TCP riceve un segmento con numero di sequenza superiore a quello atteso in ordine, rileva un buco nel flusso di dati, ossia un segmento mancante. Tale vuoto potrebbe essere il risultato di segmenti persi. Il destinatario non può inviare un acknowledgement negativo esplicito al mittente, dato che TCP non lo prevede, ma si limita a mandare nuovamente un acknowledgement relativo all'ultimo byte di dati che ha ricevuto in ordine (duplicando così un ACK).

Poiché il mittente manda spesso molti segmenti contigui, se uno di tali segmenti si perde ci saranno molti ACK duplicati contigui. Se il mittente riceve **tre ACK duplicati** per lo stesso dato, assume che il segmento successivo a quello riscontrato tre volte è andato perso. In questo caso (dopo 3 ACK) TCP effettua una **ritrasmissione rapida**, rispedendo il segmento mancante prima che scada il timer.



Raccomandazioni sulla generazione degli acknowledgement

Evento	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 (che è l'estremità inferiore del buco).
Arrivo di un segmento che colma parzialmente o completamente il buco nei dati ricevuti.	Invia immediatamente un ACK, ammesso che il segmento cominci all'estremità inferiore del buco.

APERTURA DELLA CONNESSIONE TCP

La procedura utilizzata per instaurare in modo affidabile una **connessione TCP** tra due host è chiamata **three-way handshake**, indicando la necessità di scambiare 3 messaggi tra host mittente e host ricevente affinché la connessione sia instaurata correttamente.

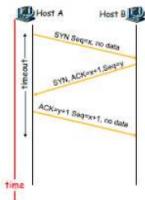
1. **A invia un segmento SYN a B** – il flag SYN è impostato a 1 e il campo Sequence number contiene il valore x casuale che specifica l'Initial Sequence number di A (ossia il **client_isn**)
2. **B invia un segmento SYN/ACK ad A** – i flag SYN e ACK sono impostati a 1, il campo Sequence number contiene il valore y che specifica l'Initial Sequence number di B (**server_isn**) e il campo Acknowledgement number contiene il valore $x+1$ (**client_isn+1**) confermando la ricezione del ISN di A.
3. **A invia un segmento ACK a B** – il flag ACK è impostato a 1 e il campo Acknowledgement number contiene il valore $y+1$ (**server_isn+1**) confermando la ricezione del ISN di B.

Il terzo segmento è necessario al fine di permettere anche all'host B una stima del timeout iniziale, come tempo intercorso tra l'invio di un segmento e la ricezione del corrispondente ACK.

Nel traffico TCP i segmenti SYN stabiliscono **nuove connessioni**, mentre quelli con il flag non attivo appartengono a connessioni già instaurate.

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

Il three-way handshake si rende necessario poiché la sequenza numerica ISN non è legata ad un clock generale della rete, inoltre ogni pacchetto IP può avere il proprio modo di calcolare l'Initial number.



CHIUSURA DELLA CONNESSIONE TCP

Dopo che è stata stabilita, una connessione TCP non è considerata una singola connessione bidirezionale, ma piuttosto come l'interazione di due connessioni monodirezionali.

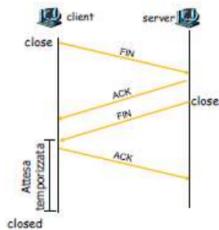
Ognuna delle parti deve terminare la sua connessione, e possono esistere anche connessioni aperte a metà, in cui solo uno dei due terminali ha chiuso la connessione e non può più trasmettere, ma può (e deve) ricevere i dati dall'altro terminale.

Se le due parti chiudono contemporaneamente le rispettive connessioni viene utilizzato un'**handshake a 3 vie** è omologo a quello usato per l'apertura della connessione con la differenza che il flag utilizzato è il FIN invece del SYN.

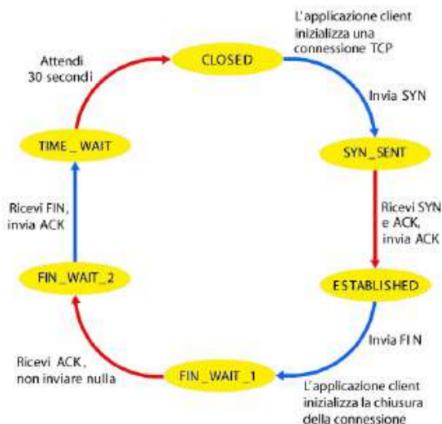
1. Un terminale invia un pacchetto con la richiesta FIN
2. L'altro risponde con un FIN/ACK
3. Il primo manda un ultimo ACK e l'intera connessione viene terminata

L'**handshake a 4 vie** invece viene utilizzato quando la disconnessione non è contemporanea tra i due terminali in comunicazione. In questo caso:

1. Uno dei due terminali invia la richiesta di FIN
2. L'altro risponde con un ACK
3. Il secondo terminale, come il primo invia la richiesta di FIN
4. Il primo risponde con un ACK.

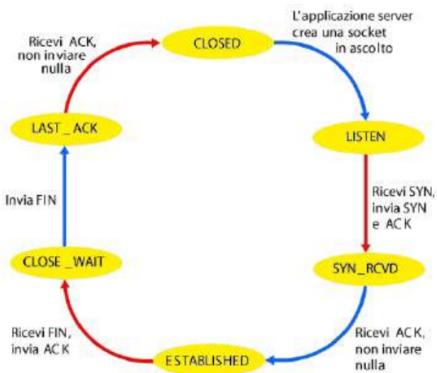


Tipicamente nell'arco di una **connessione TCP**, i protocolli TCP in esecuzione negli host attraversano vari stati TCP che cambiano a seconda del che l'host sia un client o un server.



In figura è rappresentato il diagramma a stati di una connessione TCP nel **client**:

1. il client parte dallo stato **CLOSED**;
2. L'applicazione sul lato client inizializza una nuova connessione TCP (creando una socket);
3. Questo spinge il TCP nel client a inviare il segmento SYN al TCP nel server
4. Una volta spedito il SYN, il client TCP entra nello stato **SYN_SENT**, durante il quale attende dal server TCP un segmento con un acknowledgement per un precedente segmento del client e che abbia il bit SYN posto a 1;
5. Una volta ricevuto tale segmento, il client TCP entra nello stato **ESTABLISHED**, durante il quale può inviare e ricevere segmenti che contengono dati utili (ossia generati dall'applicazione);
6. Se l'applicazione volesse chiudere la connessione il client TPP invia un segmento con il bit FIN impostato a 1 ed entra nello stato **FIN_WAIT_1**. Il client in questo stato attende dal server un segmento TCP con un acknowledgement;
7. Quando lo riceve entra nello stato **FIN_WAIT_2**, in cui il client attende un altro segmento dal server con il bit FIN impostato ad 1.
8. Dopo aver ricevuto questo segmento, il client TCP invia un acknowledgement ed entra nello stato **TIME_WAIT**, che consente al client TCP di inviare nuovamente l'ultimo acknowledgement nel caso in cui l'**ACK** vada perduto.
9. Dopo l'attesa, la connessione viene formalmente chiusa e tutte le risorse sul lato client vengono rilasciate.



In questa seconda figura, invece, sono mostrati gli stati tipici visti dal lato server TCP:

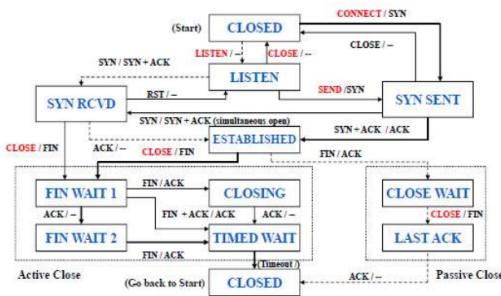
1. Il server parte dallo stato **CLOSED**;
2. L'applicazione sul lato server inizializza una nuova connessione creando una socket in ascolto transitando dunque nello stato **LISTEN**;
3. Il server riceve il SYN dal client che vuole instaurare una connessione, e risponde con un segmento di SYN/ACK.
4. Questo spinge il server nello stato **SYN_RCVD**.
5. Il server attende un ultimo ACK dal client prima di entrare nello stato **ESTABLISHED**.
6. Se in questo punto, l'applicazione client vuole chiudere la connessione, il server riceve un FIN, al quale risponde con un ACK nel caso in cui voglia anch'esso chiudere la connessione.
7. Questo fa transitare il server nello stato **CLOSE_WAIT**.
8. Quando il server invia anch'esso un segmento con il bit FIN a 1 si porta immediatamente nello stato **LAST_ACK**.
9. Quando arriva su questo stato, il server attende un ACK da parte del client.
10. Se ricevuto, il server può transitare nello stato **CLOSED** e chiudere finalmente la connessione.

In questa trattazione è stato assunto che client e server siano entrambi preparati alla comunicazione, ossia che il server sia in ascolto sulla porta alla quale il client invierà il segmento SYN. In generale non è detto che il server abbia una socket aperta in ascolto su quel determinato porto.

Lo strumento per scansionare lo stato delle porte è **nmap**. Per analizzare una specifica porta TCP, su un host bersaglio, nmap manderà a quell'host un segmento TCP SYN con porta di destinazione quella da testare. Possono verificarsi tre diverse situazioni:

- **L'host sorgente riceve un segmento TCPSYNACKN dal bersaglio.** Questo significa che un'applicazione è in esecuzione con quella porta TCP sul bersaglio. nmap restituisce “**open**”.
- **L'host sorgente riceve un segmento TCP RST dal bersaglio.** Significa che il segmento SYN ha raggiunto l'host bersaglio, ma su quest'ultimo non è in esecuzione alcuna applicazione che usa quella porta.
- **La sorgente non riceve nulla.** Questo probabilmente significa che il segmento SYN è stato bloccato da un firewall e non ha mai raggiunto l'host bersaglio.

Diagramma degli stati del TCP



CONTROLLO DI FLUSSO

L'affidabilità della comunicazione in TCP è garantita anche dal cosiddetto **controllo di flusso** ovvero far in modo che il flusso di dati in trasmissione non superi le capacità di ricezione ovvero di memorizzazione del ricevente con perdita di pacchetti e maggior peso e latenze nelle successive richieste di ritrasmissione.

Pertanto il controllo di flusso è un servizio di confronto sulle velocità, dato che paragona la frequenza di invio del mittente con quella di lettura dell'applicazione del ricevente.

TCP implementa il controllo di flusso facendo mantenere al mittente una variabile chiamata **finestra di ricezione** (receive window) che, in sostanza, 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.

Supponiamo che l'host A stia inviando un file di grandi dimensioni all'host B su una connessione TCP. Quest'ultimo alloca un buffer di ricezione per la connessione. Di tanto in tanto il processo applicativo nell'host B legge dal buffer. Sono definite alcune variabili:

- La dimensione del buffer è denotata come **RcvBuffer**;
 - **LastByteRead** è il numero dell'ultimo byte nel flusso di dati che il processo applicativo B ha letto dal buffer;
 - **LastByteRcvd** è il numero dell'ultimo byte, nel flusso di dati, che proviene dalla ricezione e che è stato copiato nel buffer di ricezione di B.

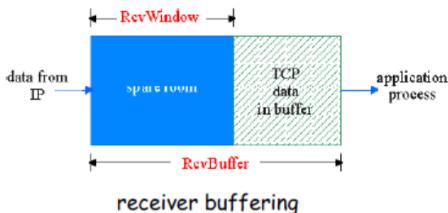
Dato che TCP può mandare in overflow il buffer allocato, si dovrebbe avere per forza:

LastByteRcvd – LastByteRead = RcvBuffer

La finestra di ricezione, indicata con **rwnd**, viene impostata alla quantità di spazio disponibile nel buffer:

$$\text{rwnd} = \text{RcvBuffer} - [\text{LastByteRcvd} - \text{LastByteRead}]$$

Dato che lo spazio disponibile è variabile nel tempo, rwnd è dinamica.



L'host B (ricevente) comunica dinamicamente all'host A (mittente) quanto spazio disponibile sia presente nel buffer della connessione, scrivendo il valore corrente di rwnd nel campo **RcvWindow** dei segmenti che manda ad A.

A sua volta l'host A (mittente) conserva due variabili, **LastByteSent** e **LastByteAcked**, e cioè rispettivamente l'ultimo byte mandato e l'ultimo byte per cui si è ricevuto acknowledgement. Si noti che la differenza tra i valori di queste due variabili esprime la quantità di dati spediti da A per cui non si è ancora ricevuto un acknowledgement.

LastByteSent – LastByteAcked <= RcvWindow

Mantenendo la quantità di dati senza acknowledgement sotto il valore di rwnd si garantisce che l'host A non mandi in overflow il buffer di ricezione dell'host B.

Discussione

In questo schema descritto esiste tuttavia un problema tecnico. Supponiamo che il buffer di ricezione dell'host B si riempia, in modo che $rwnd = 0$, e che dopo averlo notificato all'host A, non abbia più nulla da inviare ad A.

Quando il processo applicativo in B svuota il buffer, TCP non invia nuovi segmenti con nuovi valori di rwnd; infatti TCP fa pervenire un segmento all'host A solo se ha dati o un acknowledgement da mandare. Di conseguenza A non viene informato del fatto che si sia liberato un po' di spazio nel buffer di ricezione.

Per risolvere questo problema, le specifiche TCP richiedono che l'host A continui a inviare segmenti con un byte di dati quando la finestra di ricezione di B è zero. Il destinatario scarterà questi segmenti e risponderà a con un acknowledgement. Quando il buffer inizierà a svuotarsi, i riscontri conteranno un valore non nullo per rwnd.

Silly window syndrome

Un processo di scrittura molto lento da parte del mittente nel buffer di trasmissione (o di lettura da parte del ricevente) porta infatti all'invio di segmenti di dati molto piccoli, aumentando così il rapporto tra header e dati con un conseguente uso inefficiente del canale.

Sindrome causata dal mittente

Nel caso in cui il processo di scrittura del mittente sia molto lento, il protocollo spedirà una serie di pacchetti contenenti una quantità di dati molto bassa, con uso inefficiente del canale.

Un'ottima soluzione è fornita dall'**algoritmo di Nagle**, secondo il quale i dati devono essere accumulati nel buffer per poi venire spediti in un unico blocco alla ricezione dell'ACK dell'ultimo pacchetto trasmesso o quando si raggiunge la massima dimensione fissata per un segmento (MSS).

Questo algoritmo tiene conto anche della velocità di trasmissione dei pacchetti:

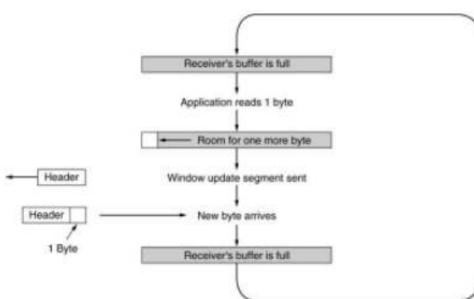
- Se questa è più lenta della scrittura dei messaggi (il mittente riesce ad accumulare molti dati nel buffer prima dell'arrivo del riscontro) vengono creati dei pacchetti con il massimo rapporto dati/header, sfruttando al meglio le risorse del canale.
- Se invece la rete è più veloce, i pacchetti risulteranno più piccoli, e verrà garantito comunque un utilizzo più efficiente delle risorse del canale che nel caso in cui l'algoritmo non venga utilizzato.

Sindrome causata dal ricevente

Nel caso sia il ricevente a leggere lentamente i pacchetti ricevuti, il buffer in ingresso tenderà a riempirsi, costringendo a richiedere al mittente di interrompere la trasmissione. Non appena una trama viene letta il mittente verrà informato dal riscontro che si è liberato dello spazio e reagisce inviando un nuovo segmento.

Viene generato un nuovo pacchetto non appena si libera spazio, dando nuovamente origine a una situazione di uso inefficiente del canale dato dal cattivo rapporto lunghezza pacchetto/dati contenuti.

Una possibilità è adottare la **soluzione di Clark** con la quale si "inganna" il mittente specificando nei messaggi di riscontro che il buffer è ancora pieno (costringendolo a bloccare l'invio) finché la coda non si sia svuotata per metà o a sufficienza per accogliere un segmento di dimensioni massime (MSS).



CONTROLLO DI CONGESTIONE

Vedere esempi di congestione sul libro. Pag 245-250

L'ultimo tipo di controllo effettuato da TCP per garantire affidabilità alla comunicazione è il cosiddetto **controllo della congestione** ovvero una funzionalità di controllo di trasmissione da parte di TCP che permette di limitare la quantità di dati trasmessi sotto forma di pacchetti e non ancora riscontrati dal mittente, adattando il flusso dati inviati all'eventuale stato di congestione della rete.

Si possono distinguere due differenti approcci al controllo della congestione utilizzati nella pratica:

- **Controllo di congestione end-to-end:** l'eventuale stato di congestione della rete è desunto indirettamente, a partire da informazioni ricavabili dallo stato della trasmissione dei pacchetti da parte dei terminali. Il livello di rete non fornisce supporto esplicito al livello di trasporto per il controllo di congestione. L'eventuale stato di congestione deve essere desunto dai sistemi periferici sulla base delle osservazioni del comportamento della rete. Ad esempio la perdita di segmenti TCP (indicata da un timeout o da acknowledgement triplicati) indica congestione di rete.
- **Controllo di congestione assistito dalla rete:** i componenti a livello di rete (router) forniscono un feedback esplicito al mittente sullo stato di congestione della rete. Un singolo bit indica lo stato di congestione. Questo approccio è adottato nelle prime architettura SNA e DECnet. Una forma di controllo più sofisticata, implementata da ATM ABR consente a un router di informare il mittente in modo esplicito sulla frequenza trasmissiva massima che il router può supportare su quel collegamento.

TCP utilizza l'approccio end-to-end ed è quindi coerente con il modello progettuale di IP che non offre ai sistemi periferici un feedback esplicito sulla congestione della rete.

Tuttavia alcune versioni avanzate del protocollo TCP prevedono il secondo approccio, come TPC RED (Random Early Detection) o anche TCP ECN (Explicit Congestion Notification).

Finestra di congestione

Il controllo di congestione del TCP si basa sul fatto che le due entità messe in comunicazione dal TCP si scambiano pacchetti contenenti dati e pacchetti di riscontro (ACK), che viaggiano nel verso opposto rispetto ai dati.

In assenza di perdite di pacchetti, il tempo che intercorre tra l'invio di un segmento (dall'host A all'host B) e la ricezione del relativo riscontro (invia dall'host B all'host A) definisce il cosiddetto RTT. In presenza di congestione della rete, le code nei router diventano più lunghe, e questo fa aumentare il ritardo subito dai pacchetti e dai relativi riscontri, e quindi l'RTT.

Il meccanismo di controllo della congestione implementato in TCP rileva l'approssimarsi della situazione di congestione nelle entità terminali delle connessioni TCP ed impone a ciascun mittente un limite alla velocità di invio dei pacchetti in funzione della congestione di rete percepita. Tale **limite** è rappresentato da un variabile, detta **CongWin**, ovvero finestra di congestione (cwnd).

Il valore di cwnd impone un limite superiore alla quantità di dati trasmessi e non ancora riscontrati dal mittente, ovvero i dati che, essendo stati consegnati per la trasmissione al livello rete, sono in viaggio sulla rete o in fase di elaborazione della entità TCP attiva sul nodo destinazione, o i cui ACK sono a loro volta in viaggio sulla rete stessa.

La quantità di dati che non hanno ancora ricevuto acknowledgement inviata da un mittente non può essere superiore al minimo tra i valori della finestra di congestione e della finestra di ricezione.

LastByteSent – LastByteAcked <= min { cwnd, rwnd }

I due meccanismi (controllo di flusso e controllo di congestione) impongono due distinti limiti superiori alla quantità **LastByteSent – LastByteAcked**. Il più piccolo tra i valori di CongWin e RcvWindow determina quale tra i due meccanismi sia l'effettivo "fattore limitante" del tasso di trasmissione.

In TCP il mittente regola costantemente CongWin a seconda del livello di congestione della rete rilevato, in tal modo modulando il tasso di trasmissione dei pacchetti inviati. Se il mittente rileva che sul percorso c'è assenza di congestione, incrementa il valore di CongWin, viceversa se rileva segnali di congestione, lo riduce.

Un mittente può rilevare la presenza di congestione della rete quando si verificano eventi che sono sintomi della perdita di pacchetti:

- Scadenza di un timeout a causa della mancata ricezione di ACK;
- Ricezione di tre pacchetti di ACK con lo stesso numero di sequenza, che manifestano un "buco" nella sequenza di pacchetti ricevuti.

Supponendo che un host A invii segmenti TCP ad un host B all'inizio di ogni RTT, nell'ipotesi che sia $\text{CongWin} < \text{RcvWindow}$, il vincolo consente al mittente di trasmettere CongWin byte sulla propria connessione; al termine del RTT il mittente riceve il riscontro dei dati. Da ciò si evince che la frequenza di invio del mittente è **CongWin/RTT byte/sec**, quindi il mittente può regolare la frequenza di invio di segmenti sulla propria connessione modificando il valore di CongWin.



Algoritmo di controllo di congestione

L'algoritmo di controllo di congestione presenta due fasi:

1. Partenza lenta (**Slow-Start**)
2. Aumento Additivo Diminuzione Moltiplicativa (**Additive Increase Multiplicative Decrease**)

Per distinguere tra le due fasi viene usata una variabile:

SSTHRESH = cwnd / 2 (ossia metà del valore che aveva la finestra di congestione quando rilevata)

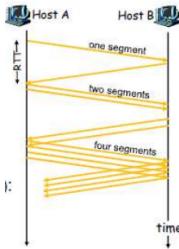
Quando il valore di CongWin è minore di SSTHRESH ci troviamo nella fase di Slow-Start, altrimenti siamo nella fase di AIMD.

La versione Reno di TCP ha inoltre una terza fase detta ripresa veloce (**Fast Recovery**).

Slow Start

Durante la fase di partenza lenta il mittente TCP inizia trasmettendo il primo segmento dati ed attende un riscontro. Ogni volta che un segmento trasmesso viene riscontrato (si riceve un ACK) il mittente incrementa la finestra di congestione CongWin di una quantità pari al MSS. Quindi ad ogni RTT la CongWin raddoppia di dimensioni fino a raggiungere la metà della dimensione massima. Da questo punto in avanti l'aumento procede in maniera lineare fino a raggiungere il massimo.

Questo valore viene mantenuto finché non si incorre in un evento di congestione, come ad esempio la perdita di un segmento dati, oppure quando la dimensione della finestra di congestione supera il valore della variabile SSTHRESH, evento che conduce alla fase di AIMD, si passa cioè all'algoritmo di Congestion Avoidance.



AIMD

Durante la fase di AIMD si ha un incremento additivo lineare del valore di CongWin di 1 MSS ogni RTT (invece di raddoppiare). Ciò si può ottenere attraverso l'incremento da parte del mittente della propria CongWin di una quantità di byte pari a **MSS*(MSS/CongWin)** ogni volta che giunge un nuovo acknowledgement. Tale comportamento viene detto **Additive Increase** o anche **Congestion Avoidance**.

Il termine **Multiplicative Decrease** si riferisce al comportamento del mittente al sopraggiungere della congestione: se dedotto uno stato di congestione è dedotta (scadenza di un timeout o tre ACK duplicati) il valore di CongWin è posto uguale a 1 MSS e il valore di SSTHRESH viene impostato alla metà del valore di cwnd al momento del timeout. Si ricomincia con una nuova partenza lenta (Slow Start)

Fast Recovery

Se la congestione è dedotta dall'evento di ricezione di tre ACK duplicati (anziché scadenza di un timeout) vuol dire che, nonostante si sia perso un pacchetto, almeno 3 segmenti successivi sono stati ricevuti dal destinatario.

Si evita in questo caso una nuova partenza lenta; TCP dimezza il valore di CongWin (aggiungendo 3 MSS per tenere conto dei duplicati consecutivi) e imposta il valore di STHRESH pari a metà del valore di cwnd al momento del ricevimento dei tre ACK duplicati. Si riparte dalla fase di Congestion Avoidance.

Questo meccanismo è detto di **Fast Recovery** ed è implementato, ad esempio dalla versione TCP Reno.

OSS. Nelle versioni TCP in cui non è previsto il Fast Recovery (es. Tahoe) sia per congestione dedotta dalla scadenza di un timeout, che dedotta tramite tre ACK duplicati è prevista la stessa azione: CongWin viene posto uguale a 1 MSS e dunque si riparte con uno Slow-Start.

Il grafico seguente mostra l'evoluzione della finestra di congestione quando si adotta **TCP Tahoe**, che porta in modo incondizionato la finestra di congestione a 1 MSS e in Slow Start dopo qualsiasi evento di perdita, e **TCP Reno** che adotta invece Fast Recovery.

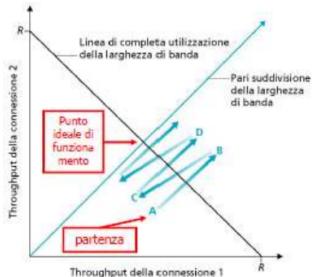


FAIRNESS

Nel caso di una rete composta da un unico collegamento a collo di bottiglia avente banda pari a R bps, condiviso da K connessioni TCP, si dice che un meccanismo di congestione sia **equo** se ciascuna delle K connessioni TCP ottiene a regime una velocità media di trasmissione dei dati pari a R/K . In altre parole ciascuna connessione ottiene la stessa porzione di banda del collegamento

AIMD può essere un algoritmo fair?

Consideriamo due connessioni TCP che condividono un collegamento con capacità R . Assumiamo che abbiano stessi valori di MSS e RTT ed inoltre che entrambe si trovino oltre lo Slow-Start, quindi operino in modalità Congestion Avoidance (AIMD).



In riferimento alla figura, se TCP sta suddividendo la larghezza di banda del collegamento in modo uguale tra le due connessioni, allora il throughput dovrebbe cadere sulla bisettrice del primo quadrante.

Idealmente la somma dei due throughput dovrebbe essere uguale a R . Certamente, non sarebbe piacevole che ciascuna connessione ricevesse la stessa porzione, nulla, della capacità del collegamento.

Quindi l'obiettivo dovrebbe essere ottenere throughput situati vicino all'intersezione tra la bisettrice e la linea di massimo utilizzo della banda.

Si dimostra che sotto le ipotesi definite, la banda utilizzata dalle due connessioni può fluttuare lungo la linea di equa condivisione di banda e che le due connessioni convergeranno a questo comportamento indipendentemente dal punto del piano in cui si trovano inizialmente.

Tuttavia in uno **scenario reale**, in cui il collegamento sia attraversato da connessioni anche non TCP, che non abbiano stesso RTT e in cui ci siano connessioni multiple tra coppie di host le applicazioni client/server otterranno sicuramente porzioni di banda assai diverse.

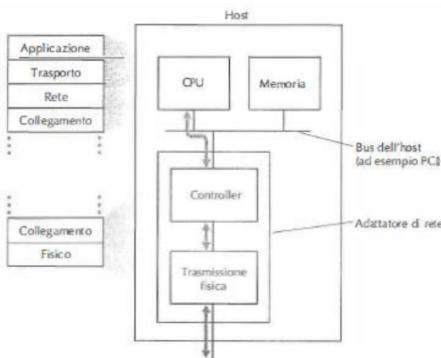
CAPITOLO 6 – LIVELLO DI COLLEGAMENTO

Qualunque dispositivo che opera a livello di collegamento (livello 2) viene indicato come **nodo** e i canali di comunicazione, che collegano nodi adiacenti sono detti **collegamenti**. Su ogni collegamento, un nodo trasmittente incapsula il datagramma in un **frame del livello di collegamento** e lo trasmette lungo il collegamento stesso.

Sebbene il servizio di base del link layer sia il trasporto di datagrammi da un nodo a quello adiacente, il livello di collegamento può includere diversi servizi che possono variare da un protocollo all'altro:

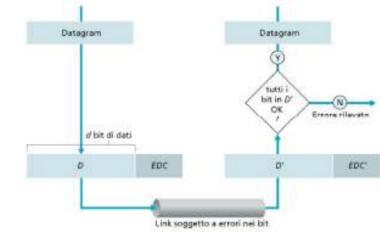
- **Framing:** Incapsulamento dei datagrammi del livello di rete all'interno di frame a livello di collegamento. I frame sono costituiti da un campo dati, nel quale è inserito il datagramma IP, e da vari campi di intestazione aggiunti. La struttura del frame è specificata dal protocollo utilizzato;
- **Accesso al collegamento:** il protocollo MAC (*medium access control*) controlla l'accesso al mezzo trasmissivo specificando regole con cui immettere i frame nel collegamento. Nei casi in cui vari nodi condividono un singolo canale di collegamento, i protocolli a livello di collegamento coordinano la trasmissione;
- **Consegna affidabile:** i protocolli a livello di collegamento che forniscono un servizio di consegna affidabile garantiscono il trasporto senza errori di ciascun datagramma. Questo servizio può essere realizzato attraverso acknowledgment e ritrasmissioni;
- **Rilevazione e correzione degli errori:** il nodo ricevente rilevare degli errori sui bit, che possono essere causati dall'attenuazione del segnale e dai disturbi elettromagnetici. Il rilevamento degli errori nel livello di collegamento è solitamente più sofisticato rispetto a quello del livello trasporto (checksum) in quanto implementato in hardware. La correzione degli errori permette al nodo ricevente di identificare e correggere errori su alcuni bit del frame, evitando ritrasmissioni da parte del mittente.

Il **livello di collegamento** di un host è implementato sostanzialmente da un **adattatore di rete** (*network adapter*), noto anche come **scheda di rete** (*NIC network interface controller*). Il cuore della scheda di rete è il link layer controller, un chip dedicato che implementa la maggior parte dei servizi del livello di collegamento in hardware. Un'altra parte dei servizi è invece realizzata in software ed eseguita dalla CPU dell'host.



RILEVAZIONE E CORREZIONE DEGLI ERRORI

Il rilevamento e la correzione degli errori sui bit sono due servizi generalmente forniti dal livello di collegamento, ma come visto, sono spesso forniti anche dal livello trasporto (in TCP e non in UDP).



Scenario di rilevazione e correzione degli errori.

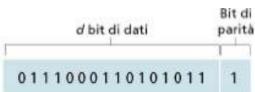
Al nodo trasmittente, ai dati D che devono essere protetti da errori vengono aggiunti dei bit detti EDC (**error detection and correction**).

I dati D e bit EDC sono inviati in sequenza in un frame al nodo ricevente. Questo legge una sequenza di bit D' e EDC' che può essere diversa dall'originale. Il nodo ricevente deve determinare se D' coincide con D, potendo contare soltanto su D' e EDC'.

Anche con l'utilizzo dei bit di rilevazione degli errori è possibile che ci siano degli **errori non rilevati**. Generalmente, le tecniche più sofisticate comportano un'elevata ridondanza, nel senso che sono necessari calcoli più complessi e la trasmissione di molti bit aggiuntivi.

Controllo di parità

La forma più semplice di rilevamento degli errori è quella che utilizza un unico **bit di parità**.



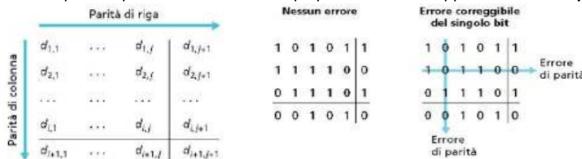
Il mittente include un bit addizionale ai d bit da inviare, e scegli il suo valore in modo da rendere pari il numero totale di bit 1 nei d+1 bit trasmessi.

Con un solo bit di parità il ricevente deve contare il numero di bit a 1 tra quelli ricevuti. Se trova un numero dispari di bit 1, sa che si è verificato almeno un errore in un bit (più precisamente, sa che si è verificato un numero dispari di errori nei bit).

Problema: se si verifica un numero pari di errori nei bit, questi non saranno rilevati.

Se la probabilità di errori nei bit è bassa e si può assumere che gli errori siano indipendenti, l'eventualità di errori multipli in un pacchetto è estremamente ridotta e un solo bit di parità potrebbe bastare. Tuttavia è stato statisticamente provato che gli errori tendono a verificarsi a raffiche (**burst**).

Uno schema più complesso e allo stesso tempo più efficiente è rappresentato dalla **parità a due bit**.



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. I risultanti $i+j+1$ bit di parità contengono bit per la rilevazione dell'errore.

Il ricevente può non solo rilevare che si è verificato un errore, ma può utilizzare gli indici di colonna e di riga per identificare **un bit alterato e correggerlo**. Inoltre anche un errore negli stessi bit di parità è rilevabile e correggibile.

Questo schema di parità può rilevare (ma non correggere) qualsiasi combinazione di due errori in un pacchetto.

La capacità del ricevente sia di rilevare sia correggere gli errori è conosciuta come **forward error correction** (FEC, correzione degli errori in avanti). La peculiarità di queste tecniche è quella di permettere al ricevente di correggere gli errori direttamente, evitando la trasmissione di un NAK e le attese di RTT.

Checksum

I d bit di dati sono trattati come una sequenza di numeri interi da k bit. Un semplice metodo per eseguire il checksum è quello di sommare questi interi e usare i bit del risultato come bit per la rilevazione degli errori.

In Internet i dati sono trattati come interi di 16 bit e sommati. Il complemento a 1 di questa somma costituisce il checksum che viene trasposto dall'**mittente** nell'intestazione dei segmenti. Il ricevente controlla il checksum calcolando il complemento a 1 della somma dei dati ricevuti (compreso il checksum stesso) e verifica che tutti i bit del risultato siano 1. Se non è così viene segnalato un errore.

Nei protocolli TCP e UDP il checksum è calcolato per tutti i campi (intestazione+dati). In IP il checksum è calcolato solo sull'intestazione, perché i segmenti TCP/UDP hanno il proprio.

Essendo uno schema di rilevazione semplice e veloce, la tecnica del checksum si presta bene per il livello di trasporto, implementato in software. Tipicamente la rilevazione di errori al livello di collegamento è implementata mediante hardware dedicato nelle schede di rete, che quindi possono sopportare tecniche più complesse come quella di CRC.

Controllo a ridondanza ciclica (CRC)

Una tecnica largamente utilizzata nelle reti è basata sui codici di controllo a ridondanza ciclica. I codici CRC sono anche detti **codici polinomiali**, in quanto è possibile vedere la stringa di bit da trasmettere come un polinomio in cui i coefficienti sono i bit della stringa, con le operazioni sulla stringa di bit interpretate come aritmetica polinomiale.



Consideriamo d bit costituenti i dati D da trasmettere e supponiamo che sorgente e destinazione si siano accordate su una stringa di $r+1$ bit, conosciuta come **generatore** G . È necessario che il bit più significativo (a sinistra) di G sia 1.

Dato un blocco di dati D , il mittente sceglierà r bit addizionali, R , e li unirà a D in modo da ottenere una stringa di $d+r$ bit che, interpretata come numero binario, sia esattamente divisibile per G (modulo 2).

Il processo di controllo CRC:

- se la divisione $(d+r)/G$ ha un resto diverso da zero, il ricevente sa che si è verificato un errore;
- altrimenti i dati sono accettati come corretti.

Con tale tecnica si possono rilevare tutti gli errori (a raffica) che coinvolgono meno di $r+1$ bit. Inoltre il CRC può rilevare qualsiasi numero dispari di errori.

PROTOCOLLI DI ACCESSO MULTIPLO

Nelle reti esistono due tipi di collegamento:

- **Punto a punto:** costituito da un trasmittente a un'estremità del collegamento e da un unico ricevente all'altra. Alcuni protocolli di questo tipo sono: PPP (point-to-point protocol), HDLC (high-level data link control) e SLIP (Serial Line IP);
- **Broadcast:** può avere più nodi trasmittenti e riceventi connessi allo stesso canale broadcast condiviso.

Nel collegamento broadcast, quando un nodo trasmette un frame, il canale lo diffonde e tutti gli altri ne ricevono una copia. I cosiddetti **protocolli ad accesso multiplo** fissano le modalità con cui i nodi regolano le loro trasmissioni sul canale condiviso.

Dato che tutti i nodi sono in grado di trasmettere frame, è possibile che due o più lo facciano nello stesso istante. Tra questi si genera una **collisione** e una conseguente perdita di frame, mentre il canale rimane inutilizzato. Occorre dunque coordinare la trasmissione dei nodi attivi.

In generale possiamo classificare tutti i protocolli ad accesso multiplo in:

1. **Protocolli a suddivisione del canale** (channel partitioning protocol)
2. **Protocolli ad accesso casuale** (random access protocol)
3. **Protocolli a rotazione** (taking-turn protocol)

Idealmente un protocollo di accesso multiplo per un canale broadcast con velocità di R bit al secondo dovrebbe avere le seguenti caratteristiche:

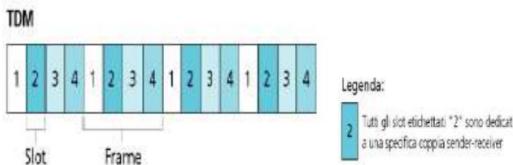
- Quando un solo nodo deve inviare dati, questo dispone di un throughput pari a R bps;
- Quando M nodi devono inviare dati ognuno di essi ha un throughput medio di R/M bps;
- Il protocollo per la gestione dell'accesso è distribuito: assenza di single points of failure;
- Il protocollo è semplice, in modo che risulti economico da implementare (una scheda di rete costa tra i €10 e €20)

PROTOCOLLI A SUDDIVISIONE DEL CANALE

Ricordiamo che il multiplexing a divisione del tempo, TDM e quello a divisione di frequenza, FDM, sono due tecniche che possono essere utilizzate per suddividere la larghezza di banda di un canale broadcast fra i nodi che lo condividono.

Time Division Multiple Access

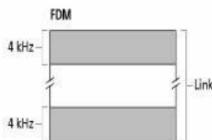
TDM suddivide il tempo in **time frame** e poi divide ciascun intervallo di tempo in N **time slot**. Ogni slot è quindi assegnato a uno degli N nodi. Ogni volta che un nodo ha un pacchetto da inviare, trasmette i bit del pacchetto durante lo slot di tempo assegnatogli.



TDM riesce ad evitare le collisioni ed è perfettamente imparziale: ogni nodo ottiene, durante ciascun intervallo di tempo, un tasso trasmissivo di R/N bps. Tuttavia, anche quando non vi sono altri nodi che devono inviare, quello che vuole inviare deve attendere il suo turno

Frequency Division Multiple Access

FDM suddivide il canale condiviso in frequenze differenti (ciascuna con larghezza di banda R/N) e assegna ciascuna frequenza a un nodo. Quindi, a partire da un canale da R bps, FDM crea N canali di R/N bps.



Come TDM, anche FDM evita le collisioni e divide equamente la larghezza di banda tra gli N nodi. Tuttavia, anche con FDM la larghezza di banda è limitata a R/N, anche quando vi è solo un nodo che deve spedire.

Code Division Multiple Access

Mentre TDM e FDM assegnano rispettivamente ai nodi slot di tempo e frequenze, CDMA assegna loro un **codice**. Ciascun nodo in trasmissione utilizza il proprio codice univoco (**chipping sequence**) per codificare i dati inviati.

- Il segnale trasmesso è ottenuto tramite il prodotto dei **dati X chip**. La sequenza in uscita dal moltiplicatore sarà successivamente modulata e infine trasmessa sul canale
- In ricezione il segnale ricevuto sarà costituito dalla somma vettoriale di tutti i segnali trasmessi dai singoli nodi. L'informazione associata al singolo nodo può essere estratta **moltiplicando il segnale totale per il chip e integrando successivamente** il segnale ottenuto in un intervallo di tempo pari alla durata del bit di informazione.

Se i codici sono scelti accuratamente (codici **ortogonali** tra loro), le reti CDMA consentiranno ai nodi differenti di trasmettere simultaneamente riducendo al minimo l'interferenza. Infatti il risultato di un'operazione di questo tipo (moltiplicazione + integrazione) permetterà di ottenere un segnale che è dato dalla somma di un segnale di ampiezza dominante (segnale utile informativo) associato alla sorgente da estrarre e di un segnale di ampiezza minore, costituito da una combinazione fra rumore e segnali associati alle altre sorgenti.

PROTOCOLLI AD ACCESSO CASUALE

Questa classe di protocolli per l'accesso multiplo concerne situazioni in cui un nodo trasmette sempre alla massima velocità consentita dal canale, cioè R bps.

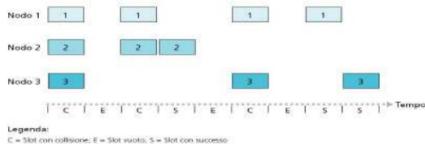
Quando si verifica una collisione, i nodi coinvolti ritrasmettono ripetutamente i loro frame fino a quando non raggiungono la destinazione senza collisioni. La ritrasmissione non è immediata, ma il nodo attende per un **periodo di tempo casuale** indipendente da quello degli altri nodi. Le differenti scelte arbitrarie del tempo di attesa operate dai diversi nodi **possono** consentire ai frame di attraversare il canale senza ulteriori collisioni.

Slotted ALOHA

Si assume nella trattazione di questo protocollo che:

- tutti i frame sono lunghi L bit e il tempo è diviso in slot di uguale durata L/R secondi.
- I nodi cominciano la trasmissione dei frame solo all'inizio degli slot
- I nodi siano sincronizzati in modo che tutti sappiano quando iniziano gli slot

Il protocollo prevede che quando un nodo ha un nuovo frame da spedire, attende fino all'inizio dello slot successivo e poi trasmette l'intero frame. Se non si verifica una collisione non occorre effettuare una ritrasmissione. Se si verifica una collisione, il nodo ritrasmette con probabilità p il suo frame durante gli slot successivi. La probabilità è un numero tra 0 e 1, quindi il nodo al prossimo slot può trasmettere o non trasmettere. Inoltre le probabilità tra tutti i nodi sono indipendenti tra loro.



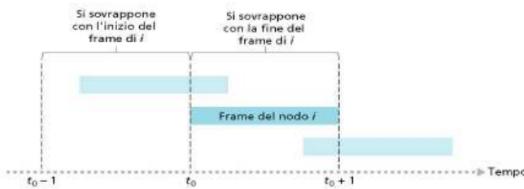
Efficienza: In presenza di molti nodi attivi esistono due possibili problematiche per portano il protocollo ad essere non particolarmente efficiente. In primo luogo, una certa frazione degli slot presenterà collisioni e di conseguenza andrà "sprecata". Il secondo problema è che una grande frazione degli slot risulterà vuota. I soli slot non sprecati saranno quelli utilizzati da un solo nodo per trasmettere.

La **probabilità** che una trasmissione abbia successo *per un dato nodo* è data dalla probabilità che quel solo nodo trasmetta (p) mentre i rimanenti $N-1$ rimangano inattivi ($1-p$) $^{N-1}$: $S = p (1-p)^{N-1}$

Poiché ci sono N nodi: $S = Np(1-p)^{N-1}$. Il **valore ottimo** di p per N che tende all'infinito è $1/e = 37\%$

ALOHA puro

Al contrario di Slotted ALOHA, il primo protocollo ALOHA non richiede che tutti i nodi sincronizzino le loro trasmissioni. Non appena arriva un frame al livello di collegamento, il nodo lo trasmette immediatamente e integralmente nel canale broadcast. Se un frame va in collisione, allora il nodo lo ritrasmette immediatamente con probabilità p o aspetterà, restando inattivo per un altro periodo di tempo, con probabilità $1-p$.

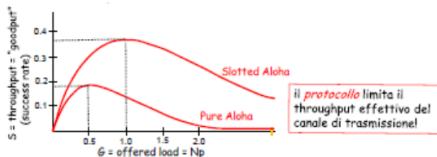


Un frame inviato da un nodo al tempo t_0 può collidere con altri frame inviati in $[t_0-1, t_0+1]$.

Quindi la probabilità di successo di un dato nodo è data dalla probabilità che il nodo trasmetta (p) congiunta alla probabilità che nessun altro trasmetta in $[t_0-1, t_0]$ congiunta con la probabilità con nessun altro trasmetta in $[t_0, t_0+1]$ $S = p (1-p)^{N-1}(1-p)^{N-1} = p(1-p)^{2N-1}$

Il successo di uno su N nodi quindi è $S = N p (1-p)^{2N-1}$

Il **valore ottimo** di p , per N che tende ad infinito, è $1/2e = 18\%$ il che significa che la probabilità di collisione raddoppia rispetto allo Slotted Aloha



CSMA : Carrier Sense Multiple Access

Al contrario del protocollo Aloha prevede alcune politiche che permettono di gestire l'evento in cui vi siano più nodi che interferiscono tra loro nella trasmissione. In particolare esistono due regole importanti per dialogare sul canale condiviso:

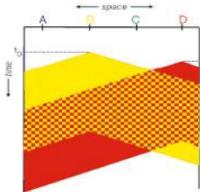
- **rilevamento della portante** (carrier sensing): impone ad un nodo di ascoltare il canale prima di trasmettere. Se il canale sta già trasmettendo un frame, il nodo aspetta finché rileva che il canale è libero per un intervallo di tempo e quindi inizia la trasmissione; Questo meccanismo è detto, per il suo funzionamento **listen before talking**.
- **rilevamento della collisione** (collision detection): il nodo che sta trasmettendo rimane contemporaneamente in ascolto del canale. Se osserva che un altro nodo sta trasmettendo un frame che interferisce con il suo, arresta la propria trasmissione, aspetta un intervallo di tempo casuale e poi ripete il processo. Questo meccanismo è detto **listen while talking**.

Queste due regole sono alla base dei protocolli **CSMA (Carrier sensing)** e **CSMA/CD**.

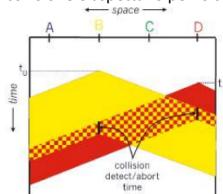
Il CSMA può essere **persistent**, se riprova immediatamente a ritrasmettere con probabilità p quando il canale si libera, o **non persistente** se riprova a ritrasmettere dopo un intervallo casuale.

Discussione:

Nel **CSMA senza collision detection** si possono avere comunque delle collisioni. Un nodo, per trasmettere attende finché il canale non sia libero, però il ritardo di propagazione fa sì che due nodi possano non ascoltare le reciproche trasmissioni. In caso di collisione il tempo di trasmissione della frame risulta completamente sprecato. La **distanza** ed il **ritardo di propagazione** concorrono a determinare la probabilità di collisione.



Nel **CSMA con collision detection** per rilevare le collisioni la scheda di rete resta sul canale anche durante la sua trasmissione e misura la potenza del segnale ricevuto comparandola con quella del segnale trasmesso. Il protocollo annulla la trasmissione non appena si riscontra energia di altri segnali. L'intervallo di tempo da scegliere per il tempo di attesa (**tempo di backoff**) deve essere casuale, perché se due frame entrano in collisione e aspettano per lo stesso periodo di tempo, continueranno ad entrare in collisione.



PROTOCOLLI A ROTAZIONE

Ricordiamo che un protocollo di accesso multiplo dovrebbe avere due caratteristiche fondamentali:

1. quando un solo nodo è attivo, deve avere un throughput di R bps
2. quando sono attivi M nodi, ciascuno deve avere un throughput vicino a R/M bps.

I protocolli ALOHA e CSMA possiedono la prima proprietà ma non la seconda. I protocolli **taking turns** cercano di prendere il meglio tra i due tipi di protocolli.

Polling

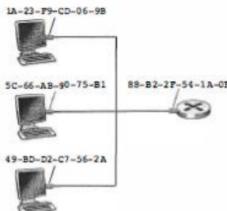
Un nodo **master** (designato come principale) invita “a turno” i nodi **slave** a trasmettere i maniera alternata. Il master invia dei messaggi di tipo “**Request to Send**” per avvisare uno slave che può iniziare ad inviare un dato numero massimo di frame. Il protocollo polling elimina le collisioni e gli slot vuoti, ma presenta alcuni svantaggi. Si introduce una latenza dovuta dal tempo di notifica, ed inoltre vi è la presenza di un single point of failure rappresentato dal master.

Token Passing

In questo caso non esiste un nodo principale, ma un frame, un messaggio di controllo detto **token**, che circola fra i nodi in ordine sequenziale. Il possesso del token dà diritto a trasmettere. Se il nodo che riceve il token non ha pacchetti da inviare, lo inoltra immediatamente al successivo. Altrimenti, procede a trasmettere il numero massimo di frame consentito, prima di inoltrare il token al nodo successivo. Questo protocollo è altamente efficiente, ma non privo di problemi: overhead nella gestione del token, latenza dovuta alla trasmissione del token, presenza ancora una volta di un single point of failure (token).

INDIRIZZI MAC

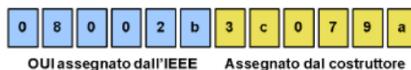
Gli host e i router collegati in una rete possiedono, oltre che un indirizzo di rete (IP), un indirizzo a livello di collegamento. In realtà, non sono propriamente i nodi, ma sono i loro adattatori (schede di rete) ad averli.



Gli indirizzi a livello rete sono detti **indirizzi MAC**, **indirizzi fisici** o anche **indirizzi LAN** assegnati in modo univoco dal produttore ad ogni scheda di rete prodotta (modificabile via software). La IEEE sovrintende alla univocità degli indirizzi e in vende blocchi di MAC.

MAC sta per **Media Access Control** e costituisce uno strato, o sottolivello del livello di collegamento. È utilizzato per l'*instradamento diretto* in reti locali per raggiungere un host una volta raggiunta la sottorete finale di destinazione (tramite IP).

L'indirizzo MAC ha una struttura piatta e non gerarchica, cioè, al contrario dell'indirizzo IP, è indipendente dalla posizione della macchina all'interno della rete. Un indirizzo MAC è lungo 48 bit (6 byte) e suddiviso in 6 cifre esadecimali; le prime 3 individuano il produttore dell'interfaccia di rete (**vendor code o OUI**) e le successive corrispondono al numero di serie della stessa.



Quando una scheda di rete vuole spedire un frame, vi inserisce l'indirizzo MAC di destinazione e lo immette nella LAN. Uno switch presente nella rete potrebbe fare broadcast del pacchetto in ingresso su tutte le sue interfacce. Così ogni scheda collegata alla LAN controlla se l'indirizzo MAC del pacchetto in arrivo corrisponde al suo. In caso affermativo la scheda di rete estrae il datagramma e lo passa al livello di rete.

La **risoluzione degli indirizzi MAC** come descritto in precedenza è demandata ai **protocolli ARP e RARP**.

Gli indirizzi MAC sono di tre tipi:

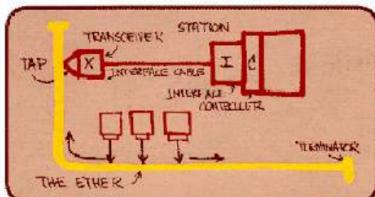
- **Single:** di una singola stazione. Quando una scheda di rete riceve un pacchetto con indirizzo di destinazione un indirizzo single, allora lo passa al livello superiore solo se il DSAP è uguale a quello hardware della scheda (scritto in una ROM) o a quello caricato da software in un apposito buffer;
- **Broadcast:** se il pacchetto ha indirizzo di destinazione broadcast (ff:ff:ff:ff:ff:ff), tutte le schede lo passano al livello superiore
- **Multicast:** di un gruppo di stazioni. Il pacchetto viene passato al livello superiore solo se è stata abilitata la ricezione via software per quel determinato indirizzo.

ETHERNET

Ethernet è una famiglia di tecnologie standardizzate per reti locali LAN, sviluppato a livello sperimentale da Metcalfe e Boggs. Ethernet attualmente è il sistema LAN più diffuso per diverse ragioni:

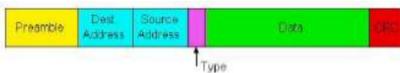
- È stata la prima LAN ad alta velocità con vasta diffusione;
- Più semplice ed economica rispetto alle LAN token ring, FDDI e ATM
- Tasso trasmissivo più alto rispetto alle tecnologie precedenti
- È adeguata all'utilizzo con TCP/IP/7

Le prime LAN Ethernet usavano un cavo coassiale come bus per connettere i nodi. Questa topologia a bus è durata fino alla metà degli anni '90, quando si iniziò a sostituire con topologie a stella basate sugli hub.



Frame Ethernet

L'interfaccia di rete del mittente incapsula i datagrammi IP (o altri pacchetti di livello rete) in **frame Ethernet** che hanno la seguente struttura:



- **Preamble** di 8 byte: i primi 6 byte hanno tutti valore 10101010 e servono a “svegliare” gli adattatori del ricevente e a sincronizzare i suoi clock con quelli del trasmittente. L'ultimo byte ha valore 10101010 e la serie dei due bit a 1 indica al destinatario che sta iniziando la parte dei dati utili;
- **Destination and Source address** di 6 byte: questi due campi contengono l'indirizzo LAN delle schede di destinazione e di sorgente; se l'indirizzo non corrisponde, il livello fisico del protocollo scarta il pacchetto in questione e non lo invia agli strati successivi;
- **EtherType** di 2 byte: indica il tipo di protocollo del livello di rete in uso durante la trasmissione;
- **Payload** da 46 a 1500 byte: contiene i dati reali, che possono essere di lunghezza variabile in base alla MTU. Se i dati superano la capacità massima vengono frammentati, mentre se i dati non raggiungono la lunghezza minima di 46 byte, viene aggiunto del *padding* riempitivo;
- **CRC controllo a ridondanza ciclica** di 4 byte: permette di rilevare se sono presenti errori di trasmissione come descritto in precedenza.

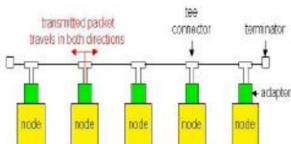
Tecnologie Ethernet

Ethernet compare in molte forme differenti con svariate denominazioni ad esempio:

- 10 BASE-T
- 10 BASE-2
- 100 BASE-T
- 1000 BASE-LX
- 10G BASE-T

La prima parte dell'identificativo fa riferimento alla velocità dello standard (10 Mbps, 100 Mbps, 1 Gbps..). BASE si riferisce a Ethernet "banda base", cioè che il mezzo fisico trasporta solo traffico Ethernet. La parte finale dell'acronimo rimanda al mezzo fisico (ad es. T si riferisce ai doppini di rame).

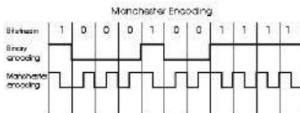
Tipicamente la massima distanza tra i nodi che comunicano via Ethernet è dell'ordine delle centinaia di metri (100m per 10BaseT e 100BaseT, 500m per 100Base2). Nelle reti con topologia a bus vengono infatti utilizzati dei **repeater** (ricevono un segnale in entrata e lo rigenerano in uscita) per ottenere dei segmenti più lunghi.



Tipo di trasmissione

Nelle reti locali Ethernet (10BaseT e 10Base2) viene utilizzata la **codifica Manchester** per evitare il trasferimento nella rete di eventuali lunghe sequenze di zeri o di uno. In pratica invece di codificare il bit 0 e il bit 1 con livelli di tensione fissi, si codificano con transizioni di livelli. La codifica specifica le seguenti convenzioni:

- Per un bit 0 i livelli saranno Basso-Alto
- Per un bit 1 saranno Alto-Basso.



Questo tipo di codifica è considerata auto-sincronizzante (non necessita di un segnale di sincronia esterno).

Ethernet è una tecnologia che fornisce al livello di rete un **servizio senza connessione**. In pratica il mittente invia il frame nella LAN senza alcun handshake iniziale in modalità broadcast a tutti gli adattatori presenti sulla LAN; solo l'adattatore che riconosce il proprio indirizzo lo recepisce.

Il frame ricevuto può contenere errori, la maggior parte dei quali sono verificabili dal **controllo CRC**. Un frame che non supera il controllo CRC viene scartato. Ethernet non è affidabile, perché non prevede la ritrasmissione del frame scartato né una notifica della sua perdita, ma grazie a ciò è semplice ed economica.

Impiego del CSMA/CD

La gestione delle collisioni e dell'occupazione condivisa del canale viene gestita mediante CSMA/CD. L'algoritmo in particolare implementato è il seguente:

- L'adattatore sistema il frame da trasmettere in un buffer
- Se il canale è inattivo, cioè non si rilevano altri pacchetti trasmessi da altre stazioni, si attende un tempo di 96 bit-time e si procede alla trasmissione, se invece è occupato si attende che il canale torni libero prima di ritrasmettere;
- Durante l'intera trasmissione l'adattatore monitora la rete: se non rileva collisioni considera il frame spedito. Le collisioni vengono rilevate come descritto precedentemente.
- Se l'adattatore rileva una collisione durante la trasmissione, arresta la trasmissione e trasmette un segnale di disturbo (*jamming signal*)
- Dopo aver abortito la trasmissione le stazioni trasmittenti applicano ciascuna un algoritmo di backoff attenendo il tempo per la ritrasmissione.

Jammin signal: è un segnale di disturbo di 32 bit che avverte le altre stazioni dell'avvenuta collisione bloccandone la contemporanea trasmissione. Questo passaggio è necessario perché sulle lunghe distanze il segnale potrebbe essere attenuato a tal punto da non rilevare la collisione.

Algoritmo di backoff esponenziale binario: gli adattatori di ciascuna stazione aspettano un tempo casuale entro un valore massimo di fissato. Se si genera una nuova collisione il valore di viene raddoppiato, così fino a questo risulti sufficientemente grande da non produrre collisioni.

HUB

Un **hub** rappresenta un concentratore, ovvero un dispositivo che funge da nodo di smistamento dati di una rete organizzata sia con topologia a bus che con topologia a stella. Nelle reti Ethernet l'hub inoltra i dati in arrivo da una qualsiasi delle sue porte su tutte le altre in broadcast. Per questa ragione può essere definito anche come **ripetitore multi porta**.

La conseguenza dell'utilizzo dell'hub è che la banda totale disponibile in uscita viene frazionata e ripartita tra i vari segnali portanti inviati a causa del moltiplicarsi dei dati da inviare.

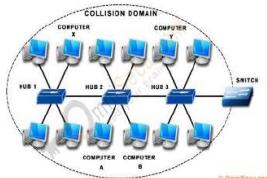
Non è considerato un dispositivo di livello 1 (fisico) in quanto ritrasmette semplicemente segnali elettrici e non entra nel merito dei dati.

Vi sono tra categorie di Hub:

- **Hub attivi:** necessitano di alimentazione elettrica, poiché amplificano il segnale per ridurre al minimo l'attenuato a destinazione;
- **Hub passivi:** non hanno la funzione di amplificazione, quindi non necessitano di alimentazione. Si limitano solo a connettere fisicamente i cavi;
- **Hub ibridi:** sono particolari ed avanzati hub che permettono il collegamento tra più tipologie di cavo;

Sono molteplici i **vantaggi** nell'utilizzo degli hub: la semplicità del comportamento di un hub ne fa uno dei componenti più economici per costruire una rete; l'organizzazione multi-livello garantisce una parziale tolleranza ai guasti; estende la massima distanza possibile tra i nodi (100m per ogni hub).

Tuttavia in alcune applicazioni è necessario considerare il grosso **svantaggio** che comporta: nel gergo, gli hub creano un **unico dominio di collisione**. Se molti host collegati a porte diverse dovessero trasmettere contemporaneamente si verifica una collisione di pacchetti in ricezione e di fatto il throughput effettivo viene limitato dalle numerose ritrasmissioni. Ciò crea delle **limitazioni al numero di nodi** che si possono connettere nella LAN.



SWITCH

È un dispositivo di rete che si occupa di **commutazione a livello di collegamento**. Lo switch agisce sull'indirizzamento e sull'instradamento all'interno delle reti LAN mediante indirizzo fisico, selezionando i frame ricevuti e dirigendoli verso la porta in uscita corretta.

Si distingue quindi dal **router** che opera a livello 3 attraverso il protocollo IP e dall'hub che invece è solo un ripetitore multi porta diffusivo e senza indirizzamento.

Filtraggio: è la funzionalità dello switch che determina se un frame debba essere inoltrato a una qualche interfaccia o scartato

Inoltre: consiste nell'individuazione dell'interfaccia verso cui il frame deve essere diretto e, quindi, nell'inviarlo a quell'interfaccia.

Le operazioni di filtraggio e inoltro in uno switch sono effettuate mediante una **tavella di commutazione** (switch table) che contiene:

- Indirizzo MAC del nodo;
- Interfaccia dello switch che conduce al nodo;
- Il momento in cui la voce per quel nodo è stata inserita nella tabella

Nell'ipotesi in cui un frame con indirizzo MAC di destinazione A giunge allo switch sull'interfaccia x gli scenari possibili sono tre:

1. Nella switch table non vi è una voce per l'indirizzo A; lo switch inoltra copie del frame ai buffer di uscita di tutte le interfacce, eccetto x (manda il frame in broadcast);
2. Nella switch table vi è una voce che associa l'indirizzo A all'interfaccia x. Il frame proviene da un segmento di rete che contiene la scheda di rete A, quindi non deve uscire su nessuna interfaccia e viene dunque scartato;
3. Nella switch table vi è una voce che associa l'indirizzo A all'interfaccia y \neq x; il frame viene inoltrato al segmento di LAN collegato all'interfaccia y.

Autoapprendimento

All'avvio dello switch la tabella risulta vuota. Lo switch esegue un **algoritmo di autoapprendimento** che gli permette di costruire automaticamente, dinamicamente e in modo autonomo le proprie tabelle:

1. La tabella è inizialmente vuota;
2. Di ogni frame che riceve, lo switch archivia nella sua tabella l'indirizzo MAC sorgente del frame, l'interfaccia da cui arriva il frame e il momento di arrivo. Quando tutti i nodi avranno inviato un frame allora la tabella sarà completa;
3. Dopo un **aging time** lo switch cancella dalla tabella gli indirizzi che non mandano più frame.

Dominio di collisione

Uno dei principali **vantaggi** degli switch (rispetto ai collegamenti broadcast o dei collegamenti tramite hub) è che possono essere utilizzati per **suddividere un dominio di collisione** in parti più piccole, riducendo le collisioni possibili e quindi aumentando l'efficienza di utilizzazione del mezzo trasmissivo condiviso.

Quando due nodi comunicano attraverso uno switch, questo crea una sorta di **mezzo trasmissivo dedicato** fra i due nodi, su cui essi possono trasmettere senza disturbare le trasmissioni fra gli altri nodi, né essere disturbati.

Si avranno collisioni solo nel caso in cui, mentre lo switch mette in comunicazione due nodi, altri nodi vogliono comunicare proprio con questi.

IP MULTICASTING

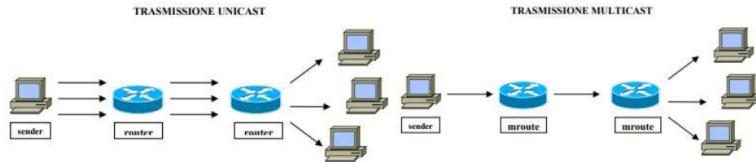
Con il termine **Multicast** si indica la distribuzione simultanea di dati verso un gruppo di destinazione, e si differenzia quindi dal Broadcast (invio a tutti), dall'Anycast (a uno qualunque del gruppo) e dall'unicast (ad uno solo).

L'operazione di trasmissione è unica, nel senso che il sender invia una sola copia dei dati che si moltiplicherà sui router nella rete fino ad arrivare a tutti i nodi che fanno parte del gruppo. Gli host che vogliono ricevere le "trasmissioni" del gruppo multicast devono registrarsi per quel gruppo e la rete si occuperà di consegnare i pacchetti multicast a tutti quelli che si sono registrati.

Il multicast è implementato in ethernet in modo abbastanza semplice: una classe di indirizzi ethernet è riservata all'uso come indirizzi multicast. Tramite il protocollo IGMP i nodi richiedono di essere iscritti ad un gruppo multicast, e tramite appositi algoritmi come DVMRP e PIM, il traffico multicast viene routato tra i diversi nodi.

Differenza unicast e multicast

Se nei sistemi unicast il server distribuisce in rete pacchetti di informazioni distinti al singolo cliente, nel multicast il flusso dati viene inviato in rete una volta sola, ed è a disposizione di qualsiasi utenza sia in grado di riceverlo e ritrasmetterlo.



Capita sempre più spesso in Internet, che un certo numero di utenti voglia accedere contemporaneamente alle stesse informazioni (es. servizi multimediali come Netflix). L'utilizzo del multicast in questi casi ha significativi vantaggi rispetto all'utilizzo dell'unicast.

- **Minore occupazione di banda:** l'occupazione nell'unicast cresce linearmente con il numero degli utenti, mentre rimane costante nel multicast;
- **Minore carico dei server:** ogni server dovrà gestire un solo flusso in uscita anche se crescono gli utenti
- **Minore carico della rete:** anche gli apparati di rete ne risentono poco, dato che anziché duplicare i dati nel multicast, si usa scambiare solo puntatori ad aree di memoria condivisa, in modo da far accedere più interfacce allo stesso buffer dati.

Indirizzi Multicast

Gli indirizzi riservati al multicasting sono quelli che nella suddivisione per classi degli indirizzi IP ricadono nella **classe D**, ossia quelli che iniziano con ‘11110’ (224.0.0.0 – 239.255.255.255); i rimanenti 28 bit identificano il gruppo verso cui il datagramma multicast deve essere spedito.

All'interno di questa classe esiste un certo numero di indirizzi riservati, detti well-known:

- 224.0.0.1 – corrisponde a “tutti gli host del gruppo”: eseguendo un ping a questo indirizzo, tutti gli host facenti parte del gruppo di multicast rispondono alla richiesta;
- 224.0.0.2 – corrispondono a “tutti i router multicast del gruppo”: tutti gli mrouter devono eseguire il join su questo gruppo per ogni interfaccia attiva che possiedono;
- 224.0.0.4 – corrisponde a tutti i DVMRP router
- 224.0.0.5 – tutti gli OSPF router
- 224.0.0.12 – tutti i PIM router

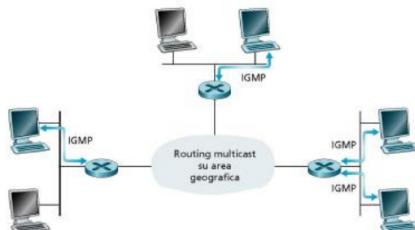
In ogni caso il range di indirizzi che va da 224.0.0.0 a 224.0.0.255 è riservato per scopi locali, amministrativi e di manutenzione. I pacchetti inviati a tali indirizzi non vengono inoltrati dagli mrouter.

IGMP

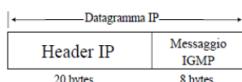
La gestione dei gruppi è di tipo dinamico: un host può unirsi o abbandonare un gruppo in qualsiasi momento e può appartenere contemporaneamente a più gruppi; non è necessario appartenere ad un gruppo per poter inviare ad esso dei messaggi; i membri del gruppo possono appartenere alla medesima rete o a reti fisiche differenti;

IGMP, ovvero Internet Group Management Protocol è lo “strumento” che un router multicast (mrouter) utilizza per essere a conoscenza di host che entrano a far parte di un gruppo (**join**) oppure che lasciano il gruppo (**leave**).

Il protocollo IGMP opera dunque tra un host ed il router ad esso direttamente collegato; mentre degli algoritmi di multicasting routing (descritti in seguito) coordinano i multicast router all'interno della rete Internet, per permettere l'instradamento dei datagrammi multicast.



Il protocollo IGMP è parte integrante del protocollo IP e quindi i messaggi IGMP sono “incapsulati” in datagrammi IP.



Esistono i seguenti tipi di messaggi IGMP utilizzati nel multicasting:

1. **Membership query generale** inviato da un router per informarsi sui gruppi multicast cui gli host locali partecipano
2. **Membership query specifico** inviato da un router per informarsi se uno o più host locali partecipano ad un determinato gruppo multicast
3. **Membership report** inviato da un host per informare il router locale che vuole unirsi (o fa parte di) un determinato gruppo multicast
4. **Leave group** inviato da un host per informare il router che vuole lasciare un determinato gruppo multicast.

Come prevedibile, le Membership Query sono inviate all'indirizzo 224.0.0.1 perché devono essere recepite da "tutti gli host" mentre le Membership Report sono inviate all'indirizzo 224.0.0.2, recepibili dunque da "tutti i router".

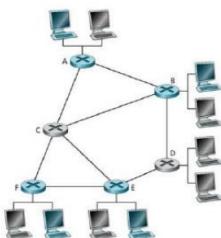
Le funzioni di IGMP sono relative a due fasi differenti:

- **Fase 1:** quando un host si unisce ad un nuovo gruppo, invia un messaggio IGMP ad un particolare indirizzo multicast; i multicast router appartenenti alla rete locale sulla quale tale host si trova, ricevono il messaggio e stabiliscono i meccanismi di routing propagando le informazioni concernenti il gruppo attraverso la rete interconnessa;
- **Fase 2:** dovendo gestire i gruppi in maniera dinamica, i multicast router interrogano periodicamente (mediante polling) gli host sulle varie reti locali, per aggiornare le informazioni relative alla composizione dei gruppi stessi.

Algoritmi di instradamento multicast

Dato il modello di servizio descritto precedentemente, è necessario introdurre un **protocollo di routing Multicast** il cui obiettivo fondamentale è quello di trovare un **albero di link** che collega tutti i router connessi a host che appartengono al gruppo. Una volta ottenuto, i pacchetti multicast saranno instradati lungo questo albero dal mittente a tutti gli host che appartengono all'albero.

Ovviamente, l'albero può contenere router non connessi a host del gruppo multicast, indispensabili, ad esempio, per connettere due rami appartenenti al gruppo.



Possono essere utilizzati due approcci per individuare l'albero di instradamento:

- **Shared Distribution Tree:** L'instradamento multicast con **albero condiviso** si basa sulla costruzione di un unico albero condiviso tra tutti i sender, che include quindi tutti i router periferici con host connessi e appartenenti al gruppo multicast.

Si usa un approccio centralizzato definendo un nodo centrale come punto di **rendezvous**.

- I router periferici con host connessi che appartengono al gruppo, inviano (tramite unicast) messaggi di adesione al nodo centrale.
- I messaggi di adesione proseguono fino a quando si raggiunge un router che già appartiene all'albero multicast o si arriva al centro.
- Il percorso seguito dal messaggio definisce il ramo dell'albero.

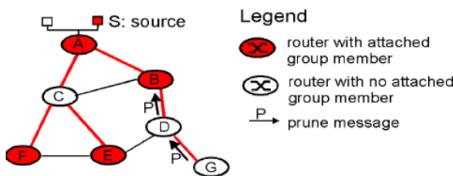
Il punto critico è il processo utilizzato per selezionare il centro stesso (esistono algoritmi appositi).

- **Source Distribution Tree:** questo tipo di approccio all'instradamento crea un albero per ciascuna per ciascuna origine nel gruppo multicast.

Si utilizza un algoritmo RPF (con nodo origine x) per costruire un albero di distribuzione multicast dedicato ai datagrammi multicast generati da x.

Utilizzando RPF classico (già descritto in precedenza) un router inoltrerebbe i pacchetti a qualsiasi router anche se quest'ultimo non è connesso a host che abbiano aderito al gruppo multicast.

La soluzione sta nell'utilizzo del **TRPF (Truncated Reverse Path Forwarding)**. I router che ricevono pacchetti multicast pur non essendo connessi ad host appartenenti al gruppo destinazione, inviano un apposito messaggio di "pruning" verso il router a monte. Un router che riceve tale messaggio da tutti i suoi successori (router connessi in down), itera il procedimento e cioè manda a sua volta un messaggio di pruning a monte.



Instradamento multicast in Internet

1. In Internet il primo protocollo di instradamento multicast utilizzato è il **distance-vector multicast routing protocol (DVMRP)** che implementa alberi basati sull'origine con RPF e potatura. Utilizza un algoritmo Distance Vector che permette ad ogni router di calcolare il link di uscita sul percorso minimo verso ciascuna possibile porta.
2. Il **multicast open shortest path first (MOSPF)** estende OSPF facendo sì che i router si scambino anche le informazioni relative all'appartenenza ai gruppi. In tal modo, i router possono costruire alberi specifici per ogni sorgente, pre-potati, relativi ad ogni gruppo multicast.
3. Il protocollo **core-based tree (CBT)** costruisce un albero "group-shared" bidirezionale, con un unico centro (core). L'aggiunta di rami avviene mediante appositi messaggi di "join" e la gestione dell'albero è affidata a meccanismi di refresh (soft-state).
4. Il più diffuso in Internet però è il **protocol-independent multicast (PIM)**, che prende esplicitamente in considerazione due scenari di distribuzione multicast:
 - In modalità **densa**, i membri del gruppo multicast sono concentrati in una determinata area; ossia la maggior parte dei router nell'area richiede di essere coinvolta nell'instradamento dei datagrammi multicast.
 - In modalità **sparsa**, il numero di router con membri di gruppo connessi è piccolo rispetto al numero totale dei router: i membri del gruppo sono disseminati su un'area più ampia.

In modalità densa, dato che la maggior parte dei router è coinvolta nella trasmissione, PIM utilizza un approccio RPF (simile a quello adottato da DVMRP).

In modalità sparsa, PIM utilizza un approccio center-based in cui i router interessati alla trasmissione inviano messaggi esplicativi di "join" (simile a CBT).

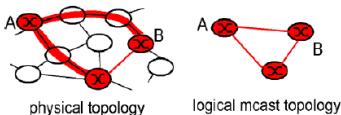
Per instradare datagrammi multicast tra differenti Autonomous System si usano due differenti protocolli:

- **DVMRP**: standard de facto;
- **BGMP** (Border Gateway Multicast Protocol): approccio group shared; ancora in via di sviluppo

Multicast BackBone (MBone)

In Internet è stata realizzata una rete virtuale, chiamata **MBone**, che intercorre tutti i router multicast, e si appoggia su porzioni dell'Internet fisica. Nasce dalla necessità di inviare dati audio e video su una rete geografica ad un certo numero di destinatari, senza congestionare il traffico.

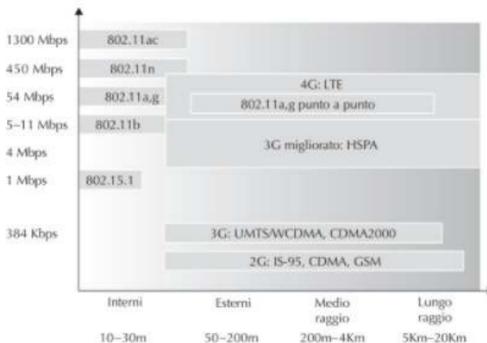
È composta da "isole" capaci di supportare l'instradamento multicast e connesse ad altre "isole" tramite collegamenti virtuali punto-punto chiamati **tunnel**. I pacchetti IP multicast vengono incapsulati come IP su IP, quindi appaiono come normali pacchetti unicast per router intermedi. L'incapsulamento viene aggiunto all'entrata del tunnel e tolto all'uscita dal tunnel.



CAPITOLO 7 - RETI MOBILI E WIRELESS

In una rete wireless possiamo identificare i seguenti elementi:

- **Host Wireless:** dispositivi periferici che eseguono applicazioni.
- **Collegamenti Wireless:** canale di comunicazione wireless. Le due principali caratteristiche degli standard per collegamenti wireless sono **area di copertura** e **frequenza del collegamento**.



- **Stazioni base:** (non ha una controparte nelle reti cablate) è responsabile dell'invio e della ricezione dei pacchetti tra gli host wireless a essa associate. Un esempio di queste stazioni sono i **ripetitori a celle** nelle reti cellulari e gli **access point** nelle LAN 802.11.
- **Infrastruttura di rete:** è la rete più ampia alla quale l'host potrebbe volersi connettere.

Quando un host si sposta sull'area di copertura di un'altra stazione base cambia il punto di collegamento con la rete globale. Questo processo si chiama di **handoff**.

TECNOLOGIE WLAN

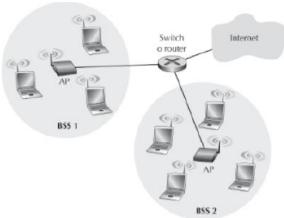
Con il termine **WLAN** o **Wireless LAN**, si indica una rete locale che sfrutta tecnologia wireless, invece di una connessione cablata via cavo.

Tra i vari standard emersi negli anni '90 che implementano l'accesso wireless alle reti LAN, quello che si è affermato sicuramente è **IEEE 802.11 wireless LAN**, conosciuto anche come **Wi-Fi**. Esistono numerosi standard 802.11. La seguente tabella ne riassume le principali caratteristiche.

Standard	Gamma di frequenze (negli Stati Uniti)	Velocità di trasferimento dati
802.11b	2,4 GHz	Fino a 11 Mbps
802.11a	5 GHz	Fino a 54 Mbps
802.11g	2,4 GHz	Fino a 54 Mbps
802.11n	2,5 GHz e 5 GHz	Fino a 150 Mbps
802.11ac	5 GHz	Fino a 1300 Mbps

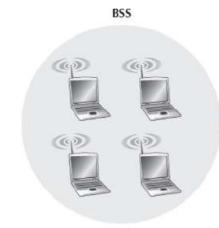
Gli standard 802.11 utilizzano tutti lo stesso protocollo di accesso al mezzo, **CSMA/CA** (accesso multiplo con evitamento delle collisioni), e la stessa struttura del frame a livello di collegamento. Tutti possono ridurre la frequenza trasmissiva per raggiungere distanze maggiori. Inoltre gli standard 802.11 hanno retro compatibilità. Le principali differenze concerno le velocità trasmissive e la gamma di frequenze, come mostrato in tabella.

In un'architettura con standard 802.11 i blocchi principali sono detti **basic service set** e contengono una o più stazioni wireless e una stazione base centrale, detta **access point**.



Le reti wireless che utilizzano AP sono anche chiamate **wireless LAN con infrastruttura**, dove l'infrastruttura è formata dagli AP, dalla rete Ethernet che li collega e da un router di accesso.

Delle stazioni 802.11 possono però anche raggrupparsi per formare una **rete ad hoc**, cioè una rete priva di controllo centrale e di connessioni con il mondo esterno, in cui gli host comunicano stabilendo delle comunicazioni punto-punto.



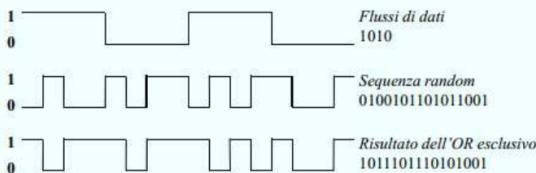
802.11 Livello fisico

802.11 è stato progettato per trasmettere dati usando tre tecniche differenti:

- **Frequency hopping**
- **Direct sequence**
- **Diffuse infrared**

Le prime due tecniche sfruttano il range di frequenza intorno ai 2.4 GHz e sono tecniche del tipo **spread spectrum**: l'obiettivo di tali tecniche è quello di diffondere il segnale su di un intervallo di frequenza ampi, in modo tale da minimizzare l'effetto dell'interferenza da parte di altri dispositivi.

Direct Sequence (DSSS): è una tecnologia a “frequenza diretta” a banda larga, nella quale ogni bit viene trasmesso come una sequenza ridondante di valori, detti chip. L’interfaccia modula il dato prima di trasmetterlo; ogni bit trasmesso viene disperso su una sequenza a 11 bit (sequenza di Barker) di durata minore di un singolo bit di informazione. Il segnale trasmesso consumerà così una maggiore larghezza di banda, consentendo però la ricezione di segnali deboli.



Il mittente in pratica invia il risultato dell’OR esclusivo del bit da inviare e di n bit scelti in maniera casuale.

- 11 canali stazionari da 22 MHz
- Data rate = 11 Mbps
- 3 canali non sovrapposti
- Codifica del bit in una stringa di bit
- Trasmissione delle chipping sequence su un range di frequenze
- Cambio di canale in caso di interferenza

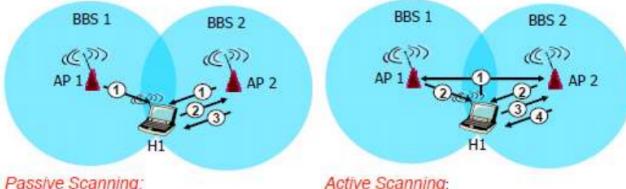
Frequency Hopping (FHSS): è una tecnica usata per aumentare la larghezza di banda di un segnale. Consiste nel variare la frequenza di trasmissione a intervalli regolari in maniera pseudo casuale attraverso un algoritmo prestabilito. Il ricevitore deve utilizzare il medesimo algoritmo del mittente inizializzato con lo stesso seme per decifrare il segnale.

- 79 canali ciascuno ampi 1 MHz
- Cambio di frequenza (hop) almeno ogni 0.4 secondi
- Richiede sincronizzazione
- Ridotta sensibilità alle interferenze
- Un pacchetto perso viene trasmesso al successivo hop

802.11 Canali ed associazione

Prima di inviare o ricevere pacchetti dati 802.11 le stazioni devono associarsi a un AP. Ad ogni AP è associato un **SSID** (service set identifie) e un **numero di canale**. Negli 85 MHz di banda di frequenza in cui opera 802.11 vengono definiti 11 canali parzialmente sovrapposti (due canali non si sovrappongono se sono separati da quattro o più canali). Quindi si potrebbe creare una LAN con capacità trasmissiva globale di 33 Mbps, installando tre AP 802.11 assegnando canali 1,6,11 e connettendoli con uno switch.

Lo standard 802.11 richiede che l’AP invii periodicamente dei **frame beacon**, che contengono il proprio codice SSID e il proprio indirizzo MAC. La stazione wireless analizza gli 11 canali in cerca di questi frame provenienti dagli AP situati nelle vicinanze. Il processo di scansione dei canali e di ascolto dei frame beacon è chiamato **scansione passiva**. Un host wireless esegue anche una **scansione attiva**, inviando in broadcast un frame sonda che verrà ricevuto da tutti gli AP nel raggio di copertura dell’host.



1. Frame beacon inviati dagli AP
 2. invio di un frame di richiesta di associazione
 3. invio di un frame di risposta di associazione
1. Frame sonda di richiesta inviato in broadcast
 2. Frame di risposta inviato dagli AP
 3. Invio di un frame di richiesta di associazione
 4. invio di un frame di risposta di associazione

Al termine della procedura di associazione tipicamente l'host effettuerà una richiesta DHCP per ottenere un indirizzo IP nella subnet dell'AP.

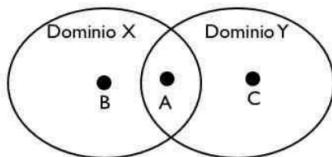
802.11 Protocollo MAC

Una volta che una stazione wireless è associata a un AP, può iniziare a trasmettere e ricevere frame dati da e verso l'AP. Poiché stazioni multiple potrebbero voler trasmettere frame di dati contemporaneamente sullo stesso canale, è necessario un protocollo ad accesso multiplo per coordinare le trasmissioni.

Come per Ethernet, in 802.11 è stato scelto un protocollo ad accesso casuale, in particolare **CSMA con prevenzione di collisioni (CSMA/CA)**. È un protocollo con rilevazione della portante, ovvero ciascuna stazione ascolta il canale prima di trasmettere e si astiene dal farlo se rileva che il canale è occupato.

La differenza del CSMA/CA, con il CSMA/CD di Ethernet è che il secondo **previene le collisioni** invece di rilevarle. Questo è importante in quanto il rilevamento delle collisioni non è realizzabile:

- Esistenza dei **nodi nascosti**: un nodo è visibile da un AP, ma non da altri nodi che possono vedere lo stesso AP. Non potendo comunicare B e C non sanno dell'interferenza in comune con A.



- La possibilità di rilevare collisioni richiede la capacità di inviare (il segnale proprio della stazione) e ricevere (per determinare su un'altra stazione sta trasmettendo) contemporaneamente. Essendo in generale la potenza della ricezione nettamente inferiore (**fading**) risulta molto costoso costruire hardware che possa rilevare collisioni.

Dato che le reti 802.11 non utilizzano la rilevazione delle collisioni, una volta che una stazione inizia a trasmettere un frame, lo trasmette interamente. Dunque devono essere implementati meccanismi per prevenire completamente la collisione ed evitare la ritrasmissione.

Un'altra differenza rispetto a CSMA/CS è che utilizza uno **schema di conferma di avvenuta ricezione** a livello di collegamento, essendo il tasso di errore nei bit nei canali wireless piuttosto elevato.

Quando la stazione di destinazione riceve un frame che passa il controllo CRC, attende per un breve periodo di tempo, noto come **SIFS** (short inter-frame space), dopo il quale invia al mittente un frame, utilizzando ancora il protocollo CSMA/CA per accedere al canale. Se il frame di conferma non sarà ricevuto dopo un numero prefissato di ritrasmissioni, la stazione trasmittente passerà oltre e scarterà il frame.



Descrizione del protocollo CSMA/CA nello scenario in cui una stazione abbia un frame da trasmettere:

1. Se inizialmente la stazione percepisce il canale come inattivo, allora trasmette il suo frame dopo un breve periodo di tempo detto **DIFS** (distributed inter-frame space).
2. Se il canale è occupato, la stazione sceglie un valore casuale di ritardo usando un algoritmo di backoff esponenziale e decrementa questo valore dopo DIFS solo quando il canale viene percepito come inattivo. Se il canale è percepito come occupato, il contatore rimane fermo.
3. Quando il contatore arriva a zero (si verifica quando il canale è inattivo), la stazione trasmette l'intero frame e aspetta il frame di conferma.
4. Se riceve la conferma, la stazione sa che il frame stato ricevuto correttamente e, qualora avesse un altro frame da inviare, riattiva il CSMA/CA da passo 2. Se il frame di conferma non viene ricevuto, la stazione trasmittente ritorna al passo 2, ma con un valore di ritardo maggiore.

Questo algoritmo di prevenzione delle collisioni funziona abbastanza bene. Supponiamo che due stazioni vogliano trasmettere mentre una terza sta occupando il canale. In Ethernet le stazioni trasmetterebbero non appena il canale si sia liberato, con conseguente collisione e interruzione della trasmissione. In 802.11 la collisione non può essere rilevata e i frame verrebbero inviati per intero e consegnati.

Allora in questo scenario, le due stazioni appena rilevano che il canale è occupato vanno in ritardo con valore casuale, con la speranza di scegliere due valori di ritardo diversi. Se è così la collisione sarà stata evitata.

Certamente in tale scenario se le due stazioni fossero nascoste possono ancora verificarsi collisioni.

802.11 Terminali nascosti: RTS e CTS

Il protocollo MAC 802.11 include anche un (opzionale) schema di prenotazione che aiuta ad evitare collisione anche in presenza di terminali nascosti.

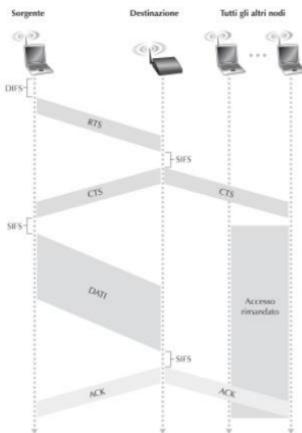


Le due stazioni sono collegate nel raggio dell'AP al quale sono associate. A causa dell'attenuazione del segnale, anche il raggio di copertura delle stazioni wireless è limitato all'interno dei cerchi laterali più chiari. Ciascuna delle stazioni wireless è nascosta all'altra, sebbene nessuna sia nascosta all'AP.

Se H1 sta trasmettendo un frame e, a metà della trasmissione, H2 voglia trasmettere un frame all'AP. H2 non rilevando la trasmissione di H1, attende un DIFS per poi trasmettere il frame, causando una collisione.

Sono previsti due frame di controllo per evitare questo problema: **RTS** (Request to send) e **CTS** (clear to send):

1. Il trasmittente, quando vuole inviare un frame DATI, invia un RTS all'AP, indicando il tempo totale per la trasmissione di DATI+ACK.
2. L'AP quando riceve il frame RTS, risponde diffondendo in broadcast il frame CTS. Questo comunica al trasmittente il permesso di inviare e comunica alle altre stazioni di non trasmettere durante il periodo di tempo riservato.



L'utilizzo di questi due frame risolve il problema del terminale nascosto perché il canale deve essere prenotato e quindi utilizzato in esclusione mutua.

Possono esserci delle collisioni ma di breve durata, dato che coinvolgono solo i frame RTS e CTS che sono piccoli

PACCHETTO IEEE 802.11

I frame 802.11 sono molto simili ai frame Ethernet.



Dettaglio del campo di controllo: (I numeri indicano la lunghezza del campo in bit)

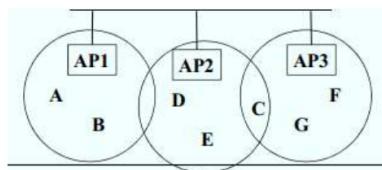
2	2	4	1	1	1	1	1	1	1	1	1
Versione del protocollo	Tipo	Sottotipo	Verso AP	Da AP	Frammentazione	Copia	Alimentazione	Altri dati	WEP	Riservato	

- **Campo Payload e CRC:** solitamente il payload consiste di un datagramma IP o di un pacchetto ARP. Come in Ethernet il campo CRC di 32 bit permette al ricevente il rilevamento degli errori nei bit.
- **Campo indirizzo:** sono previsti 4 campi indirizzo MAC di 6 byte:
 - Indirizzo MAC della stazione AP o dell'host ricevente
 - Indirizzo MAC della stazione che trasmette il frame
 - Indirizzo MAC dell'interfaccia del router al quale l'AP è collegato
 - Indirizzo MAC usato sono nelle reti ad hoc senza infrastruttura
- **Campi numero di sequenza:** permette al ricevente di distinguere tra un frame appena trasmesso e la ritrasmissione di un frame.
- **Campo durata:** usato dalla stazione trasmittente per riservare il canale per un periodo di tempo che include la trasmissione del suo frame dati e quella del frame di consegna
- **Campo controllo del frame:** contiene diversi sottocampi:
 - **Tipo:** può essere data, RTS frame, CTS frame, ACK
 - **Protocol:** specifica la versione di 802.11 utilizzata
 - **Verso AP e da Ap:** definiscono la funzione dei diversi campi indirizzo.

802.11 Distribution system

Per fornire supporto alla mobilità e la connessione ad altre reti, si utilizzano dei nodi speciali:

- **Access Point (AP):** si tratta di nodi connessi a un'infrastruttura di rete fissa detta **Distribution System**.



Ogni nodo si associa ad un particolare access point. Se A vuole comunicare con F:

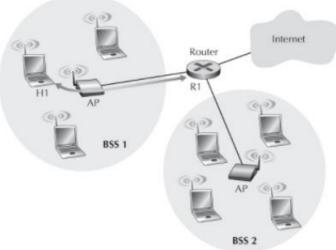
1. A invia un frame al suo access point AP1
2. AP1 inoltra ad AP3 il frame attraverso il distribution system
3. AP3 trasmette il frame a F

La tecnica per selezionare un Access Point è detta **scanning** e prevede 4 fasi:

1. Il nodo invia un frame di **probe**;
2. Tutti gli AP alla portata del nodo rispondono con un frame di **risposta al probe**;
3. Il nodo seleziona uno degli AP (tipicamente quello con la migliore qualità del segnale ricevuto), e gli invia un frame di **richiesta di associazione**;
4. L'AP selezionato risponde con un frame di **conferma di associazione**.

Indirizzamento 802.11

Supponiamo si trasportare un datagramma dall'interfaccia R1 del router alla stazione wireless H1.



Il router non è consapevole della presenza di un AP tra sé e H1 (H1 è visto semplicemente come un host delle sottoreti alle quali il router è connesso).

1. Il router utilizza ARP per determinare l'indirizzo MAC di H1 (dato che conosce il suo IP).
2. Dopo aver ottenuto l'indirizzo MAC di H1, l'interfaccia del router incapsula il datagramma in un frame Ethernet. Il campo indirizzo sorgente conterrà l'indirizzo MAC di R1 e il campo indirizzo destinazione conterrà l'indirizzo MAC di H1.
3. Quando il frame Ethernet giunge all'AP, questo lo converte in un frame 802.11. L'AP riempie i campi indirizzo 1 e 2 con l'indirizzo MAC di H1 e il proprio indirizzo MAC. Per l'indirizzo 3, AP inserisce l'indirizzo MAC di R1.

Quando l'host H1 risponde ad R1:

1. H1 crea un frame 802.11, riempiendo i campi indirizzo 1 e 2 con l'indirizzo MAC dell'AP e l'indirizzo MAC di H1. Per l'indirizzo 3, H1 inserisce l'indirizzo MAC di R1.
2. Quando l'AP riceve il frame 802.11, lo converte in frame Ethernet. Il campo dell'indirizzo sorgente per questo pacchetto è l'indirizzo MAC di H1, mentre il campo di indirizzo di destinazione è l'indirizzo MAC di R1.

In sintesi lo scambio tra reti cablate e reti wireless, è giocato tutto dall'indirizzo 3 che permette di interconnettere i due tipi di rete, di fatto scambiando gli indirizzi nella trasmissione.

CAPITOLO 8 – SICUREZZA NELLE RETI

La sicurezza della comunicazione in rete coinvolge diversi aspetti:

- **Riservatezza:** solo mittente e destinatario dovrebbero essere in grado di comprendere il contenuto del messaggio trasmesso.
- **Integrità del messaggio:** occorre che il contenuto della comunicazione non subisca, durante la trasmissione, alterazioni dovute a cause fortuite o a manipolazioni.
- **Autenticazione:** mittente e destinatario devono essere reciprocamente sicuri della loro identità, cioè devono poter confermare che l'altra parte sia effettivamente chi dichiara di essere.
- **Sicurezza operativa:** i servizi offerti in rete devono essere protetti da eventuali attacchi che pregiudichino la loro accessibilità e disponibilità.

Esempi di comportamenti malevoli:

- **Eavesdropping:** intercettare i messaggi in una comunicazione tra due interlocutori;
- Inserire messaggi fasulli nel flusso di una comunicazione
- **Spoofing:** inviare pacchetti con il campo source address fasullo;
- **Hijacking:** dirottare la comunicazione piazzandosi “in mezzo” alla comunicazione
- **Denial of service:** impedire ad un servizio offerto in rete di essere utilizzabile.

Un **sistema crittografico** è un sistema in grado di cifrare (encryption) e decifrare (decryption) un messaggio attraverso l'uso di un **algoritmo** e una **chiave** (stringa alfanumerica). Il messaggio da cifrare è detto **testo in chiaro** (plaintext o cleartext) mentre il risultato dell'algoritmo crittografico è detto **testo cifrato** (ciphertext).

Esistono due tipologie di crittografia:

- Crittografia a **chiave simmetrica**: mittente e destinatario usano la stessa chiave (segreto condiviso) per encryption e decryption;
- Crittografia a **chiave pubblica**: la chiave per la encryption è pubblica (nota a tutti) mentre la chiave per la decryption è segreta (privata).

CRITTOGRAFIA A CHIAVE SIMMETRICA

In un sistema a chiave simmetrica, entrambi gli interlocutori devono conoscere la chiave $K_{A,B}$ per cifrare e decifrare il messaggio.

Si è dimostrato che con una chiave di almeno 128 bit (13 caratteri) è impossibile decrittare un codice in tempi relativamente limitati. Il problema di questo tipo di crittografia dunque non è tanto legato alla complessità di crittazione ma tanto alla funzionalità logistica. La chiave deve essere distribuita tra mittente e destinatario mantenendola rigorosamente segreta e quindi adoperando un **canale di comunicazione sicuro**.

Due esempi di crittografia simmetrica:

Cifrario per sostituzione: unità di testo del plaintext sono sostituite con corrispondenti sequenze di simboli nel testo cifrato secondo uno schema regolare. In particolare, in un **cifrario monoalfabetico**, abbiano una corrispondenza fissa tra ciascuna lettere dell'alfabeto in chiaro ed una lettera dell'alfabeto cifrato.

plaintext:	abcdefghijklmnopqrstuvwxyz
ciphertext:	mnbvexzasdfghjklpoiuwtrewq

Con un attacco a forza bruta in $26! = 2^{88}$ tentativi è possibile decifrare il codice. Con tecniche di critto-analisi che prevedono lo studio di pattern ricorrenti ed analisi statistiche delle frequenze di occorrenza è ancora più semplice decifrare.

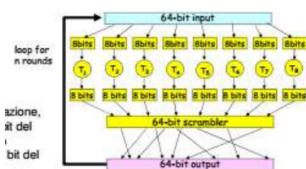
Cifrari a blocchi: un blocco di k bit del testo in chiaro è codificato con altri k bit nel testo in codice secondo uno schema fisso, ad esempio:

000 → 110	001 → 111	010 → 101	011 → 100
100 → 011	101 → 010	110 → 000	111 → 001

In pratica vi è una corrispondenza uno a uno tra ingresso e uscita. Prendendo blocchi di k bit è possibile creare 2^k permutazioni (corrispondenze) diverse. Si può vedere ciascuna di queste corrispondenze come una chiave. Aumentando quindi il valore di k è possibile aumentare la sicurezza.

Il problema è mantenere, sia per la cifratura che per la decifratura una tabella di 2^k elementi in memoria.

Quello che si usa fare tipicamente è usare delle funzioni che simulano in modo casuale tabelle permutate. Quello che segue è un esempio di una funzione di quel tipo.



1. Suddivide il blocco di 64 bit in 8 parti di 8 bit ciascuno
2. Ciascuna parte viene elaborata da una tabella di 8x8 bit
3. Le parti criptate sono riassemblate e le posizioni dei 64 bit sono rimescolate.
4. Questo risultato viene rinviato all'ingresso a 64 bit, dove inizia un'altra interazione (solitamente 16 iterazioni).

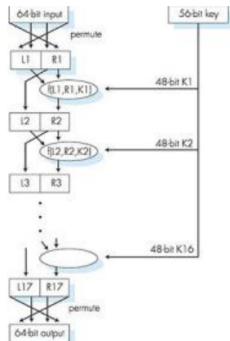
Data Encryption Standard DES

Si tratta di un algoritmo a chiave simmetrica con chiave a 64 bit (ma solo 56 utili poiché 8 sono di controllo).



Sui 64 bit dopo ogni 7 bit, l'ottavo è utilizzato come bit di parità per i 7 precedenti (ottavo, 16-esimo,...,64-esimo).

L'algoritmo prevede 16 cicli identici. Ci sono inoltre una permutazione iniziale e una iniziale che sono tra di loro diverse. Prima dell'iterazione principale il blocco è diviso in due metà di 32 bit e processato alternativamente. Questo assicura che la cifratura e la decifratura siano processi simili, con l'unica differenza che le sottochiavi sono applicate nell'ordine inverso.



Ad ogni iterazione viene usata la cosiddetta **funzione di Festel** che mescola metà del blocco con una parte della chiave. Il risultato della funzione è poi combinato con l'altra metà del blocco, e le due metà sono scambiate prima del ciclo successivo.

Attualmente DES è considerato insicuro per via della chiave di solo 56 bit. Con la potenza di calcolo disponibili oggi si può forzare una chiave DES in poche ore. L'algoritmo è ritenuto più sicuro reiterandolo 3 volte nel **Triple DES**. Negli ultimi anni DES è stato sostituito da AES che elimina molti dei problemi del DES.

Advanced Encryption Standard AES

A differenza del DES elabora i dati a blocchi da 128 bit e la chiave può essere di 128, 192 o 256 bit. L'algoritmo opera utilizzando matrici di 4x4 byte chiamati **states**. Per cifrare sono previsti diversi **round** o cicli di processamento in cui vengono operate:

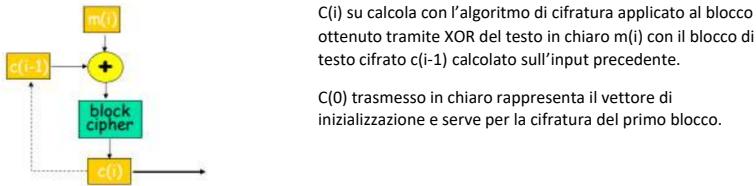
- sostituzioni non lineari di tutti i byte
- spostamento dei byte in base alla riga di appartenenza
- combinazione dei byte con un'operazione lineare
- combinazione dei byte con la chiave di sessione.

Con la forza bruta una chiave AES a 64 bit è stata violata in 5 anni.

Cipher Block Chaining

La debolezza più forte dei cifrari a blocchi è che blocchi in input uguali producono lo stesso testo cifrato e si prestano dunque a crittoanalisi.

Per ovviare a questo problema si usa la tecnica detta **cipher block chaining**. Con questa modalità, l'algoritmo a blocchi crittografa il messaggio creando una catena di blocchi in cui ognuno di essi dipende dalla cifratura del blocco precedente. Questa interdipendenza assicura che un cambiamento ad un qualsiasi bit del testo in chiaro causerà un cambiamento nel blocco finale crittografato.



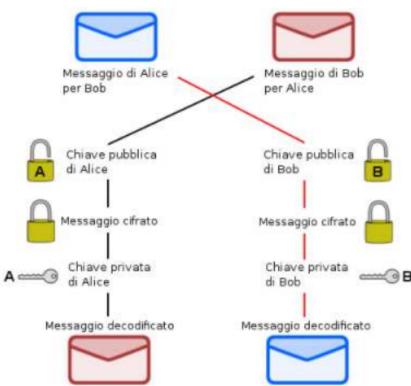
CRITTOGRAFIA A CHIAVE PUBBLICA

L'inconveniente della crittografia a chiave simmetrica, come già accennato, è che mittente e destinatario devono in qualche modo scambiarsi la chiave tramite un canale sicuro.

In un **sistema a chiave pubblica**:

- la **chiave pubblica** usata solitamente per la cifratura deve essere nota a tutti.
- la **chiave privata** usata solitamente per la decifratura è nota solo al destinatario.

Se con una delle due chiavi si cifra il messaggio, allora quest'ultimo sarà decifrato solo con l'altra. È importante notare che le due chiavi sono intercambiabili nei ruoli di cifratura e decifratura.



La forza di un sistema a chiave pubblica si basa sulla difficoltà di determinare la chiave privata corrispondente alla chiave pubblica.

A causa del peso computazionale della crittografia a chiave pubblica, essa di solito è usata per piccoli blocchi di dati, in genere il trasferimento di una chiave simmetrica. Questa è poi utilizzata per cifrare messaggi lunghi.

È possibile utilizzare le chiavi fondamentalmente per due scopi:

1. **Inviare un messaggio cifrato ad un destinatario.** Il mittente cifra il messaggio con la chiave pubblica del destinatario. L'unico a poter decifrare è il destinatario, possessore della chiave privata.
2. **Verificare l'autenticità di un messaggio.** Il possessore della chiave privata cifra il messaggio con la sua chiave privata. Il destinatario verifica l'autenticità del messaggio decifrando con la chiave pubblica del mittente. Tutti i possessori della chiave pubblica possono leggere il messaggio.

RSA Generare una coppia di chiavi

Per utilizzare questo tipo di crittografia, è necessario dunque **creare una coppia di chiavi** tale che:

$$K_B^{-1}(K_B^+(m)) = m$$

Gli algoritmi utilizzati per generare le chiavi si basano spesso su problemi matematici che attualmente non ammettono alcuna soluzione particolarmente efficiente. Uno di questi è l'**algoritmo RSA** (Rivest, Shamir, Adleman):

1. Si scelgono due numeri primi grandi **p** e **q**
2. Si calcolano **n=p x q** chiamato **modulo** e il prodotto **z=(p-1)(q-1)**
3. Si sceglie un numero **e** ($e < n$) chiamato **esponente pubblico** che non abbia fattori comuni con **z**
4. Si sceglie un numero **d** chiamato **esponente privato** tale che **e x d-1** sia esattamente divisibile per **z** ($exd \bmod z = 1$)

La **chiave pubblica** è (n, e) , la **chiave privata** è (n, d)

La robustezza dell'algoritmo sta nel fatto che per calcolare **d** da **e** (o viceversa) non basta la conoscenza di **n** ma serve il numero $z=(p-1)(q-1)$, e che il suo calcolo richiede tempi molto elevati. Infatti **fattorizzare** in numeri primi è un'operazione computazionalmente costosa.

RSA cifratura e decifratura

Dati (n, e) e (n, d) come calcolati precedentemente:

- La **cifratura** del testo in chiaro **m** si effettua calcolando
 $c = m^e \bmod n$ (resto della divisione di m^e per n)
- La **decifratura** del testo cifrato **c** si effettua calcolando
 $m = c^d \bmod n$ (resto della divisione di c^d per n)

$$m = (\underbrace{m^e \bmod n}_c)^d \bmod n$$

INTEGRITA' DEI MESSAGGI

Un altro tema ugualmente importante, dopo la trattazione della **riservatezza** di una comunicazione è quello dell'**integrità dei messaggi**, anche noto come autenticazione dei messaggi. Una soluzione a questo problema sono le **funzioni di hash crittografiche**.

Una funzione di hash prende un testo in ingresso m , e produce una stringa di lunghezza prefissata $H(m)$. La funzione H è tale che è computazionalmente impossibile trovare 2 messaggi diversi x e y tali che $H(x)=H(y)$. Equivalentemente noto $m = H(x)$.

MD5: È un algoritmo di hashing molto usato ed è in grado di calcolare una hash di 128 bit con un processo a quattro fasi.

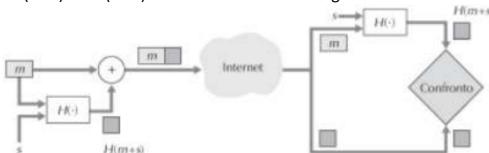
- Si inizia con la normalizzazione attraverso l'aggiunta del valore 1 seguito da una serie di 0, in un numero tale da soddisfare determinate condizioni.
- Aggiunta in coda di una rappresentazione a 64 bit della lunghezza del messaggio originale
- Inizializzazione di un accumulatore
- I blocchi composti da 16 gruppi di 32 bit del messaggio vengono "triturati" tramite 4 cicli di elaborazione.

SHA-1: È un altro importante algoritmo di hashing basato su principi simili a quelli di MD\$. Produce una sintesi (**digest**) del messaggio di 160 bit. La maggiore lunghezza del risultato rende SHA-1 più sicuro.

Autenticazione dei messaggio

Per garantire l'integrità dei messaggi, oltre alle funzioni di hash, mittente e destinatario hanno bisogno di un segreto condiviso, chiamato **chiave di autenticazione**. L'integrità è realizzata come segue:

1. Il mittente crea un messaggio m , concatena s con m per creare $m+s$, calcola la stringa di hash $H(s+m)$. Questa stringa calcolata è detta **codice di autenticazione del messaggio** (MAC)
2. Il mittente aggiunge il MAC al messaggio m e lo manda al destinatario
3. Il destinatario riceve il messaggio esteso ($m+MAC$) e avendo ricevuto m e conoscendo s , calcola il MAC $H(m+s)$. Se $H(m+s)=h$ è stata realizzata l'integrità.



Legenda:

[m] = Messaggio
s = Segreto condiviso

AUTENTICAZIONE

La **firma digitale** è una tecnica crittografica che permette di realizzare l'**autenticazione** nelle comunicazioni in rete. Le firme digitali si propongono di avere tre importanti requisiti:

- **Autenticazione:** il destinatario deve poter verificare l'identità del mittente;
- **Non ripudio:** il mittente deve poter disconoscere un documento da lui firmato;
- **Integrità:** il destinatario non deve poter modificare un documento firmato da qualcun altro.

Un tipico schema di firma elettronica basata sulla crittografia a chiave pubblica prevede i seguenti step:

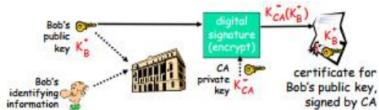
1. Il mittente A firma il messaggio m crittografandolo con la sua chiave privata creando un messaggio firmato.
2. Il destinatario B riceve il messaggio m con la firma digitale di A.
3. Il destinatario verifica che m sia stato effettivamente firmato da A decifrando con la chiave pubblica di B.

Certificazione della chiave pubblica

Un importante applicazione della firma digitale è la **certificazione della chiave pubblica**, cioè la certificazione che una chiave pubblica appartenga a una specifica entità. Per utilizzare la crittografia a chiave pubblica, infatti, è necessario che gli utenti abbiano garanzie sulle chiavi pubbliche fornite.

Generalmente, la relazione tra una data chiave pubblica e una determinata entità è stabilita da un'**autorità di certificazione** (CA *certification authority*), il cui compito è di validare l'identità ed emettere **certificati di autenticazione**. L'unico modo per sapere se la validazione di identità è stata fatta in modo corretto è fidarsi dell'autorità.

La CA una volta validata l'identità emette un certificato firmato con l'identità stessa della CA.



Quando un utente A desidera la chiave pubblica di un'altra entità B deve ottenere il certificato di B e poi applicare la chiave pubblica della CA al certificato di B per ottenere la chiave pubblica di B.

