# RETI DI CALCOLATORI

(A. S. Tanenbaum)

CAPITOLO 1

# INTRODUZIONE

Una **rete di calcolatori** è costituita da un insieme di computer autonomi collegati da una singola tecnologia e, più in generale, ci troviamo in presenza di una rete ogni qual volta un concetto possa essere schematizzato mediante un grafo. Due computer sono interconnessi quando sono in grado di scambiare informazioni. Un **sistema distribuito**, a differenza di una rete di calcolatori, è un insieme di computer che appare all'utente come un singolo sistema coerente. A questo scopo è posto al di sopra del sistema operativo uno strato software, detto *middleware*, che ha come scopo quello di dare al sistema un alto grado di coesione e trasparenza alla rete sottostante.

## 1.1 SCOPI DELLE RETI DI CALCOLATORI

### Applicazioni aziendali

Il punto focale in questo ambito è la **condivisione delle risorse** con l'obiettivo di rendere disponibili a chiunque sulla rete programmi, periferiche e soprattutto dati indipendentemente dalla posizione fisica di utente e risorsa. Nella sua forma più semplice ci si può immaginare che nel sistema informatico di un'azienda i dati siano memorizzati in computer ad alte prestazioni chiamati *server* e che gli impiegati abbiano accesso a macchine più semplici chiamati *client* tramite le quali hanno accesso ai dati remoti.
Una rete di calcolatori può inoltre offrire un potente **mezzo di comunicazione** (e-mail, documenti condivisi, videoconferenze, ecc) e permettere alle aziende di realizzare affari con altre aziende in modo elettronico (*e-commerce*).

### Applicazioni domestiche

Se inizialmente il motivo determinante per l'acquisto di un computer era la possibilità di giocare e scrivere testi ora è sicuramente l'accesso ad Internet. Alcuni dei suoi utilizzi più popolari sono: 1) **accesso a informazioni** remote; 2) **comunicazione** da persona a persona (e-mail, *instant messaging*, *chat room*, *newsgroup*, P2P, ecc); 3) **intrattenimento** interattivo (video *on demand*, gioco *on-line*, ecc); 4) ***e-commerce*** (B2C, B2B, G2C, C2C, P2P, ecc).

### Applicazioni mobili

I *notebook* e in PDA sono uno dei segmenti in maggiore crescita dell'industria informatica per un semplice motivo: chiunque con un computer portatile e un modem *wireless* può essere connesso ad Internet come se il computer fosse collegato alla rete fissa. A questo scopo i principali utilizzi di questa tecnologia sono: 1) l'**ufficio portatile**; 2) l'***m-commerce***; 3) le **PAN** e i computer da indossare (*wearable network*).

## Risvolti sociali

Con l'avvento delle reti di calcolatori sono sorti problemi quali la necessità o meno di censura, la legittimità o meno della violazione della privacy, i furti d'identità e le violazioni di copyright.

# 1.2 HARDWARE DI RETE

Le **reti *broadcast*** sono costituite da un solo canale di comunicazione che è condiviso da tutte le macchine. Brevi messaggi posso essere inviati da ciascuna macchina e ricevuti da tutte le altre macchine sulla rete (*broadcasting*), da un loro sottoinsieme (*multicasting*) o da una singola (*unicast*). Alla ricezione del pacchetto la macchina controlla il campo indirizzo: se il pacchetto è indirizzato alla macchina ricevente viene processato, altrimenti viene semplicemente ignorato.
Al contrario le reti **punto-punto** consistono di molte connessioni tra le singole coppie di macchine, quindi per andare dalla sorgente al destinatario un pacchetto deve visitare una o più macchine intermedie. La presenza di più percorsi con diversa lunghezza causa la necessità di trovare quello migliore per l'invio del messaggio.

## Reti locali

Solitamente chiamate LAN, hanno una dimensione compresa tra i 10 m e 1 km il che significa che il tempo di trasmissione più sfavorevole è noto. Le LAN possono usare una tecnologia di trasmissione rappresentata da un cavo a ci sono connesse tutte le macchine. Per le reti locali di tipo *broadcast* sono possibili due differenti topologie: 1) la **rete a bus**, in cui ad ogni istante è *master* al più una macchina e sono necessari meccanismi di arbitraggio (centralizzati o distribuiti) per risolvere i conflitti quando due o più macchine desiderano trasmettere simultaneamente; 2) la **rete ad anello**, in cui ogni bit si propaga in modo autonomo senza aspettare il resto del pacchetto. Le reti di tipo *broadcast* si possono ulteriormente dividere in statiche e dinamiche a seconda del modo in cui allocano il canale. A loro volta le reti dinamiche possono essere centralizzate o meno a seconda della presenza di un'unità centrale che gestisce di volta in volta l'uso del mezzo trasmissivo.

## Metropolitan Area Network

Una MAN si estende fino a 10 km e l'esempio più noto di questo tipo di rete è la televisione via cavo. In prima approssimazione nello schema di una MAN è presente una sorta di *head end* a cui viene demandata la successiva distribuzione del servizio nelle singole abitazioni.

## Wide Area Network

Una WAN si estende in un'area compresa tra i 10 e i 1000 km e racchiude una serie di macchine di proprietà di privati destinate ad eseguire programmi utente (*host*) collegati da una *subnet*, posseduta e gestita da una compagnia telefonica o da un ISP, che ha il compito di trasportare i messaggi da un *host* all'altro. Nella maggior parte delle WAN la *subnet* è formata da linee di trasmissione che spostano i bit tra le macchine e elementi di commutazione (*router*) che scelgono la linea di uscita su

cui inoltrare il messaggio inviato da una linea ricevente.
Nella maggior parte delle WAN la rete contiene molte linee di trasmissione, ciascuna delle quali collega una coppia di *router.* Se due *router* che non condividono la stessa linea di trasmissione vogliono comunicare devono farlo indirettamente attraverso altri *router*: quando un pacchetto viene inviato da un *router* verso un altro attraverso *router* intermedi ciascuno di essi riceve integralmente il pacchetto, che viene memorizzato finché non si libera la linea di uscita necessaria e poi inoltrato. Una *subnet* organizzata secondo questo principio è chiamata ***store-and-forward*** o ***packet-switched***.
Un altro tipo di organizzazione della *subnet* è quello delle **reti satellitari**: ogni *router* ha un antenna attraverso la quale può trasmettere e ricevere, tutti i *router* possono ascoltare l'uscita dal satellite e in alcuni casi anche le trasmissioni dei *router* che vanno nella direzione opposta. Le reti satellitari sono intrinsecamente di tipo *broadcast*.

## Reti wireless

Le **connessioni all'interno di un sistema** che coinvolgono il collegamento delle periferiche di un computer tramite segnali radio a portata ridotta (es: *Bluetooth*) nella loro forma più semplice usano il sistema *master-slave*: l'unità centrale è il *master* che dialoga con gli altri componenti e regolamenta la comunicazione degli *slave*. Le **LAN *wireless***, invece, sono sistemi dove ogni computer ha un'antenna con cui può comunicare con gli altri sistemi. Lo standard per questo tipo di reti è chiamato IEEE 802.11. Le **WAN *wireless*** sono molto simili alle LAN, ma vengono utilizzati nei sistemi su area estesa. Questa tecnologia è già arrivata alla terza generazione: mentre le prime due riguardavano la trasmissione analogica e digitale della voce, la terza generazione si occupa di trasportare voce e dati. A complemento di queste reti a bassa velocità è in sviluppo il *local multipoint service*, ovvero reti *wireless* a larga banda e copertura estesa il cui standard è lo IEEE 802.16.

## Reti domestiche

L'idea di base è che in futuro la maggior parte delle abitazioni sarà predisposta per la rete locale: ogni dispositivo domestico sarà in grado di comunicare con gli altri e tutti quanti saranno accessibili tramite Internet. Le reti di questo tipo devono avere caratteristiche fondamentali e diverse dagli altri tipi di rete: devono essere facili da installare e i dispositivi devono essere di semplice utilizzo e ad alta compatibilità, la rete deve inoltre avere una certa capacità, essere sicura e affidabile. Le reti domestiche potrebbero essere ovviamente cablate o *wireless*, ma per quanto i costi avvantaggino il secondo tipo di connessione il problema delle onde radio è che passano facilmente attraverso le barriere e quindi possono essere facilmente accessibili anche a estranei.

## Reti tra sistemi

Un *internetwork* viene creato quando si collegano tra di loro reti distinte, ovvero in linea di massima ci troviamo di fronte ad una rete di questo tipo quando diverse organizzazioni hanno pagato la costruzione di parti distinte della rete e ciascuna gestisce la propria o quando abbiamo tecnologie differenti.
Per connettere più reti tra di loro (*internetwork* o *internet*) è necessario un dispositivo chiamato ***gateway*** che stabilisce la connessione e offre servizi di conversione necessari per far si che due sistemi con *hardware* e/o *software* differenti possano comunicare tra loro.

## 1.3 SOFTWARE DI RETE

### Gerarchie dei protocolli

Per diminuire la complessità la maggior parte delle reti è organizzata come pila di *layer* costruiti l'uno sull'altro il cui scopo è quello di offrire determinati servizi agli strati di livello superiore schermandoli dai dettagli sull'implementazione dei servizi (*information hiding*). Lo strato di un computer stabilisce una comunicazione con lo stesso strato di un altro computer, le cui regole e convenzioni sono globalmente noti come protocollo di tale strato. L'insieme di strati e protocollo si chiama **architettura di rete**, mentre un elenco di protocolli usati da uno specifico computer è chiamato ***protocol stack***. Tra ciascuna coppia di strati contigui si trova un interfaccia che definisce le operazioni elementari e i servizi che lo strato inferiore rende disponibili a quello soprastante.
Per realizzare il trasferimento vero e proprio del messaggio ogni livello passa dati e informazioni di controllo allo strato immediatamente sottostante fino a raggiungere quello più basso, al di sotto del quale si trova il supporto fisico attraverso cui è possibile la comunicazione vera e propria.

### Progettazione degli strati

I problemi fondamentali da affrontare nel progetto delle reti sono presenti in più strati, come per esempio l'indirizzamento, il controllo degli errori, l'ordine dei messaggi inviati, il controllo del flusso, la lunghezza dei messaggi ricevuti, il *multiplexing* e il *routing*.

### Servizi orientati alla connessione e senza connessione

Per utilizzare un **servizio *connection-oriented*** l'utente deve innanzi tutto stabilire una connessione, usarla e quindi rilasciarla. Questo tipo di servizi si caratterizza per la possibile presenza di negoziazioni dei parametri da usare (massima dimensione del messaggio, qualità di servizio richiesta, ecc) in cui tipicamente un lato fa la proposta e il corrispondente la può accettare, rifiutare o proporre una controproposta. Un'altra caratteristica di questo tipo di servizi è che generalmente i bit arrivano nell'ordine in cui sono stati trasmessi. Al contrario, un **servizio *connectionless*** ogni messaggio è instradato nella rete in modo indipendente dagli altri e quindi è possibile che non venga mantenuto l'ordine di invio.
Ogni servizio si può classificare in base alla qualità come affidabile o meno. Un servizio affidabile non perde mai dati e di solito è implementato in modo tale che il ricevitore confermi il ricevimento di ciascun messaggio, questo però introduce appesantimenti e ritardi. I servizi affidabili orientati alla connessione possono essere di tipo ***message sequence*** (i messaggi vengono preservati) e ***byte stream*** (la connessione è un semplice flusso di byte senza divisioni tra i messaggi). Per quanto riguarda le connessioni affidabili *connectionless*, invece, possiamo avere il servizio ***datagram* con conferma** (una ricevuta di ritorno viene automaticamente spedita al mittente quando il messaggio è stato ricevuto dal destinatario) e il servizio ***request-reply*** (la sorgente trasmette un singolo datagramma che contiene una richiesta e il messaggio ricevuto dal mittente rappresenta la risposta).

### Primitive di servizio

Un servizio è specificato da un insieme di primitive che i processi utenti hanno a disposizione per accedere al servizio e che lo istruiscono a eseguire particolari azioni o riferire quelle prese da un

entità di pari stato. Se la pila di protocolli si trova nel sistema operativo le primitive sono dette di sistema e causano la commutazione in modalità kernel, che a sua volta fa prendere il controllo del computer al sistema operativo per spedire i pacchetti necessari. Le primitive per un servizio *connection-oriented* differiscono da quelle di un servizio *connectionless*.
Prima di iniziare una comunicazione il *server* esegue ***LISTEN*** per indicare che è pronto ad accettare connessioni in arrivo e il processo resta quindi bloccato fino a che non appare una richiesta di connessione. Il processo *client* esegue ***CONNECT*** per stabilire una connessione, specificano a chi connettersi ed inviando un pacchetto chiedendo al *server* di connettersi (1), e resta in sospeso fino a quando non arriva una risposta (2) generata dal codice del protocollo stesso che sblocca il *client*. Il *server* a questo punto è sbloccato ed esegue ***RECEIVE***, preparandosi ad accettare la prima richiesta. Il *client* esegue ***SEND*** per trasmettere le sue richieste (3) seguito da *RECEIVE* per ottenere la risposta, il *server* invece usa *SEND* per restituire una risposta al *client* (4) che se non ha richieste aggiuntive può usare ***DISCONNECT*** per terminare la connessione. La chiamata a questa primitiva sospende il *client* ed invia un pacchetto al *server* indicando che la connessione non è più necessaria (5), il *server* emette a sua volta un *DISCONNECT* dando conferma al *client* e rilasciando la connessione (6).
Dato che per completare questo protocollo occorrono sei pacchetti ci si potrebbe chiedere perché non si usa, invece, un protocollo *connectionless* che necessiterebbe solamente di due pacchetti: uno per al richiesta e uno per la risposta. Nel mondo reale un semplice protocollo del tipo *request-reply*, però, risulterebbe su una rete inaffidabile e spesso inadeguato.

## La relazione tra servizi e protocolli

Un **servizio** è un insieme di primitive che uno strato offre a quello superiore, ma non dice nulla di come queste operazioni siano implementate. Un **protocollo**, invece, è un insieme di regole che controllano il formato e il significato dei messaggi scambiati tra entità *peer* all'interno di uno strato, queste entità usano i protocolli per implementare le loro definizioni dei servizi.
In altri termini: i servizi si riferiscono alle interfacce tra gli strati, i protocolli invece riguardano i pacchetti scambiati tra entità di pari strato che risiedono su computer diversi.

### 1.4 MODELLI DI RIFERIMENTO

## Il modello di riferimento OSI

Questo modello è stato sviluppato su proposta dell'ISO (*International Standards Organization*) come primo passo verso la standardizzazione internazionale dei protocolli impiegati nei diversi strati ed è stato chiamato OSI (*Oper System Interconnection*) perché riguarda la connessione di sistemi che sono aperti verso la connessione con altri.
Il modello OSI ha sette strati:

**Lo strato fisico.** Si occupa della trasmissione di bit grezzi sul canale di comunicazione e le specifiche riguardano per lo più interfacce meccaniche o elettriche e temporizzazioni, oltre che il mezzo di trasmissione che sta sotto allo strato fisico.

**Lo strato *data link*.** Il suo compito è quello di rilevare gli errori di trasmissione forzando il trasmettitore a suddividere i dati d'ingresso in *data frame* che vengono trasmessi sequenzialmente e, se il servizio è affidabile, il ricevitore confermerà la corretta ricezione di ognuno tramite un

*acknowledgement frame*. Un altro obiettivo di questo strato è quello di evitare che un trasmettitore veloce saturi le possibilità di un ricevitore lento.

**Lo strato *network*.** Questo strato controlla il funzionamento della *subnet* e, in generale, i problemi che deve affrontare riguardano la qualità del servizio (ritardo, tempo di congestione, ecc).

**Lo strato trasporto.** La sua funzione essenziale è quella di accettare dati dallo strato superiore, dividerli quando necessario e passarli allo strato di *network* assicurandosi che tutti i pezzi arrivino correttamente, il tutto in maniera efficiente che isoli gli strati superiori dalle differenze di *hardware*. Lo strato di trasporto definisce il tipo di servizio che verrà offerto allo strato sessione (e quindi all'utente) al momento della connessione.

**Lo strato sessione.** Permette agli utenti su computer diversi di stabilire tra loro una sessione e offre quindi servizi come il controllo del dialogo, la gestione dei *token* e la sincronizzazione.

**Lo strato presentazione.** Consente la comunicazione tra computer differenti con differenti rappresentazioni dei dati occupandosi della sintassi e della semantica dell'informazione trasmessa.

**Lo strato applicazione.** Comprende una varietà di protocolli comunemente richiesti dagli utenti.

## Il modello di riferimento TCP/IP

Questo modello deriva da un'evoluzione della rete sperimentale ARPANET, che connetteva centinaia di università e installazioni governative tramite linee telefoniche affittate. Quando furono aggiunte reti satellitari e via radio i protocolli esistenti incontrarono difficoltà nel collegamento facendo nascere l'esigenza di una nuova architettura di riferimento, che divenne poi nota col nome di TCP/IP.

**Lo strato internet.** Il suo scopo è quello di consentire agli *host* di mandare pacchetti in qualsiasi rete e farli viaggiare in modo indipendente l'uno dall'altro fino alla destinazione. Per questo motivo lo strato internet fornisce un formato ufficiale per i pacchetti, un protocollo chiamato IP (*Internet Protocol*), provvede all'instradamento dei pacchetti e garantisce l'assenza di congestioni. Ha funzionalità simili allo strato *network*.

**Lo strato trasporto.** È stato progettato per consentire la comunicazione tra entità pari degli *host* sorgente e destinazione, ed è paragonabile allo strato trasporto OSI. A questo livello vengono definiti due protocolli: TCP (*Transmission Control Protocol*), un protocollo affidabile *connection-oriented*, e UDP (*User Datagram Protocol*) un protocollo inaffidabile *connectionless*.

**Lo strato applicazione.** I primi protocolli di questo strato gestivano un terminale (TELNET), lo scambio di file (FTP, *File Transfer Protocol*) e la posta elettronica (SMTP, *Simple Mail Transfer Protocol*), ma nel corso degli anni se ne sono aggiunti molti altri quali DNS (*Domain Name System*), NNTP (*Network News Transfer Protocol)*, HTTP (*HyperText Transfer Protocol)*, ecc.

**Lo strato *host-to-network*.** Questo strato permette all'*host* di collegarsi alla rete usando qualche protocollo, che varia in base all'*host* e alla rete, che gli permetta di spedire pacchetti IP.

## Confronto tra i modelli di riferimento OSI e TCP/IP

Entrambi sono basati su uno *stack* di protocolli indipendenti e la funzione degli strati è grosso modo simile, ma nonostante queste similitudini i due modelli di riferimento hanno molte differenze.
Nel modello OSI sono presenti tre concetti essenziali: 1) ogni strato offre un **servizio** a quello che

lo sovrasta e la definizione di servizio descrive ciò che fa lo strato e non le modalità di accesso da parte delle entità sovrastanti o quelle di funzionamento, definisce la semantica dello strato; 2) l'**interfaccia** di uno strato spiega quali sono le modalità di accesso ai processi sovrastanti specificando quali sono i parametri e i risultati, ma non dice nulla riguardo alle modalità di funzionamento interno; 3) ogni strato può usare i **protocolli** che preferisce e li può cambiare senza disturbare il *software* degli strati superiori. Il modello TCP/IP, in origine, non faceva una netta distinzione tra servizio, interfaccia e protocollo. La conseguenza è che nel modello OSI i protocolli sono nascosti meglio che nel modello TCP/IP e si possono sostituire con relativa facilità all'evolvere della tecnologia, infatti questo modello è stato concepito prima di inventare i protocolli corrispondenti cosa che lo rende altamente generico. Nel caso del TCP/IP, al contrario, prima sono arrivati i protocolli e poi fu realizzato il modello, che rappresentava una semplice descrizione dei protocolli esistenti.
Una seconda differenza tra i due modelli è il numero di strati: il TCP/IP ne ha quattro mentre l'OSI ne ha sette. Entrambi hanno gli strati di (*inter*)*network*, trasporto e applicazione, ma gli altri sono diversi.
L'ultima differenza riguarda le modalità di comunicazione. Il modello OSI supporta a livello di *network* entrambi i tipi di comunicazione ma nello strato di trasporto supporta solo servizi *connection-oriented*, il modello TCP/IP al contrario ha la modalità *connectionless* nello strato di *network* ma supporta entrambe in quello di trasporto.

## Critica del modello e dei protocolli OSI

**Poca tempestività.** Quando sono apparsi i protocolli OSI le controparti TCP/IP erano già largamente impiegate nelle università che facevano ricerca, quindi al suo arrivo non c'è stata disponibilità per supportare una seconda pila di protocolli.

**Tecnologia scadente.** Due strati (sessione e presentazione) sono quasi vuoti, mentre altri due (*data link* e *network*) sono sovraccarichi. Oltre a questo il modello OSI e le corrispondenti definizioni di servizi e protocolli hanno una complessità elevatissima e alcune funzioni (indirizzamento, controllo del flusso e di errore) compaiono più e più volte in ogni strato.

**Implementazioni carenti.** Le implementazioni iniziali del modello furono enormi, scomode e lente. Per contrasto una delle prime implementazioni del TCP/IP faceva parte dello UNIX ed era decisamente buona oltre che gratuita, di conseguenza venne rapidamente adottata sollecitando la formazione di una grande comunità di utenti.

**Incapacità politica.** In seguito all'implementazione iniziale molti ritenevano TCP/IP parte integrante di UNIX, molto amato all'epoca. OSI, d'altro canto, era largamente ritenuto un prodotto dei ministeri delle telecomunicazioni europei, della Comunità Europea e del governo USA, evocando l'immagine di un gruppo di burocrati che cercavano di spingere uno standard mediocre ostacolando gli sforzi di chi voleva ottenere qualcosa di realmente funzionante.

## Critica del modello di riferimento TCP/IP

Prima di tutto il modello non distingue in modo chiaro i concetti di servizio, interfaccia e protocollo, diventando pressoché inutile nel momento in cui bisogna progettare reti basaste su tecnologie nuove. Oltre a questo lo strato *host-to-network* non è un vero e proprio strato ma piuttosto un'interfaccia tra gli strati di *network* e *data link* e non fa distinzione tra gli strati fisico e *data link* che sono totalmente diversi tra loro. In secondo luogo è poco generale e inadatto a descrivere pile di protocolli diverse dal TCP/IP. Parlando di protocolli, IP e TCP furono pensati con

cura e implementati bene ma molti altri protocolli risolvono problemi *ad-hoc*.
Nonostante i problemi il modello OSI si è dimostrato eccezionalmente utile per discutere le reti di computer, mentre i protocolli non OSI sono diventati popolari. Al contrario il modello TCP/IP è praticamente inesistente ma i protocolli sono molto usati.

CAPITOLO 2

# LO STRATO FISICO

Lo strato fisico definisce le interfacce meccaniche, elettriche e le temporizzazioni della rete.

## 2.1 LE BASI TEORICHE DELLA COMUNICAZIONE DATI

Le informazioni possono essere trasmesse via cavo variando alcune proprietà fisiche. Rappresentando il valore di queste attraverso una funzione ad una variabile, *f(t)*, è possibile modellare il comportamento del segnale e analizzarlo matematicamente.

### Analisi di Fourier

Qualunque funzione periodica sufficientemente regolare, *g(t)*, con periodo T può essere ottenuta sommando un numero idealmente infinito di seni e coseni. Questa scomposizione è chiamata anche **serie di Fourier**. Un segnale che ha durata finita può quindi essere gestita immaginando semplicemente che esso ripeta infinite volte l'intero schema.

### Segnali a banda limitata

Nessun mezzo di trasmissione è in grado di trasmettere segnali senza perdere parte dell'energia durante il processo. Se tutti i componenti di Fourier fossero attenuati in modo uniforme il segnale risultante verrebbe ridotto in ampiezza ma non risulterebbe distorto, sfortunatamente però questo non succede e quindi l'attenuazione del segnale è causa distorsione. Di solito le ampiezze sono trasmesse senza modifiche da 0 ad una certa frequenza $f_c$ e attenuate per tutte le frequenze superiori a questo limite. L'intervallo di frequenze trasmesse senza una forte attenuazione è chiamato **banda passante**, ed è una proprietà fisica del mezzo di trasmissione che dipende dal suo spessore e dalla lunghezza. In alcuni casi viene introdotto nel circuito un filtro per limitare l'ampiezza della banda a disposizione per ogni utente, ma in ogni caso il limite imposto all'ampiezza di banda limita la velocità dei dati anche su canali perfetti. Per ovviare a questo problema esistono particolari schemi di codifica.

### La velocità massima di un canale

Se si trasmette un segnale arbitrario attraverso un filtro *low-pass* la cui ampiezza di banda è pari ad *H*, il segnale può essere ricostruito completamente prendendo solo *2H* campioni al secondo in quanto le frequenze superiori sono già state filtrate. Se sul canale è presente rumore casuale la velocità massima di trasmissione sul canale peggiora rapidamente, ma purtroppo il rumore termico causato dal movimento delle molecole del sistema è sempre presente. Il livello di rumore termico si misura rapportando la potenza del segnale a quella del rumore e misurata in dB.

## 2.2 MEZZI DI TRASMISSIONE GUIDATI

### Mezzi magnetici

Uno dei sistemi più comuni adottati per trasferire i dati da un computer ad un altro è quello di scrivere informazioni su un supporto rimovibile e utilizzare un'apposita unità installata nel computer di destinazione per leggere i dati. Questa è spesso la soluzione più economica.

### Il doppino

Il doppino (UTP, *Unshielded Twisted Pair*) è composto da due conduttori di rame isolati avvolti uno intorno all'altro in una forma elicoidale in modo tale che i campi elettromagnetici generati dai due conduttori si annullino a vicenda. I doppini si possono usare per trasmettere segnali sia analogici che digitali mentre l'ampiezza di banda dipende dal diametro del cavo e dalla distanza percorsa. Per il loro basso costo e discreto livello di prestazioni sono largamente utilizzati, specie nel sistema telefonico.
Esistono diversi tipi di doppini tra cui:

- UTP3 (250 MHz). Composti da quattro coppie di cavi, a due a due isolati e attorcigliati, raggruppate in una guaina di plastica che protegge e tiene uniti i conduttori.
- UTP5 (600 MHz). Sono simili agli UTP3 ma utilizzano più spire per centimetro in maniera tale da ridurre l'interferenza e migliorare la qualità del segnale trasmesso sulle lunghe distanze.

### Cavo coassiale

Il cavo coassiale è composto da un nucleo conduttore coperto da un rivestimento isolante a sua volta circondato da un conduttore cilindrico, solitamente realizzato con una calza di conduttori sottili, che è infine avvolto da una guaina di plastica. La costruzione e la schermatura del cavo forniscono un'ampiezza di banda vicina a 1 GHz ed un'eccellente immunità al rumore. Essendo più schermato del cavo UTP, il cavo coassiale può estendersi per distanze più lunghe e consente velocità più elevate. Per questi motivi è ancora molto utilizzato per le reti metropolitane e le televisioni via cavo.

### Fibra ottica

Un sistema di trasmissione ottico è formato da tre componenti fondamentali: la sorgente luminosa, il mezzo di trasmissione (una fibra di vetro sottilissima realizzata in silicio) e il rilevatore. Quando viene colpito dalla luce il rilevatore genera un impulso elettrico. Collegando ad un estremo della fibra una sorgente di luce ed un rilevatore all'altro si crea un sistema di trasmissione unidirezionale che accetta un segnale elettrico, lo converte e lo trasmette sotto forma di impulso luminoso. Quando un raggio luminoso passa da un materiale ad un altro si rifrange sul confine tra i due materiali. L'entità della rifrazione dipende dall'indice di rifrazione dei due materiali: per angoli di incidenza che superano un particolare valore critico la luce rimane intrappolato nella fibra e può propagarsi per molti chilometri senza dispersioni significative.
La fibra può contenere molti raggi che rimbalzano ad angoli diversi, in questo caso è detta **fibra multimodale**. Al contrario, quando il diametro della fibra è ridotto fino a poche lunghezze d'onda e la luce può propagarsi solo in linea retta è detta **fibra monomodale**.

**Trasmissione della luce attraverso la fibra.** Il livello di dispersione cromatica degli impulsi luminosi dipendono dalla loro lunghezza d'onda (0,85, 1,30 e 1,55 micron). Per evitare sovrapposizione degli impulsi diffusi in questo modo è necessario aumentare la distanza tra gli stessi, ma ciò può essere fatto solo riducendo la velocità di segnale. È stato scoperto che creando impulsi di una particolare forma (solitoni) è possibile annullare tutti gli effetti della dispersione cromatica e inviare impulsi per migliaia di chilometri senza che la loro forma subisca modifiche.

**Cavi in fibra.** Al centro del cavo si trova un nucleo (*core*) di vetro attraverso il quale si propaga la luce (50 micron nelle fibre multimodali e 8-10 micron per quelle monomodali). Il nucleo è circondato da un rivestimento in vetro (*cladding*) che ha un indice di rifrazione più basso, in maniera tale da costringere la luce a rimanere nel nucleo, che a sua volta è rivestito da una sottile fodera di plastica. Le fibre sono solitamente raggruppate in fasci protetti da guaine.
Le fibre si possono collegare in tre diversi modi: 1) tramite connettori inseriti in apposite prese, semplificando la riconfigurazione del sistema (perdita del 10-20% della luce); 2) possono essere attaccate meccanicamente in poco tempo, è anche possibile allineare meglio la luce per massimizzare il segnale (perdita del 10% della luce); 3) due pezzi di fibra possono essere fusi generando una piccola attenuazione del segnale. In tutti i casi possono presentarsi fenomeni di riflessione che interferirebbero col segnale.
Per generare il segnale possono venire impiegati due tipi di sorgenti luminose: LED (*Light Emitting Diode*) e semiconduttori laser.

**Reti in fibra ottica.** Collegarsi ad una rete in fibra è più difficile rispetto ad una Ethernet, ma gli ostacoli possono essere superati realizzando una rete ad anello composta da una serie di connessioni punto a punto. Per realizzare questa struttura si usano due tipi di interfacce: 1) l'interfaccia passiva è composta da due spine fuse nella fibra principale, ad un'estremità della spina si trova un LED o un diodo laser e all'estremità opposta un fotodiodo; 2) il ripetitore attivo, in cui la luce in entrata è convertita in segnale elettrico, rigenerata se necessario e infine riconvertita sotto forma di segnale luminoso.

**Fibre ottiche o cavi di rame?** La fibra offre una maggiore ampiezza di banda e non è influenzata da sorgenti elettriche, campi elettromagnetici e interruzioni della linea elettrica, è sottile e leggera e inoltre sostituendo cavi in rame con cavi in fibra si potrebbe recuperare il materiale (che ha un alto prezzo di vendita sul mercato) vendendolo ai raffinatori di rame. D'altro canto la fibra è una tecnologia meno nota le cui interfacce costano di più di quelle elettrice, si può danneggiare se la si piega troppo e la comunicazione bidirezionale ottica richiederebbe due fibre (o due bande di frequenza su una singola fibra) in quanto la trasmissione è intrinsecamente unidirezionale.

## 2.3 TRASMISSIONI *WIRELESS*

### Lo spettro elettromagnetico

Quando si spostano gli elettroni creano onde elettromagnetiche che si propagano attraverso lo spazio. Il numero di oscillazioni di un'onda è chiamato **frequenza** (*f*, Hz) e la distanza tra due massimi (o minimi) consecutivi è chiamata **lunghezza d'onda** (m/sec). Nel vuoto tutte le onde elettromagnetiche si propagano alla **velocità della luce** ($c = 3\times10^8$ m/sec).
Quando un'antenna di dimensioni appropriate è collegata ad un circuito elettrico le onde elettromagnetiche sono trasmesse in modo efficiente e un rilevatore posto ad una certa distanza le può captare. Le porzioni di spettro indicate come radio, microonde, infrarosso e luce visibile si

possono utilizzare per trasmettere informazioni modulando l'ampiezza, la frequenza o la fase delle onde. La luce ultravioletta e i raggi X funzionerebbero anche meglio grazie alle loro elevate frequenze, ma sono difficili da generare e da modulare, non si propagano bene attraverso gli ostacoli e sono dannose per gli esseri viventi. La quantità di informazione che un'onda elettromagnetica può trasportare dipende dalla sua **banda**, maggiore è la banda e più grande è la velocità. La maggior parte delle trasmissioni utilizza una banda a frequenza ristretta per ottenere una migliore ricezione ma in certi casi si usa una banda larga, con due varianti: 1) a spettro distribuito a frequenza variabile (*frequency hopping*), in cui il trasmettitore salta da una frequenza all'altra centinaia di volte al secondo rendendo difficile rilevare le trasmissioni e risultando quasi impossibile da disturbare; 2) a spettro distribuito a sequenza diretta (*direct sequence*), che propaga il segnale in banda larga.

## Trasmissioni radio

Le onde radio sono semplici da generale, possono viaggiare per lunghe distanze e possono attraversare facilmente gli ostacoli, per questi motivi sono largamente utilizzate per la comunicazione.
Una proprietà di questo tipo di onde è l'**omnidirezionalità**, ovvero si espandono dalla sorgente in tutte le direzioni. Grazie a questo il trasmettitore e il ricevitore non devono essere fisicamente allineati. Le proprietà delle onde radio dipendono dalla **frequenza**: alle frequenze più basse attraversano bene gli ostacoli ma la potenza diminuisce allontanandosi dalla sorgente, alle frequenze più alte tendono a viaggiare in linea retta rimbalzando contro gli ostacoli e sono assorbite dalla pioggia. A tutte le frequenze le onde radio sono soggette a **interferenze** sia da parte di dispositivi elettrici che tra gli utenti stessi.
Nelle bande VLF, LF e MF le onde radio seguono il terreno e si possono ricevere per circa 1.000 Km alle frequenze più basse e a distanze più brevi con frequenze maggiori. Nel caso della comunicazione dati il problema principale di queste bande è la loro ridotta ampiezza di banda. Nelle bande HF e VHF le onde terrestri tendono ad essere assorbite dal pianeta, ma le onde che raggiungono la ionosfera sono riflesse e tornano verso il pianeta rimbalzando.

## Trasmissione a microonde

Le onde sopra i 100 MHz viaggiano quasi il linea retta e, pertanto, si mettono a fuoco facilmente. Concentrando l'energia in un piccolo raggio si riesce ad ottenere un rapporto segnale/rumore molto più alto, ma le antenne trasmittenti e riceventi devono essere accuratamente allineate. Inoltre, se le antenne sono troppo lontane entra in gioco la curvatura terrestre e di conseguenza sono necessari dei ripetitori. Le microonde non attraversano molto bene gli ostacoli e sono soggette a *multipath fading* (divergenza nello spazio). Le bande sopra i 10 GHz sono ormai di uso comune, ma intorno ai 4 GHz si è scoperto anche che le onde vengono assorbite dall'acqua.
La comunicazione a microonde è talmente utilizzata nelle comunicazioni che si sta manifestando una grave scarsità di spettro. Questo sistema di trasmissione offre diversi vantaggi rispetto alla fibra, per esempio non richiede alcun diritto di passaggio ed è relativamente conveniente.

**La politica dello spettro elettromagnetico.** Sono stati stipulati accordi nazionali e internazionali sui soggetti e e le frequenze utilizzabili. Anche quando lo spettro viene assegnato con un particolare utilizzo c'è il problema della suddivisione delle frequenze tra i fornitori dei servizi. I tre algoritmi più utilizzati in passato sono: 1) il *beauty contest*, richiede che ogni fornitore specifici il valore della sua proposta per il pubblico interesse e in seguito gli agenti del governo sceglieranno la motivazione che appare più interessante; 2) una lotteria tra le aziende interessate; 3) la vendita

all'asta delle porzioni dello spettro elettromagnetico. Un approccio completamente diverso consiste nel non assegnare affatto le frequenze lasciando che ognuno trasmetta a piacere ma regolando la potenza utilizzata per limitare la portata delle stazioni. Seguendo questo principio la maggior parte dei governi ha riservato alcune bande di frequenza, chiamate ISM (*Industrial Scientific, Medical*) per l'utilizzo senza licenza.

### Infrarossi e onde millimetriche

Per le comunicazioni a corto raggio si usano i raggi infrarossi. Questo sistema è relativamente direzionale, economico e facile da costruire, ma non riesce ad attraversare gli ostacoli. Questo è un grande difetto, ma rappresenta un vantaggio visto che sistemi a infrarossi in stanze differenti non interferiranno tra loro. Oltretutto grazie a questo i sistemi di questo tipo sono più sicuri rispetto a quelli basati sulle onde radio.

### Trasmissione a onde luminose

La segnalazione ottica basata sui laser è intrinsecamente unidirezionale e la sua potenza rappresenta anche la sua debolezza: il raggio è molto sottile quindi richiede un'altissima precisione. Per ovviare a questo problema vengono introdotte lenti per rendere il raggio meno focalizzato. Un altro svantaggio è che i raggi laser non possono attraversare la pioggia e la nebbia senza subire modificazioni.

## 2.4 COMUNICAZIONI SATELLITARI

Nella sua forma più semplice un satellite di comunicazioni non è altro che un ripetitore di microonde che contiene diversi *transponder*, ovvero ricetrasmettitori satellitari, ognuno dei quali ascolta una parte dello spettro amplificando il segnale d'ingresso e ritrasmettendolo su un'altra frequenza per evitare interferenze con il segnale di arrivo. Questa modalità operativa si chiama ***bent pipe***.
Il periodo orbitale di un satellite varia in base alla sua orbita: più è alto il satellite e più è lungo il periodo, di conseguenza i satelliti con orbite più basse scompaiono dalla vista rapidamente e ne servono molti per fornire una copertura continua. Un altro problema che riguarda il posizionamento dei satelliti è collegato alle fasce di Van Allen, strati di particelle molto cariche intrappolate dal campo magnetico terrestre, in quanto un satellite che le attraversasse verrebbe distrutto. Per questi motivi sono state individuate tre zone in cui i satelliti possono essere collocati senza pericolo.

### Satelliti geostazionari

I satelliti **GEO** (*Geostationary Earth Orbit*) sono posti in un'orbita equatoriale circolare a 35.800 Km di altezza, in maniera tale da apparire fermo nel cielo agli osservatori terrestri. Posizionandoli a intervalli di 2 gradi ne bastano 180 per coprire tutta la superficie terrestre. Gravità solare, lunare e planetaria tendono ad allontanarli dagli *slot* e dagli orientamenti assegnati, ma l'effetto è contrastato dai motori a razzo installati a bordo. Questa continua messa a punto è chiamata ***station keeping***. Il

lungo viaggio di andata e ritorno introduce nei satelliti di questo tipo un ritardo di trasmissione di circa 250/300 msec. La proprietà più importante dei satelliti è che sono mezzi di trasmissione *broadcast* per natura, l'altra faccia della medaglia è che per proteggere le comunicazioni è necessario adottare sistemi di crittografia.
Le bande a disposizione delle società di telecomunicazioni commerciali sono la banda **Ku** (*K under*) e la banda **Ka** (*K above*). Il problema principale che riguarda queste bande è la presenza della pioggia, che tende ad assorbire le microonde corte, mentre per quanto riguarda in particolare la Ka l'attrezzatura necessaria è ancora molto costosa.
Un satellite moderno ha circa una quarantina di *transponder*, ciascuno dei quali opera come un *bent pipe*. Nei primi satelliti l'ampiezza di banda era divisa in bande di frequenza prefissate, oggi ogni raggio di transponder è diviso in *slot* temporali. I primi satelliti GEO avevano una singola emissione, che illuminava circa ⅓ della superficie terrestre, chiamata impronta. Oggi ogni satellite è dotato di più antenne e ogni raggio diretto verso il basso può essere concentrato verso una piccola area geografica, perciò possono avvenire contemporaneamente più trasmissioni nei due sensi.
Un nuovo sviluppo nel settore delle comunicazioni satellitari è rappresentato dalle microstazioni a basso costo, chiamate **VSAT** (*Very Small Aperture Terminal*). In molti di questi sistemi le microstazioni non hanno abbastanza energia per comunicare direttamente con le altre, per questo motivo è necessario installare delle stazioni terrestri (***hub***) dotate di antenne ad alto guadagno che trasmettono il traffico attraverso le stazioni VSAT.

## Satelliti su orbite medie

Compresi tra le due fasce di Van Allen si trovano i satelliti **MEO** (*Medium Earth Orbit*), che impiegano circa 6 ore per compiere un giro intorno al pianeta e di conseguenza devono essere rintracciati mentre si spostano nel cielo e nonostante coprano un'area più piccola dei GEO possono essere raggiunti da trasmettitori meno potenti.
I 24 satelliti GPS (*Global Positioning System*) sono di tipo MEO orbitano a circa 18.000 Km di altezza.

## Satelliti su orbite basse

I satelliti **LEO** (*Low Eath Orbit*) sono molto vicini alla superficie del pianeta, tanto che le stazioni terrestri non hanno bisogno di molta energia e il ritardo nelle comunicazioni è di pochissimi millisecondi.

**Iridium.** Nel 1990 Motorola chiese il permesso di lanciare in LEO i 77 satelliti del progetto Iridium, che vennero poi ridotti a 66 posizionati in maniera tale che appena un satellite spariva dalla vita un altro sarebbe apparso all'orizzonte (con 6 catene di satelliti è possibile coprire l'intero pianeta). I satelliti Iridium vennero lanciati nel 1997, e il servizio di comunicazione iniziò nel 1998. Non riuscendo a essere redditizio Iridium fu costretto alla bancarotta e venne riavviato solo nel 2001. Questo servizio fornisce un servizio di telecomunicazione a livello mondiale basato su dispositivi in grado di comunicare direttamente con i satelliti, che si trovano a circa 750 Km di altezza in orbite polari circolari. Una proprietà interessante di Iridium è che la comunicazione tra clienti avviene direttamente nello spazio, dove ogni satellite trasmette i dati al successivo.

**Globalstar.** Si tratta di un progetto alternativo a Iridium che si basa su 48 satelliti LEO che utilizzano un modello tradizionale *bent pipe*: la chiamata è ritrasmessa e raccolta da una grande stazione terrestre alla stazione più vicina al destinatario. Questo schema ha il grande vantaggio di tenere la complessità della comunicazione sulla Terra, dove è più facile da gestire, inoltre le grandi antenne delle basi terrestri consentono di adoperare telefoni di bassa potenza.

**Teledesic.** Obiettivo di questo sistema è fornire contemporaneamente a milioni di utenti Internet collegamenti a 100 Mbps in trasmissione e a 720 Mbps in ricezione basati su una piccola antenna fissa di tipo VSAT completamente indipendente dal sistema telefonico. Il progetto originale si basava su un sistema di 288 satelliti con piccola impronta organizzati in 12 piani posti appena sotto la fascia di Van Allen più bassa. Successivamente venne trasformato in un sistema di 30 satelliti a commutazione di pacchetto con impronta più grande che trasmettono in banda Ka.

### Satelliti o fibra?

Le aziende telefoniche iniziarono a sostituire le loro reti a lunga tratta con fibre ottiche introducendo servizi a banda larga come l'ADSL (*Asymmetric Digital Subscriber Line*) e smisero di far pagare prezzi altissimi per le chiamate a lunga distanza per sovvenzionare il servizio locale.
Per quanto riguarda la fibra, questa ha in linea di principio un'ampiezza di banda potenziale maggiore di quella di qualsiasi satellite ma quest'ampiezza di banda non è a disposizione della maggior parte degli utenti: le fibre installate sono utilizzate dal sistema telefonico per gestire più chiamate interurbane simultaneamente. Altri settori che la fibra non può raggiungere sono quelli della comunicazione mobile, forzatamente *broadcast*, in luoghi con terreni scarsamente dotati di infrastrutture terrestri e quando è d'importanza critica una rapida installazione.
In conclusione il sistema di comunicazione principale del futuro sarà quello terrestre basato su fibre ottiche combinato con una rete radio cellulare, ma per alcune applicazioni specializzate i satelliti sono da preferire.

## 2.5 LA RETE TELEFONICA PUBBLICA COMMUTATA

### Struttura del sistema telefonico

Il telefono fu ufficialmente inventato da A.G. Bell nel 1876. Gli apparecchi inizialmente erano venduti in coppie e spettava all'utente tirare un cavo tra i due telefoni. Questa politica di gestione si rivelò ben presto poco funzionale e quindi nacque nel 1878 la Bell Telephone Company, la prima compagnia telefonica. Il suo compito era di stendere cavi dai suoi uffici fino alle case dei clienti che, per fare una chiamata, dovevano solamente richiamare l'attenzione di un operatore all'interno della centralina che lo metteva manualmente in contatto con chi avrebbe dovuto ricevere la chiamata. Alla fine questa gerarchia di centraline crebbe fino ad arrivare il quinto livello.
Da ogni apparecchio telefonico partono due cavi in rame che si collegano direttamente alla centrale più vicina (**ultimo miglio**) dell'azienda telefonica, **centrale locale**. Se il telefono chiamato è collegato ad una centrale locale differente da quella del chiamante la chiamata viene deviata ad uno o più centri di comunicazione attigui, chiamati **centrali interurbane**, fino a che i due telefoni non risultano essere collegati alla stessa centrale.
Oggi i collegamenti locali sono costruiti con cavi UTP3 mentre le centrali di commutazione utilizzano cavi coassiali, microonde e fibre ottiche, i collegamenti locali rappresentano quindi l'ultimo tassello di tecnologia analogica del sistema. La comunicazione di tipo digitale è preferita perché rende superfluo riprodurre in modo accurato una forma d'onda analogica dopo il passaggio attraverso molti amplificatori in una chiamata a lunga distanza.

## Le politiche dei telefoni

Nel 1984 la AT&T venne divisa in AT&T *Long Lines* e 23 BOC. Questo evento fece aumentare la concorrenza e migliorò il servizio facendo di conseguenza diminuire i prezzi delle telefonate interurbane e aumentare quelli delle telefonate locali. Gli Stati Uniti vennero divisi in 164 LATA (*Local Access and Transport Area*), ad ognuna delle quali corrispondeva un un prefisso telefonico, in cui il monopolio del servizio telefonico era arrivato ad un LEC (*Local Exchange Carrier*). Tutto il traffico telefonico tra le LATA era gestito da un'azienda definita IXC (*IntereXchange Carrier*). Nel 1995 venne deciso che non era più possibile mantenere una distinzione tra i diversi tipi di aziende e venne lanciata l'idea per cui ogni società poteva offrire ai suoi clienti un singolo pacchetto integrato in maniera tale da poter competere tra di loro su servizi e prezzi. La legge che ne scaturì nel 1996 permise per la prima volta la portabilità del numero locale, permettendo quindi al cliente di cambiare fornitore del servizio telefonico senza dover per forza cambiare numero.

## I collegamenti locali: *modem*, ADSL e connessioni *wireless*

Per inviare segnali digitali attraverso una linea telefonica il computer deve convertire i dati in forma analogica per poterle trasmettere attraverso l'ultimo miglio. La conversione è eseguita attraverso un *modem*. Una volta nella centrale i dati sono riconvertiti in formato digitale prima di essere trasmessi attraverso le linee a lunga distanza. L'ISP (*Internet Service Provider*) ha potuto gestire tante connessioni quanti sono i suoi *modem* fino all'apparizione dei *modem* a 56 Kbps che hanno imposto dei cambiamenti.
I problemi principali delle linee di trasmissione sono: 1) **attenuazione** (dB/Km), ovvero la perdita di energia causata dalla propagazione del segnale verso l'esterno che dipende dalla frequenza del segnale; 2) **distorsione**, causata dal fatto che ogni componente di Fourier si propaga a velocità differente attraverso il cavo; 3) **rumore**, energia indesiderata generata da sorgenti esterne al trasmettitore.

**Modem.** Per minimizzare gli effetti che attenuazione, distorsione e rumore hanno sul segnale è meglio che questo non abbia un largo intervallo di frequenze. Le onde quadre utilizzate nei segnali digitali utilizzano un ampio spettro di frequenza e perciò rendono adatta la trasmissione in banda base (DC) solo a velocità basse e distanze brevi. Viene quindi usata la trasmissione AC introducendo un tono continuo, chiamato portante **d'onda sinusoidale**, la cui ampiezza, frequenza o fase possono essere modulate per trasmettere informazioni. Un apparecchio che accetta un flusso seriale di bit in ingresso e produce una portante modulata attraverso uno o più di questi metodi è chiamato *modem*, e si trova tra il computer e il sistema telefonico.
La **banda passante** di un mezzo di trasmissione è l'intervallo di frequenze che passa attraverso il mezzo con un'attenuazione minima. Il numero di campioni al secondo è detto ***baud-rate*** (baud), durante ogni baud viene prodotto un simbolo quindi *baud-rate* e frequenza dei simboli si equivalgono. La tecnica di modulazione utilizzata determina il numero di bit per simbolo. Le combinazioni valide di ampiezza e fase sono chiamati **diagrammi costellazione**, ogni *modem* ad alta velocità ha un suo schema di costellazione e può comunicare solo con altri *modem* che adottano lo stesso schema.
Tutti i *modem* moderni permettono di trasmettere contemporaneamente in entrambe le direzioni utilizzando frequenze diverse (***full duplex***). Una connessione che permette ai dati di scorrere in entrambi i sensi ma solamente un senso alla volta è detta ***half duplex***, mentre quella che permettere di trasmettere i dati in una sola direzione è detta ***simplex***.

**Linee DSL (*Digital Subscriber Line)*.** Quando l'accesso ad Internet divenne parte importante dell'attività commerciale le aziende telefoniche iniziarono a offrire servizi con un'ampiezza di banda superiore a quella del servizio telefonico standard, raccolti sotto il nome di **xDSL**. Nel punto

in cui ogni collegamento locale termina nella centrale locale il cavo attraversa un filtro che attenua tutte le frequenze al di sotto dei 300 Hz e sopra i 3.400 Hz, l'ampiezza di banda si considera solitamente di 4.000 Hz anche se la distanza tra i punti è di 3.100 Hz. Con l'xDSL quando un cliente si abbona al servizio la linea di ingresso viene collegata a un diverso tipo di commutatore che, non usando il filtro, rende disponibile l'intera capacità del collegamento locale.
Il primo servizio ADSL (*Asymmetric DSL*), offerto da AT&T, utilizzava la tecnica del *multiplexing* a divisione di frequenza, ovvero divideva lo spettro disponibile sui collegamenti locali in tre bande di frequenza: 1) POTS (*Plan Old Telephone Service*), 2) *upstream* e 3) *downstream*. Le offerte successive proposte da altri fornitori invece hanno seguito un approccio diverso che si è rivelato vincente: il **DMT** (*Discrete MultiTone*). Lo spettro di 1,1 MHz disponibile sui collegamenti locali viene diviso in 256 canali indipendenti: il canale 0 viene utilizzato per POTS, i canali da 1 a 5 non sono utilizzati, dei rimanenti canali uno è utilizzato per il controllo della trasmissione e l'altro per il controllo della ricezione, tutti gli altri canali sono a disposizione degli utenti. Il fornitore del servizio decide quanti canali utilizzare per la trasmissione e quanti per la ricezione e la maggior parte dei fornitori assegna al canale utilizzato per scaricare l'80/90% dell'ampiezza di banda. È possibile aumentare l'ampiezza di banda rendendo bidirezionali alcuni dei canali più alti di trasmissione aggiungendo uno speciale circuito per annullare l'effetto eco. La qualità di ogni linea è, comunque, tenuta costantemente sotto controllo e la velocità dati è regolata continuamente in maniera tale che canali differenti possono avere velocità diverse.
Per installare un servizio di questo tipo la società telefonica deve piazzare un **NID** (*Network Interface Device*) nell'edificio del cliente, a cui sarà affiancato uno ***splitter*** che divide i dati dalla banda 0-4.000 Hz utilizzata da POTS. In questo modo i dati sono instradati verso un ***modem* ADSL** e da lì al computer utilizzando una scheda Ethernet o una porta USB. Nella centrale locale è installato un analogo *splitter* che invia il segnale sopra i 26 kHz al **DSLAM** (*Digital Subscriber Line Access Multiplexer*) che contiene lo stesso tipo di processore integrato in un *modem DSL*. Il segnale digitale è prima organizzato in pacchetti e poi inviato all'ISP.
Uno svantaggio di questo tipo di configurazione è la presenza del NID e dello *splitter* nell'abitazione del cliente in quanto risulta necessario organizzare e pagare un intervento dell'azienda telefonica sul luogo. Per questo è stato creato uno standard diverso che differisce da quello precedente per l'assenza dello *splitter*: la linea telefonica è usata così com'è ma occorre interpolare un microfiltro tra ogni presa telefonica e il telefono/*modem* collegato. Il filtro per il telefono è un *low-pass* che elimina le frequenze sopra i 3.400 Hz mentre quello per il *modem* ADSL è un *high-pass* che elimina le frequenze sotto i 26 kHz. Questo sistema non è affidabile come quello basato sullo *splitter.*

## Linee e *multiplexing*

Dato che installare e gestire una connessione tra centrali di commutazione basata su una linea a banda larga costa quasi quanto una connessione basata su linea a banda stretta, le aziende telefoniche hanno sviluppato degli schemi per convogliare molte conversazioni lungo un singolo collegamento fisico.

***Multiplexing* a divisione di frequenza (FDM, *Frequency Division Multiplexing*).** Lo spettro di frequenza è diviso in bande di frequenza e ogni utente ha il possesso esclusivo di una parte di banda. I filtri limitano l'ampiezza di banda utilizzabile a circa 3.100 Hz per canale, ma quando si uniscono in *multiplexing* diversi canali a ciascuno sono allocati 4.000 Hz in modo da mantenere gli elementi ben separati. Nonostante i canali sono separati dalle cosiddette bande di guardia c'è sempre una leggera sovrapposizione tra canali adiacenti dovuta al fatto che i filtri non hanno bordi netti, a causa di ciò un forte picco di segnale sul bordo di un canale viene avvertito nel canale adiacente sotto forma di rumore non termico.
Uno standard molto comune per lo schema FDM è quello basato su 12 canali vocali a 4.000 Hz

uniti in *multiplexing* nella banda compresa tra 60 e 108 kHz. Questa unità è chiamata **gruppo**. Cinque gruppi possono essere uniti per creare un **supergruppo**, mentre l'unità successiva è chiamata ***mastergoup*** ed è composta da cinque (standard CCITT, *Comité Consultatif International Téléphonique et Télégraphique*) o dieci (sistema Bell) supergruppi.

***Multiplexing* a divisione di lunghezza d'onda (WDM, *Wavelength Division Multiplexing*).** Per i canali in fibra ottica si utilizza una variazione del *multiplexing* a divisione di frequenza in cui quattro fibre sono inserite in un combinatore ottico ed ognuna trasporta energia a diversa lunghezza d'onda. I quattro raggi sono uniti in una singola fibra condivisa prima di essere trasmessi a destinazione e nuovamente scomposti all'altra estremità. Ogni fibra in uscita contiene un corto pezzo di *core* costruito in modo da filtrare tutto tranne una lunghezza d'onda. L'unica differenza con la tecnica FDM elettrica è la presenza di un sistema ottico completamente passivo, e per questo altamente affidabile, basato su un reticolo di diffrazione. Quando il numero di canali è molto elevato e le lunghezze d'onda molto vicine tra loro il sistema è definito anche **DWDM** (*Dense* WDM).
WDM è molto popolare anche perché l'energia su una singola fibra è generalmente ampia pochi GHz e facendo viaggiare più canali in parallelo su lunghezze d'onda diverse è possibile aumentare l'ampiezza in modo lineare rispetto al numero di canali. Un altro vantaggio di questo sistema di *multiplexing* è rappresentato dalla possibilità di amplificare il segnale ogni 1.000 Km senza eseguire alcuna conversione elettro-ottica, al contrario di ciò che accadeva in passato quando era necessario farlo ogni 100 Km suddividendo e convertendo tutti i canali in un segnale elettrico, amplificare ogni singolo segnale, riconvertirlo in segnale ottico e infine ricombinare tutti i canali.

***Multiplexing* a divisione di tempo (TDM, *Time Division Multiplexing*).** Gli utenti si danno il cambio acquisendo periodicamente il possesso di tutta la banda per un tempo brevissimo. Questo sistema è largamente impiegato perché può essere gestito completamente da dispositivi elettrici digitali, ma sfortunatamente può essere utilizzata solo da segnali digitali. Per questo motivo è necessario eseguire una conversione analogico-digitale nella centrale locale prima dell'immissione sulle linee in uscita.
I segnali analogici sono digitalizzati nella centrale locale da un ***codec*** (*coder-decoder*) che genera una serie di numeri ad 8 bit. Il *codec* elabora 8.000 campioni al secondo (125 μsec/campione), visto che il teorema di Nyquist afferma che tale velocità è sufficiente a catturare tutte le informazioni dall'ampiezza di banda del canale telefonico a 4 kHz. Questa tecnica è detta **PCM** (*Pulse Code Modulation*). Quando il CCITT elaborò il suo standard 1,544 Mbps adottò 8 bit (e non 7) di dati, quantizzando il segnale analogico in 256 livelli discreti. Vengono fornite due variazioni: nel **sistema di segnalazione a canale comune** il bit aggiuntivo assume valori 1010... nei *frame* dispari e contiene informazioni di segnale per tutti i canali nei *frame* pari, mentre nel **sistema a canale associato** ogni canale ha il suo sottocanale di segnalazione privato organizzato in modo da dedicare 5 *frame* su 6 ai dati e nel sesto assegnare alla gestione dei segnali 1 degli 8 bit utente in maniera tale che 5 campioni su 6 sono lunghi 8 bit e 1 lungo 7.
Dopo aver digitalizzato il segnale si usano tecniche statistiche per ridurre il numero di bit necessari ad ogni canale. Tutti questi metodi si basano sul principio che il segnale cambia in modo relativamente lento rispetto alla frequenza di campionamento, perciò gran parte dell'informazione risulta ridondante. Una tecnica chiamata ***differential pulse code modulation*** non trasmette l'ampiezza digitalizzata ma la differenza tra il valore corrente e quello precedente. Con questa tecnica se il segnale compie un salto ampio la logica di codifica può richiedere diversi periodi di campionamento per mettersi al pari, quindi l'errore introdotto può essere ignorato. Per migliorare questo tipo di tecnologia si possono utilizzare le cosiddette **codifiche per ipotesi**, nelle quali vengono estrapolati pochi valori precedenti per predire il valore successivo e poi codificare la differenza tra il segnale reale e quello previsto. Queste codifiche risultano vantaggiose perché riducono la dimensione dei numeri da codificare e di conseguenza il numero di bit da trasmettere.

## Commutazione

Il sistema telefonico è diviso in due parti principali: l'impianto esterno alla centrale di commutazione (collegamenti locali e linee) e l'impianto interno (commutatori). Attualmente si utilizzano diverse tecniche di commutazione.

**Commutazione a circuito.** Quando viene avviata una telefonata l'apparecchiatura di commutazione posta dentro il sistema telefonico cerca di creare un percorso fisico completo tra il telefono del chiamante e quello dell'utente chiamato. L'impostazione della chiamata crea un percorso dedicato tra le due estremità, che perdura fino al termine della chiamata.
Una proprietà importante di questa tecnica è legata alla necessità di configurare un percorso da un punto all'altro prima di iniziare a trasmettere i dati. Tra la fine della composizione del numero e l'inizio dello squillo viene cercato di definire il percorso della connessione, e prima che la connessione dati abbia inizio il segnale della richiesta di chiamata deve propagarsi fino alla destinazione ed essere riconosciuto. Una volta completata l'impostazione l'unico ritardo per i dati è quello dovuto al tempo di propagazione del segnale elettromagnetico (circa 5 msec per 1.000 Km), e sempre a causa di ciò non si corre il rischio di creare una congestione in quanto una volta che la chiamata è stata messa in linea non è più possibile ricevere il segnale di occupato.

**Commutazione a messaggio.** Quando il trasmettitore ha un blocco di dati da inviare le informazioni sono passate prima alla centrale di commutazione e da lì sono instradate un salto alla volta. Ogni blocco ricevuto nella sua interessa è prima esaminato per verificare la presenza di errori e poi ritrasmesso.
Una rete di questo tipo è detta ***store and forward***. I primi sistemi di comunicazione elettromeccanici hanno utilizzato la comunicazione a messaggio per trasmettere telegrammi.

**Commutazione di pacchetto.** Le reti di questo tipo impongono un limite superiore alla dimensione del blocco dati consentendo ai pacchetti di essere temporaneamente memorizzati nella memoria principale del *router* invece che su disco. Dato che si assicurano che nessun utente monopolizzi la linea di trasmissione per troppo tempo le reti a commutazione di pacchetto riescono a gestire benissimo il traffico interattivo. Un ulteriore vantaggio di questo tipo di rete è che il primo pacchetto di un messaggio viene diviso in più pacchetti e può essere inviato prima che il secondo sia completamente arrivato, riducendo il ritardo e migliorando le capacità di trasporto.
Le differenze tra la commutazione a circuito e quella a pacchetto sono molteplici. Prima di tutto la commutazione a circuito richiede la configurazione di un circuito punto a punto prima dell'inizio della comunicazione, mentre quella di pacchetto non richiede alcuna preimpostazione visto che il primo pacchetto può essere inviato non appena disponibile. Nella commutazione a circuito la configurazione del percorso fa si che i pacchetti lo seguano e che i dati arrivino già nell'ordine corretto, al contrario nella commutazione di pacchetto i diversi blocchi dati possono seguire percorsi differenti a seconda della situazione della rete al momento della trasmissione e quindi i dati possono arrivare non in ordine. Al contrario di quello che si potrebbe pensare la commutazione a pacchetto è più resistente agli errori della commutazione a circuito in quanto se un commutatore si blocca tutti i circuiti che lo utilizzano vengono chiusi e nessun dato può essere inviato, mentre nel caso della commutazione a pacchetto i blocchi dati possono essere instradati aggirando i commutatori bloccati. La preimpostazione di un percorso permette di riservare in anticipo ampiezza di banda e questo consente di non sovraccaricare la linea, ma d'altra parte il tentativo di stabilire un circuito potrebbe non andare a buon fine se la linea è sovraccarica. L'altra faccia della medaglia è, però, che se un circuito è stato riservato ad un particolare utente e non c'è traffico da trasmettere l'ampiezza di banda di quel circuito è sprecata in quanto non può essere utilizzata per gestire altro traffico.
Una commutazione di tipo *store and forward* aggiunge ritardo in quanto il pacchetto è accumulato nella memoria del *router* e successivamente inviato al *router* successivo. Con la commutazione a circuito, invece, i bit scorrono attraverso il cavo senza interruzioni.

Un'ulteriore differenza tra i due tipi di commutazione è che nel caso della commutazione a circuito ci troviamo di fronte ad un sistema completamente trasparente in quanto trasmittente e ricevente possono utilizzare qualunque velocità, formato o metodo di *framing* senza che queste decisioni influiscano sulla portante. Nel caso della commutazione a circuito, invece, è proprio la portante a determinare i parametri alla base della trasmissione.
L'ultima differenza riguarda gli aspetti cruciali della comunicazione: con la commutazione a circuito l'accento viene posto da tempo e distanza mentre con la commutazione a pacchetto diventa un problema il volume del traffico.

## 2.6 IL SISTEMA TELEFONICO MOBILE

Esistono due tipi di telefoni *wireless*: *cordless* e cellulari. Per quanto riguarda i telefoni cellulari possono essere suddivisi il tre generazioni, ciascuna caratterizzata da una diversa tecnologia: 1) voce analogica, 2) voce digitale e 3) voce e dati digitali.
Il primo sistema mobile è stato ideato negli Stati Uniti da AT&T e affidato in mandato per l'intero paese dalla FCC (*Federal Communication Commission*). Quando il sistema mobile arrivò in Europa ogni paese ideò il suo sistema e la situazione rimase tale fino a che le PTT concordarono un singolo sistema standardizzato (GSM, *Global System for Mobile communication*) in maniera tale che qualunque telefono mobile europeo funzioni in qualunque paese europeo. Nonostante questo iniziale ritardo il sistema telefonico mobile europeo risulta più efficiente di quello statunitense per una serie di motivi. In primo luogo negli Stati Uniti i telefoni cellulari sono mischiati ai telefoni fissi perciò chi chiama non è in grado di sapere se il numero appartiene ad un telefono fisso oppure ad un cellulare e quindi nemmeno di prevedere il costo della telefonata, cosa che non accade in Europa in quanto i telefoni cellulari usano prefissi facilmente riconoscibili. Un altro aspetto da non sottovalutare è l'utilizzo molto esteso delle schede prepagate.

### Cellulari della prima generazione: voce analogica

I primi radiotelefoni mobili furono utilizzati sporadicamente per la comunicazione marittima e militare durante i primi decenni del ventesimo secolo.
Nel 1964 venne creato il primo sistema telefonico per auto, che utilizzava un singolo trasmettitore di grandi dimensioni collocato in cima ad un alto edificio e aveva un unico canale per trasmettere e per ricevere. Questi sistemi sono chiamati ***push to talk*** in quanto l'utente doveva premere un pulsante che attivava il trasmettitore e disattivava il ricevitore.
Negli anni '60 venne installato **IMTS** (*Improved Mobile Telephone System*), che utilizzava sempre un trasmettitore ad alta potenza ma questa volta le frequenze per trasmettere e per ricevere erano distinte. A causa del limitato numero di canali gli utenti spesso dovevano attendere molto tempo prima di ottenere il segnale libero, e inoltre grazie alla grande potenza dei trasmettitori sistemi adiacenti dovevano trovarsi a centinaia di chilometri di distanza per evitare interferenze.

**Sistema telefonico mobile avanzato (AMPS, *Advanced Mobile Phone System*).** Questo sistema venne chiamato TACS in Inghilterra e MCS-L1 in Giappone. In tutti i sistemi telefonici un'area geografica è divisa in celle, che in AMPS sono ampie 10-20 Km, ognuna delle quali utilizza frequenze non utilizzate dalle celle vicine. L'idea chiave è l'utilizzo di celle relativamente piccole e il riutilizzo delle frequenze di trasmissione delle celle vicine ma non adiacenti. L'architettura cellulare aumenta la capacità del sistema di almeno un ordine di grandezza, inoltre tanto più le

cellule sono piccole tanto minore è la potenza richiesta e tanto più piccoli ed economici sono i trasmettitori e gli apparecchi portatili.
Le celle di solito hanno una forma approssimativamente circolare, organizzate a gruppi di sette, e sono tutte della stessa dimensione. In un'area dove il numero di utenti è cresciuto al punto che il sistema si è sovraccaricato la potenza viene ridotta e le celle sovraccaricate sono divise in celle più piccole, chiamate microcelle, per aumentare il riutilizzo delle frequenze. Al centro di ogni cella si trova una stazione di base, che comunica con tutti i telefoni che si trovano nella cella, composta da un computer e un trasmettitore/ricevitore collegato ad un'antenna. In un piccolo sistema tutte le stazioni sono collegate ad un singolo dispositivo chiamato **MTSO** (*Mobile Telephone switching Office*) o **MSC** (*Mobile Switching Center*), mentre un sistema più grande richiede diversi MTSO collegati a MTSO di secondo livello e così via. Gli MTSO comunicano tra di loro mediante una rete di comunicazione a pacchetto.
Quando un telefono mobile abbandona fisicamente una cella la stazione di base di quella cella verifica il livello di potenza del segnale ricevuto dalle stazioni che si trovano nelle celle adiacenti e trasferisce la gestione dell'apparecchio alla cella che riceve il segnale più forte, ossia la cella in cui ora si trova il telefono. Il telefono viene informato dell'identità della nuova centrale di controllo e se durante lo spostamento era in corso una chiamata l'apparecchio viene forzato dal MTSO a passare su un nuovo canale (***handoff***, 300 msec). Nel ***soft handoff*** il telefono è acquisito dalla nuova stazione di base prima di interrompere il segnale precedente evitando perdita di continuità (l'apparecchio dev'essere in grado di sintonizzare due frequenze nello stesso momento), nell'***hard handoff*** invece la vecchia stazione di base rilascia il telefono prima che la nuova lo acquisisca.

**Canali.** Il sistema AMPS utilizza 832 canali *full duplex* ognuno composto da una coppia di canali *simplex* (832 canali di trasmissione *simplex* compresi tra 824 e 849 MHz e 832 canali di ricezione *simplex* compresi tra 869 e 894 MHz). Ognuno di questi canali *simplex* è ampio 30 kHz, perciò AMPS utilizza FDM per separarli.
Gli 832 canali sono divisi in quattro categorie: 1) controllo (base-apparecchio), per gestire il sistema (21, predeterminati dalla PROM presente nel telefono); 2) *paging* (base-apparecchio) per avvisare gli utenti mobili di chiamate in arrivo; 3) accesso (bidirezionale) per impostare la chiamata e assegnare il canale; 4) dati (bidirezionale) per voce, fax e dati. Poiché le frequenze non possono essere riutilizzate delle celle adiacenti il numero reale di canali a disposizione della voce in genere è circa 45.
Nella banda 800 MHz le onde radio sono lunghe circa 40 cm e viaggiano in linea retta, sono assorbite da alberi e terreno e rimbalzano sugli edifici generando eco e distorcendo il segnale (*multipath fading*).

**Gestione della chiamata.** Ogni telefono mobile AMPS ha un numero seriale (32 bit) e un numero di telefono di 10 cifre registrato nella PROM (prefisso, 3 cifre = 10 bit, e numero dell'abbonato, 7 cifre = 24 bit). Quando viene acceso il telefono esplora i 21 canali di controllo per trovare il segnale più potente e trasmette *broadcast* il numero seriale e il numero di telefono. Durante il normale utilizzo la registrazione del telefono mobile viene eseguita circa una volta ogni 15 minuti.
Per fare una chiamata il telefono trasmette il numero da chiamare e la propria identità attraverso il canale di accesso alla stazione base che, quando riceve la richiesta, informa la richiesta. L'MTSO cerca un canale libero da assegnare alla chiamata e quando ne trova uno trasmette il numero del canale attraverso il canale di controllo. Il telefono mobile, allora, passa automaticamente al canale vocale selezionato e attende che la persona sollevi il ricevitore.
Riguardo le chiamate in arrivo, tutti i telefoni inattivi rimangono in ascolto sul canale. Quando una chiamata viene trasmessa l'MTSO principale dell'utente chiamato riceve un pacchetto che individua il destinatario e quindi viene trasmesso un altro pacchetto verso la cella dove si trova l'utente, che trasmette *broadcast* attraverso il canale un messaggio per richiamare l'attenzione del telefono del destinatario. Alla risposta di questo sul canale di accesso la base invia un messaggio che permette al chiamato di individuare il canale vocale, il telefono inizierà a squillare.

## Cellulari di seconda generazione: voce digitale

Un sistema di seconda generazione è spesso chiamato **PCS** (*Personal Communications Services*) e può utilizzare quattro diversi standard: D-AMPS, GSM, CDMA e PDC (D-AMPS modificato per mantenere una compatibilità col sistema analogico giapponese.

**D-AMPS (*Digital*-AMPS).** D-AMPS è stato progettato per coesistere con AMPS, perciò i telefoni cellulari di prima e seconda generazione possono operare contemporaneamente nella stessa cella. In base alla combinazione di telefoni presenti in una cella l'MTSO sceglie canali analogici e quelli digitali e, se la combinazione di apparecchi cambia all'interno della cella, l'MTSO può cambiare in modo dinamico i tipi di canali. Assieme all'introduzione di questo tipo di servizio si è resa disponibile una nuova banda di frequenza per gestire l'aumento di carico: i canali in trasmissione spaziano nell'intervallo 1.850-1.910 MHz e i corrispondenti canali in ricezione occupano l'intervallo 1.930-1.990 MHz, ancora una volta a coppie. In questa banda le onde sono lunghe 16 cm e vengono utilizzati gli stessi canali a 30 kHz.
Su un telefono mobile D-AMPS il segnale vocale raccolto dal microfono viene digitalizzato e compresso da un circuito presente all'interno del telefono chiamato ***vocoder*** in maniera tale da ridurre il numero di bit trasmessi attraverso il collegamento. La digitalizzazione e la compressione eseguite nell'apparecchio offrono un enorme miglioramento, talmente elevato che tre utenti D-AMPS possono condividere una singola coppia di frequenza usando la tecnica di *multiplexing* a divisione di tempo.
La struttura di controllo di D-AMPS utilizza sei canali principali: configurazione di sistema, controllo in tempo reale e non in tempo reale, *paging*, risposta di accesso e messaggi brevi di testo. Concettualmente funziona come AMPS, l'unica differenza riguarda il modo di gestire l'*handoff*: in D-AMPS per ⅓ del tempo il cellulare misura la qualità della linea e quando scopre che il segnale si sta degradando avvisa l'MTSO che può interrompere la connessione permettendo all'apparecchio mobile di tentare di sintonizzare un segnale più forte trasmesso da un'altra stazione base (**MAHO**, *Mobile Assisted HandOff*).

**Comunicazioni GSM (*Global System for Mobile communications*).** Ogni banda di frequenza è ampia 200 kHz e un sistema GSM ha 124 coppie di canali *simplex* ognuno dei quali supporta 8 connessioni separate mediante *multiplexing* a divisione di tempo. Ad ogni stazione attiva è assegnato uno *slot* temporale su una coppia di canali. In teoria il GSM supporta 992 canali, ma molti non sono disponibili per evitare conflitti di frequenza con le celle adiacenti. Trasmissione e ricezione non avvengono nello stesso intervallo temporale in quanto le radio GSM non sono in grado di ricevere e trasmettere contemporaneamente.
Ogni *slot* TDM ha una struttura specifica e gruppi di *slot* formano un *multiframe*. Ogni *slot* è composto da un *frame* con 148 bit dati (3 bit a 0, 1 campo *information* a 56 bit più 1 bit di controllo che indica se il successivo campo è per la voce oppure per i dati, 1 campo *sync* a 26 bit utilizzato dal ricevitore per sincronizzarsi con i limiti del *frame* del trasmettitore, 1 campo *information*, 3 bit a 0). Un *frame* dati è trasmesso in 547 μsec e tra uno *slot* e l'altro è presente un intervallo di guardia di 30 μsec. Come nel caso di AMPS il tempo di elaborazione consuma gran parte dell'ampiezza di banda e dopo la correzione degli errori rimangono a disposizione della voce soli 13 kbps ma, grazie alla maggiore ampiezza di banda, la qualità della voce è migliore di quella di D-AMP.
Esaminando i 26 *frame* TDM di un *multiframe* si nota che lo *slot* 12 è utilizzato per operazioni di controllo e il 25 è riservato ad applicazioni future, perciò il traffico dell'utente ha a disposizione solo 24 *slot*. Oltre a questo tipo di *multiframe* sono utilizzati anche quelli a 51 *slot* per contenere i canali di controllo necessari a gestire il sistema: 1) il **canale di controllo *broadcast***, ovvero un flusso di continuo di dati trasmetti dalla stazione base che annuncia identità e stato del canale della stazione base; 2) il **canale di controllo dedicato** utilizzato per aggiornare la posizione, registrare il terminale nella rete e configurare la chiamata; 3) il **canale di controllo comune** che è diviso in canale di *paging* (utilizzato dalla stazione base per annunciare le chiamate in arrivo), canale ad

accesso casuale (che permette agli utenti di richiedere uno *slot* sul canale di controllo dedicato) e canale di assegnazione dell'accesso (che annuncia lo *slot* assegnato).

**CDMA (*Code Division Multiple Access*).** Grazie ad una singola società (Qualcomm) il sistema CDMA è maturato fino ad essere considerato la migliore soluzione tecnica disponibile e la base per i sistemi mobili di terza generazione. Invece di dividere l'intervallo di frequenze assegnate in canali a banda stretta CDMA permette ad ogni stazione di trasmettere per tutto il tempo attraverso l'intero spettro di frequenza. Trasmissioni multiple simultanee sono separate utilizzando la teoria della codifica e presumendo che segnali sovrapposti si sommino linearmente, la chiave di questo sistema è quindi la capacità di estrarre il segnale desiderato scartando tutto il resto.
In CDMA ogni tempo è suddiviso in *m* intervalli brevi chiamati *chip* (64 o 128 chip/bit). Ad ogni stazione è assegnato un codice *m*-bit univoco chiamato **sequenza di *chip***, per trasmettere un bit 1 la stazione invia la sua sequenza di *chip* mentre per trasmettere il bit 0 invia il complemento a uno della propria sequenza. È possibile incrementare la quantità di informazioni inviabili passando da *b* bit/sec a *mb* bit/sec solo aumentando l'ampiezza di banda di un fattore di *m*. Questo rende il CDMA una forma di comunicazione a spettro distribuito. Ogni stazione CDMA utilizza 1 MHz perciò la frequenza di *chip* è pari a 1 Mchip al secondo, dato che con meno di 100 chip per bit l'ampiezza di banda effettiva per ogni stazione è superiore a quella di FDM  il problema dell'assegnazione dei canali scompare.
Ogni stazione adotta una sequenza di *chip* univoca, tutte le sequenze sono a due a due ortogonali (il prodotto interno normalizzato di qualunque coppia di sequenze di *chip* è uguale a 0) e vengono generate utilizzando i **codici Walsh**. Detto in altri termini il numero di coppie identiche è pari al numero di coppie diverse. Quando due o più stazioni trasmettono contemporaneamente, i loro segnali bipolari si sommano linearmente. Per recuperare il flusso di bit di una singola stazione il ricevitore deve già conoscere la sequenza di *chip* della stazione: i dati si ottengono calcolando il prodotto interno normalizzato della sequenza ricevuta per la sequenza di *chip* della stazione il cui flusso di bit si desidera recuperare.
In un sistema CDMA ideale la capacità (il numero di stazioni) può essere resa arbitrariamente grande, ma nella realtà i limiti fisici riducono notevolmente la capacità. Tanto per cominciare in realtà una sincronizzazione perfetta dei *chip* è impossibile: ciò che si può fare è sincronizzare trasmettitore e ricevitore facendo emettere al primo dei due una sequenza di *chip* predefinita la cui lunghezza permette al ricevitore di allacciarsi. A questo punto tutte le altre trasmissioni saranno viste come rumore di fondo e, ovviamente, tanto più è lunga la sequenza di *chip* tanto maggiore è la probabilità di rilevarla correttamente in presenza di rumore. In secondo luogo i livelli di potenza ricevuti dalla stazione base dipendono dalla distanza dei trasmettitori: una stazione mobile che riceve dalla stazione base un segnale debole utilizza potenza maggiore di quella utilizzata da una stazione mobile che riceve un segnale forte. La stazione base può anche dare comandi espliciti alle stazioni mobili in modo da aumentare o diminuire la loro potenza di trasmissione. In ultimo solo in linea di principio se si dispone di una sufficiente capacità di calcolo il ricevitore può ascoltare tutti i trasmettitori in una sola volta ed eseguire in parallelo l'algoritmo di decodifica per ognuno di essi.

## Cellulari della terza generazione: voce e dati digitali

Nel 1992 l'ITU (*International Telecommunication Union*) pubblicò un piano di sviluppo chiamato **IMT-200** (*International Mobile Communications*) che avrebbe dovuto offrire servizi come trasmissione voce ad alta qualità, trasmissione messaggi ed applicazioni multimediali disponibili in tutto il mondo istantaneamente e con garanzie di qualità. Per consentire ai produttori di costruire un singolo apparecchio che potesse essere venuto e utilizzato ovunque nel mondo ITU concepì IMT-2000 per un'unica tecnologia.
Vi furono due possibilità. La prima fu **W-CDMA** (*Wideband*-CDMA) e venne proposta da Ericsson. Si tratta di un sistema che utilizza una modulazione a spettro distribuito a sequenza diretta, del tipo

CDMA, che opera su una banda di 5 MHz concepito per interagire con le reti GSM. Una sua caratteristica, infatti, dovrebbe permettere agli utenti di lasciare le celle W-CDMA ed entrare nelle celle GSM senza perdere le chiamate. Questo sistema venne battezzato dall'Unione Europea UMTS (*Universal Mobile Telecommunications System*). L'altro contendente fu **CDMA2000**, proposto da Qualcomm, che utilizza anch'esso un'ampiezza di banda di 5 MHz ma non è stato progettato per interagire con GSM, include una diversa frequenza di frammento rispetto a W-CDMA, un diverso tempo di *frame*, un differente spettro e una diversa tecnica di sincronizzazione.
Il problema della scelta tra le due tecnologie si basa sul fatto che l'Europa voleva un sistema che interagisse con GSM mentre gli Stati Uniti ne volevano uno che fosse compatibile con CDMA. Le vertenze legali vennero risolte quando nel 1999 Ericsson accettò di acquistare Qualcomm e le due aziende si accordarono su un singolo standard 3G, ma con diverse opzioni incompatibili che appianano sul piano formale le differenze. In attesa della conclusione dalla lotta vennero proposti sistemi definiti il **2.5G**. Uno di questi è l'**EDGE** (*Enhanced Datarates for GSM Evolution*), ovvero GSM con più bit per baud che, però, comportano un maggior numero di errori per baud. Un altro standard di questo tipo è **GPRS** (*General Packet Radio Service*), ovvero una rete a pacchetti costruita sopra D-AMPS e GSM che permette alle stazioni mobili di inviare e ricevere pacchetti IP in una cella basata su un sistema vocale.

CAPITOLO 3

# LO STRATO *DATA LINK*

## 3.1 PROGETTO DELLO STRATO *DATA LINK*

Lo strato ***data link*** si prende carico di diverse funzioni, tra cui: 1) fornire un ben definito servizio d'interfaccia per lo strato di *network*, 2) gestire gli errori di trasmissione e 3) regolare il flusso dati in modo che i dispositivi riceventi lenti non vengano sopraffatti dai trasmettitori veloci. Per raggiungere questi obiettivi il *data link* prende i pacchetti provenienti dallo strato di *network* e li incapsula in *frame* prima di trasmetterli. Ogni *frame* contiene un'*header*, un campo con il carico per contenere il pacchetto e una coda.

### Servizi forniti allo strato *network*

Il servizio principale che lo strato *data link* fornisce allo strato *network* è quello di trasferire i dati dallo strato *network* della macchina sorgente a quello della macchina di destinazione. Risulta semplice pensare ad una situazione in cui i due processi a livello *data link* comunicano tra loro direttamente tramite un protocollo.
Tre tipi di servizi che vengono comunemente forniti a questo livello sono:

- ***unacknowledged* senza connessione**, ovvero quando la macchina sorgente invia dei *frame* indipendenti alla macchina destinazione senza che quest'ultima debba dare conferma dell'avvenuta ricezione. L'uso di questa classe di servizi è legittimo quando la frequenza degli errori di trasmissione è molto bassa e nel traffico di tipo *real time* (es: reti LAN, trasmissioni vocali);

- ***acknowledged* senza connessione**, il quale continua a non usare alcun tipo di connessione ma nel quale ciascun *frame* è inviato individualmente e ne viene fatto l'*acknowledged* in modo da sapere se è arrivato correttamente a destinazione oppure no. Risulta utile per i canali di trasmissione non affidabili (es: reti *wireless*);

- ***acknowledged* con connessione**, in questo tipo di servizio ogni *frame* trasferito attraverso la connessione è numerato in maniera tale che lo strato *data link* possa garantire che l'intero pacchetto venga ricevuto correttamente e i *frame* siano ordinati. Il trasferimento dati, in questo caso, avviene in tre fasi distinte: 1) viene stabilita la connessione, il mittente e il ricevente inizializzano variabili e contatori per tenere traccia dei *frame* ricevuti; 2) uno o più *frame* vengono trasmessi; 3) la connessione viene rilasciata, liberando variabili, *buffer* e risorse.

### Suddivisione in *frame*

Per servire lo strato di *network* lo strato *data link* deve usare a sua volta un servizio che gli è offerto dallo strato fisico, che ha come compito quello di prendere un flusso di bit e cercare di portarli a destinazione. Il tipico approccio usato per lo strato *data link* è quello di suddividere il flusso di bit

in una serie di *frame* e calcolare il *checksum* per ogni *frame*. Quando il *frame* arriva a destinazione il *checksum* è ricalcolato e se è differente da quello contenuto nel *frame* lo strato *data link* sa che c'è stato un errore e prende i provvedimenti necessari.
Un metodo per realizzare la suddivisione in *frame* è quello di inserire intervalli temporali fra singoli *frame*, ma le reti difficilmente riescono a garantire la successione temporale dei segnali quindi può capitare che gli intervalli temporali vengano modificati. Data la scarsa affidabilità delle reti sono stati sviluppati altri metodi.
Nel metodo del **conteggio dei caratteri** viene usato un campo nell'intestazione per specificare il numero di caratteri del *frame* in maniera tale che quando lo strato *data link* legge questo numero sa quanti caratteri seguiranno e quindi sa dove si trova la fine del *frame*. Il problema di questo algoritmo è che il conteggio può essere alterato da un errore di trasmissione che non è possibile trovare la posizione di partenza del *frame* successivo nemmeno tramite il calcolo del *checksum*. Il secondo metodo, chiamato ***flag* byte con byte *stuffing***, aggira questo problema introducendo un *flag* per delimitare l'inizio e la fine di ogni *frame* in maniera tale che quando il destinatario perde la sincronizzazione può semplicemente cercare il nuovo *flag* per trovare la fine del *frame* corrente. In questo caso il problema si pone quando vengono trasferiti dati binari, visto che può facilmente accadere che il valore corrispondente al *flag* byte compaia naturalmente dentro ai dati. Un modo per risolvere questo problema consiste nel far inserire alla sorgente un byte di *escape* subito prima di ogni occorrenza del *flag* nei dati, che verrà rimossa dalla destinazione prima di passare i dati allo strato di *network*. Questa tecnica è chiamata **byte *stuffing*** ed è intrinsecamente legata all'uso dei caratteri a 8 bit, motivo per il quale è stato introdotto un nuovo metodo di *framing* che consenta di gestire caratteri arbitrariamente lunghi. Nel **bit *stuffing*** ogni *frame* comincia e finisce con la sequenza di bit 0111110: ogni volta che lo strato *data link* della sorgente incontra cinque 1 consecutivi nei dati inserisce automaticamente un bit con valore 0 nel flusso di uscita e quando la destinazione riceve cinque bit consecutivi con valore 1 seguiti da uno 0 automaticamente elimina lo 0.
Molti protocolli *data link* per aumentare la ridondanza usano in abbinamento la tecnica del conteggio dei caratteri con uno degli altri metodi: il *frame* è accettato come valido sono se il *flag* è presente nella posizione prevista è il *checksum* è corretto, altrimenti si procede a leggere il flusso in ingresso fino al successivo delimitatore.

## Controllo degli errori

Per assicurare l'affidabilità dell'arrivo dei dati tipicamente il protocollo richiede che la destinazione mandi indietro degli speciali *frame* di controllo che contengono degli ***acknowledgement*** relativamente ai *frame* ricevuti. Quando problemi *hardware* fanno scomparire un *frame*, però, il destinatario non reagirà in alcun molto bloccando la sorgente. Un problema di questo tipo può essere risolto introducendo un ***timer*** impostato dopo un intervallo abbastanza lungo da permettere che il *frame* giunga a destinazione, venga elaborato e che l'*acknowledgement* possa ritornare alla sorgente. Se il *frame* o l'*acknowledgement* vengono persi viene inviato nuovamente il *frame* incriminato, ma questa soluzione risulta efficace solamente nel momento in cui viene assegnato un numero di sequenza ad ogni *frame* in uscita. In caso contrario si rischia che la destinazione lo accetti più volte e lo passi ripetutamente allo strato di *network*

## Controllo di flusso

Un altro problema notevole dello strato *data link* riguarda il saper gestire una situazione in cui la sorgente vuole trasmettere i *frame* più velocemente di quanto la destinazione sia in grado di accettarli. Anche se la trasmissione non presenta errori si arriverà ad un certo punto in cui il

destinatario sarà incapace di trattare i *frame* in arrivo e comincerà a perderne alcuni.
Gli approcci più utilizzati sono due:

- il **controllo di flusso tramite *feedback***, nel quale la destinazione da alla sorgente il permesso di mandare altri dati;
- il **controllo di flusso tramite limitazione della velocità**, in cui il protocollo stesso contiene al suo interno un meccanismo che limita la velocità di trasmissione dei dati a quella della destinazione senza bisogno di *feedback*.

## 3.2 RILEVAZIONE E CORREZIONE DEGLI ERRORI

### Codici per la correzione degli errori

Sono state sviluppate due strategie di base per gestire gli errori:

- la **codifica a correzione d'errore** (*forward error correction*), nella quale viene incluso in ciascun blocco dati trasmesso una quantità di informazioni tale che permetta di ricostruire il contenuto del blocco in caso di errore (canali molto rumorosi, es: reti *wireless*);
- la **codifica a rilevazione d'errore**, che consiste nell'introdurre abbastanza ridondanza da permettere alla destinazione di capire che c'è stato un errore ma non di correggerlo (canali affidabili, es: fibre ottiche).

Normalmente un *frame* di $n$ bit consiste di $m$ bit di dati e $r$ bit ridondanti ($n = n + r$). Un *frame* di questo tipo viene chiamato ***codeword*** di $n$ bit. Il numero di bit diversi di due sequenze (determinato eseguendo l'X-OR tra due *codeword* e contando il numero di bit a 1 del risultato) è detto **distanza di Hamming**, e se due sequenze hanno distanza pari a $d$ significa che saranno necessari $d$ errori su singoli bit per convertire una sequenza nell'altra. Dato che non tutte le possibili $2^n$ *codeword* sono usate, dato l'algoritmo per calcolare i bit di controllo è possibile costruire una lista completa delle *codeword* legali e trovare due *codeword* con distanza di Hamming minima che è, per definizione, la distanza di Hamming dell'intera codifica. Per trovare $d$ errori è necessaria una codifica con distanza $d + 1$ mentre per correggere $d$ errori è necessaria una codifica con distanza $2d + 1$.
Una semplice codifica a rilevazione di errore si può realizzare aggiungendo un **bit di parità** ai dati, che viene calcolato in modo che il numero 1 nella *codeword* sia sempre pari (o dispari). Una codifica di questo tipo ha distanza 2. Per sfruttare l'efficacia della parità viene introdotta la **codifica di Hamming**, in cui i bit della *codeword* vengono numerati consecutivamente a partire dal primo bit sulla sinistra: i bit che sono una potenza di 2 sono bit di controllo, i restati bit sono riempiti con bit di dati. Ogni bit di controllo forza la parità di alcuni gruppi di bit, incluso se stesso, a essere pari (o dispari) facendo in modo che un bit sia controllato solo da bit di controllo che sono presenti nella sua espansione come somma di potenze di 2. Quando una *codeword* valida viene ricevuta la destinazione inizializza il contatore a 0 ed esamina ogni bit di controllo $k$ per vedere se la parità è corretta, in caso negativo aggiunge il valore $k$ al contatore. Se il contatore è ancora a 0 quando tutti i bit sono stati esaminati la *codeword* viene accettata come valida, in caso contrario il valore non nullo del contatore indica il numero del bit in errore. Le codifiche di Hamming riescono a correggere solo gli errori singoli.
Considerando una sequenza di $k$ *codeword* consecutive come una matrice, per correggere gli errori *burst* i dati anziché riga per riga vengono trasmessi colonna per colonna. Dopo che il *frame* è

arrivato a destinazione la matrice viene ricostruita a partire dalle colonne e, se si verifica un errore *burst* di lunghezza *k* al massimo 1 bit per *codeword* verrà toccato, sapendo che la codifica di Hamming è in grado di correggere proprio 1 errore per *codeword* l'intero blocco può essere corretto. Questo metodo dura *kr* bit di controllo per rendere immuni *km* bit dati agli errori *burst* di lunghezza al più *k*.

## Codifiche a rilevazione d'errore

Aggiungendo ai blocchi dati un singolo bit di parità nel caso di un errore *burst* di lunga estensione la probabilità che l'errore venga rilevato è solamente 0,5. Questa probabilità può essere aumentata considerevolmente. Considerando ogni blocco da trasmettere come una matrice rettangolare di *n* colonne e *k* righe viene calcolato un bit di parità separatamente per ogni colonna e aggiunto come ultima riga della matrice, che verrò poi trasmessa una riga alla volta. All'arrivo del blocco la destinazione controlla tutti i bit di parità e se uno qualunque di questi bit è errato richiede la trasmissione del blocco. Questo metodo riesce a rilevare errori di lunghezza *n* ma un errore di lunghezza *n* + 1 può passare senza essere rilevato. Se il blocco viene profondamente alterato la probabilità che ognuna delle *n* colonne abbia casualmente un valore di parità giusto è 0,5, quindi la probabilità che un blocco in errore venga accettato è pari a $2^{-n}$.
Il metodo utilizzato comunemente è detto **codifica polinomiale** (o CRC) ed è basata sul fatto di trattare sequenze di bit come dei polinomi a coefficienti che possono assumere solo i valori 0 o 1. Un *frame* di *k* bit è visto come una lista di coefficienti per un polinomio con *k* termini, di grado *k* - 1, il cui coefficiente di termine più alto è quello più a sinistra. Quando si utilizza una codifica di questo tipo la sorgente e la destinazione devono mettersi d'accordo in anticipo su un polinomio generatore *G(x)* il quale deve avere i bit di ordine più alto e più basso uguali a 1. Per poter calcolare il *checksum* di un *frame* di *m* bit che corrisponde al polinomio *M(x)* il *frame* deve essere più lungo del polinomio generatore. L'idea è quella di aggiungere un *checksum* alla fine del *frame* in modo che il polinomio rappresentato sia divisibile per *G(x)*: se c'è un resto vuol dire che c'è stato un errore di trasmissione. Una codifica di questo tipo con *r* bit di controllo è in grado di rilevare delle sequenze di errori di lunghezza al più *r*. Considerando tutte le combinazioni come ugualmente possibili la probabilità di un *frame* incorretto venga accettato come valido è $½^{r-1}$.

# 3.2 PROTOCOLLI *DATA LINK* ELEMENTARI

Nei seguenti protocolli assumeremo che negli strati fisico, *data link* e *network* siano presenti dei processi indipendenti che comunicano passandosi dei messaggi. Esamineremo il caso in cui la macchina A voglia mandare un consistente flusso di dati alla macchina B usando un servizio affidabile orientato alla connessione avendo a disposizione una quantità illimitata di dati pronti da trasmettere. Le macchine non saranno soggette a *crash*.

## Un protocollo *simplex* senza restrizioni

I dati sono trasmessi in un'unica direzione (*simplex*) e gli strati *network* della sorgente e della destinazione sono sempre pronti, il tempo per elaborare i dati può essere ignorato ed è disponibile un *buffer* infinito. Oltre a ciò il canale di comunicazione non perde mai nessun *frame*.

Il protocollo consiste di due distinte procedure, un mittente e un destinatario, che girano negli srati *data link* delle due macchine. Il mittente invia dati all'interno di un *loop* infinito alla massima velocità che gli è possibile. Questo protocollo utilizza solo in campo *info* del *frame* in quanto gli altri campi hanno a che vedere con il controllo degli errori e del flusso. Il destinatario aspetta che succeda qualcosa, e l'unica possibilità è l'arrivo di un *frame* integro. La porzione di *frame* che contiene i dati viene passata allo strato di *network* mentre lo strato *data link* torna ad aspettare un nuovo *frame*.

## Un protocollo *simplex stop-and-wait*

Si assume che il canale di comunicazione sia senza errori e che il traffico sia di tipo *simplex*, ma rimuoviamo l'ipotesi della quantità infinita di spazio *buffer* nel cui immagazzinare i *frame* in arrivo prima di elaborarli. In questo protocollo bisogna quindi prevenire che il mittente inoltri al ricevente dati trasmessi a velocità maggiore di quanto quest'ultimo sia in grado di elaborare.
In certe circostanze particolari si può inserire un ritardo nel protocollo *simplex* senza restrizioni per rallentarlo in modo da non inondare il ricevente. Una soluzione più generale, invece, consiste nel far sì che il ricevente mandi un *feddback* al mittente tramite un piccolo *frame* senza dati (*dummy*) dopo aver passato il pacchetto allo strato di *network*. Il mittente è quindi obbligato dal protocollo ad aspettare finchè non riceve l'*acknowledgement* prima di procedere e lo strato *data link* della sorgente non deve nemmeno esaminare il *frame* in arrivo in quanto l'unica possibilità è che sia un'*acknowledgement*. Anche se il traffico in quest'esempio è di tipo *simplex* i frame viaggiano in entrambe le direzioni, conseguentemente il canale di comunicazione tra i due strati *data link* deve essere in grado di trasmettere informazioni in modo bidirezionale: è necessario al più un canale *half-duplex*.
Protocolli di questo tipo sono spesso chiamati PAR (*Positive Acknowledgement with Retrasmission*) o ARQ (*Automatic Repeat reQuest*).

## Un protocollo *simplex* per canali rumorosi

Consideriamo adesso una situazione in cui il canale di comunicazione commette degli errori: i *frame* possono essere danneggiati o persi completamente. L'*hardware* della destinazione riesce ad accorgersi di questi errori calcolando il *checksum*.
Inizialmente può sembrare che inserire un *timer* all'inteno di un protocollo *stop-and-wait* possa risolvere il problema, ma lo strato di *network* attualmente non è in grado di sapere se il pacchetto è stato perso o duplicato. Quello che si rende veramente indispensabile, a questo punto, è un metodo per distinguere i *frame* originali dalle ritrasmissioni, ovvero occorre inserire un numero di sequenza nell'intestazione (1 bit). Dopo la trasmissione del *frame* la sorgente fa partire il *timer* (o esegue un *reset* nel caso in cui fosse già stato avviato) con un intervallo di tempo scelto in modo da permettere ai *frame* di arrivare a destinazione, esservi elaborati e permettere al *frame* di *acknowledgement* di tornare indietro. Dopo la sorgente resta in attesa, e le possibilità sono tre: 1) l'arrivo di un *frame* di *acknowledgement* intatto, e in questo caso la sorgente prende il successivo pacchetto dallo strato di *network* e lo mette nel *buffer* sovrascrivendolo a quello precedente, e 2) l'arrivo di un *frame* di *acknowledgement* danneggiato o 3) il *timeout* del *timer*, casi in cui viene inviato un duplicato. Ogni *frame* in arrivo con un numero di sequenza sbagliato viene scartato come duplicato, quando invece arriva un *frame* con il numero di sequenza corretto viene accettato e passato allo strato di *network*.
Questo protocollo differisce dai suoi predecessori per il fatto che sorgente e destinazione hanno delle variabili che devono essere ricordate (numero di sequenza del successivo *frame* da inviare) mentre lo strato *data link* è in stato di attesa, quindi ogni protocollo ha una fase di inizializzazione prima di entrare nel *loop*.

## 3.4 PROTOCOLLI *SLIDING WINDOW*

Nella maggior parte delle situazioni reali risulta necessario trasmettere in entrambe le direzioni. Un modo per ottenere una trasmissione *full-duplex* è quello di avere due canali di trasmissione e usare ciascuno di essi per una trasmissione *simplex* ottenendo due canali fisici separati. In questo modo, però, la banda del canale di ritorno risulta quasi interamente specata. Un'idea migliore è quella di usare lo stesso circuito per entrambe le direzioni di comunicazione mescolando i *frame* dati a quelli di *acknowledgement*, che rimarranno comunque distinuguibili guardando il campo *kind* dell'*header*. Quando arriva un *frame* dati la destinazione non invia subito un successivo *frame* di controllo, ma aspetta che lo strato di *network* gli passi un successivo pacchetto a cui viene aggiunto l'*acknowledgement* usando il campo *ack* dell'*header*. Questa tecnica è detta ***piggy backing***, e il vantaggio sta nel miglior uso della banda disponibile oltre che nel minor numero di *frame* inviati. Perché questo non interferisca con il *timer*, se il nuovo pacchetto arriva rapidamente viene fatto il *piggybacking* dell'*acknowledgement* altrimenti se al termine del periodo di attesa non è arrivato lo strato di *data link* invia un *frame* di *acknowledgement* separatamente.
L'essenza dei protocolli di tipo ***sliding window*** è che, ad ogni istante, la sorgente tiene traccia di un insieme di numeri di sequenza corrispondenti ai *frame* che è autorizzata ad inviare. Si dice che questi *frame* si trovano nella finestra di invio e rappresentano i *frame* che sono già stati trasmessi correttamente oppure che sono in transito ma non hanno ancora ricevuto l'*acknowledgement* corrispondente. In modo analogo la destinazione tiene traccia della finestra di ricezione, che corrisponde all'insieme dei *frame* che può accettare. Quando un nuovo pacchetto arriva dallo strato di *network* gli viene assegnato il numero di sequenza successivo in ordine crescente e il limite superiore della finestra viene incrementatato, quando invece arriva un *acknowledgement* si incrementa il limite inferiore. Poiché i *frame* che sono all'interno della finestra di invio potrebbero andare persi o danneggiati la sorgente deve mantenerli tutti nella sua memoria per un'eventuale ritrasmissione. Se la finestra cresce fino alla massima dimensione lo strato *data link* è costretto a chiudere forzatamente lo strato di *network* finché non si libera almeno un *buffer*.

### Un protocollo *sliding window* a 1 bit

In circostanze normali uno dei due strati *data link* parte e trasmette il primo *frame*. Quando questo arriva a destinazione lo strato *data link* controlla se è un duplicato e, se il *frame* è quello che la destinazione si aspettava, lo passa allo strato di *network* e la finestra della destinazione viene portata avanti. Il campo di *acknowledgement* contiene il numero dell'ultimo *frame* ricevuto senza errori. Se questo numero concorda con il numero di sequenza che la sorgente vuole trasmettere allora la sorgente sa di aver finito il lavoro con il *frame* memorizzato nel *buffer* e può prendere il successivo pacchetto dallo strato di *network*. Se i numeri di sequenza non concordano allora deve provare ad inviare di nuovo lo stesso *frame*.
Una situazione particolare si ha quando entrambi i soggetti mandano simultaneamente un pacchetto iniziale. Questo può accadere come risultato di *timeout* prematuri, anche se un soggetto inizia a trasmettere nettamente prima dell'altro.

### Un protocollo che usa *go back n*

Fino ad ora abbiamo supposto che il tempo necessario a trasmettere un *frame* alla destinazione sommato al tempo per l'*acknowledgement* sia trascurabile. In alcuni casi quest'ipotesi è chiaramente falsa in quanto in alcune situazioni i tempi lunghi di tragitto del *frame* possono avere

pesanti implicazioni in termini di efficientza dell'uso della banda. La soluzione è permettere alla sorgente di mandare un numero di *frame w* maggiore di 1 prima di bloccarsi. Con una scelta appropriata di *w* la sorgente è in grado di trasmettere per un tempo pari al tempo totale di transito senza arrivare a riempire la finestra.
Il bisogno di avere una finestra larga da parte della sorgente si ha quando il prodotto tra banda e ritardo di trasmissione totale è particolarmente grande: se la banda è larga anche in presenza di un piccolo ritardo la sorgente è in grado di saturare la finestra velocemente (a meno che questa non sia di grandi dimensioni), se il ritardo è grande la sorgente può saturare la finestra anche con una banda moderata. Il prodotto di questi due fattori è essenzialmente la capacità della *pipeline*, e la sorgente per poter operare alla massima efficienza ha bisogno di poterla utilizzare tutta senza fermarsi. Questa tecnica va sotto il nome di ***pipelining***.
Per il ripristino degli errori in presenza di *pipelining* sono disponibili due approcci base. Il primo è chiamato ***go back n*** e vuole che la destinazione semplicemente scarti tutti i *frame* successivi all'errore senza mandare l'*acknowledgement* per i *frame* scartati. La strategia corrisponde ad una finestra di ricezione di dimensione 1. Se la finestra della sorgente si riempie prima che il *timer* scatti la *pipeline* inizia a svuotarsi, dopo un po' di tempo la sorgente va in *timeout* e ritrasmette in ordine tutti i *frame* che non hanno ricevuto l'*acknowledgement* cominciando da quello danneggiato (o perso). Questo approccio può far perdere molta banda se la frequenza degli errori è alta.
Anche se un protocollo di tipo *back to n* non mette in *buffer* i *frame* che arrivano dopo un errore, questo non elimina del tutto il problema dell'uso del *buffer*: dato che ad un certo punto una sorgente si può trovare nella situazione di dover ritrasmettere tutti i *frame* che non hanno ricevuto *acknowledgement* deve per forza tenere traccia di tutti i *frame* inviati finché non è sicura che siano stati accettati dalla destinazione. Quando arriva un *acknowledgement* lo strato *data link* controlla se può rilasciare il *buffer* e, in questo caso, viene liberato dello spazio nella finestra. Per questo protocollo assumiamo che ci sia sempre del traffico all'indietro sul quale effettuare il *piggybacking* degli *acknowledgement*, in caso contrario questi non possono essere trasmessi. Un'altra cosa di cui necessita questo protocollo sono più *timer*, ognuno collegato ad un *frame*, che possono essere facilmente simulati via *software* usando un singolo orologio *hardware* che lancia *interrupt* periodici. I *timeout* registrati formano una *linked list* in cui ogni nodo della lista contiene il numero di intervalli (*tick*) che mancano per arrivare al *timeout*, il *frame* a cui corrisponde il *timer* e un puntatore al nodo successivo. Quando il contatore dei *tick* arriva a 0 viene generato un *timeout* e il nodo è rimosso dalla lista.

## Un protocollo che usa la ripetizione selettiva

Con la **ripetizione selettiva** un *frame* in errore viene scartato mentre i *frame* buoni ricevuti successivamente vengono messi in un *buffer*. Sorgente e destinazione mantengono una finestra di numeri di sequenza accettabili. La destinazione, inoltre, ha un *buffer* riservato per ogni numero di sequenza all'interno della finestra al quale è associato un bit che indica quando è pieno o vuoto. Quando arriva un *frame* il suo numero di sequenza è controllato per vedere se si trova all'interno della finestra e, in caso affermativo il *frame* viene accettato e memorizzato. Se, invece, la sorgente va in *timeout* solo il *frame* più vecchio senza *acknowledgement* viene ritrasmesso e, se arriva a destinazione, il mittente può passare in sequenza allo strato di *network* tutti i *frame* che ha nel *buffer*. Questa tecnica corrisponde ad avere una finestra di ricezione maggior di 1.
Dopo che la destinazione ha portato avanti la sua finestra il nuovo intervallo di numeri di sequenza validi si è sovrapposto con quello vecchio e, di conseguenza, la serie di *frame* trasmessa successivamente può essere un duplicato della precedente (se tutti gli *acknowledgement* sono stati persi) oppure una nuova serie (se tutti gli *acknowledgement*) sono stati ricevuti. Il metodo per risolvere questo problema consiste nell'essere sicuri che, quando la destinazione ha portato avanti la finestra, non ci siano sovrapposizioni con la finestra originale, ovvero la dimensione massima della

finestra dovrebbe essere la metà della sequenza di numeri. Anche il numero di *buffer* necessari è uguale alla dimensione della finestra, come pure il numero di *timer.*
All'arrivo di un *frame* danneggiato o diverso da quello atteso viene mandato indietro un *frame* di *acknowledgement* negativo (NAK), ovviamente sempre tenendo traccia di tutti i *frame* di cui ha richiesto la ritrasmissione. Se il NAK viene perso o corrotto la sorgente andrà comunque in *timeout* e ritrasmetterà il *frame* che se arriverà sbagliato farà partire un *timer* ausiliario. Al raggiungimento del *timeout* sarà inviato un ACK per sincronizzare nuovamente lo stato corrente della sorgente e della destinazione.

## 3.6 ESEMPI DI PROTOCOLLI *DATA LINK*

### HDLC (*High-level Data Link Control*)

HDLC è orientato a bit e usa il *bit stuffing* per la trasparenza dei dati. Tutti i protocolli orientati a bit sono delimitati da una sequenza flag (01111110) e usano la seguente struttura di *frame*: 1) campo *address* (8 bit), indica il terminale; 2) campo *control* (8 bit), viene usato per i numeri di sequenza e gli *acknowledgement*; 3) campo *data*, contiene qualsiasi tipo di informazione e può avere lunghezza arbitraria; 4) campo *checksum* (16 bit), è il codice di ridondanza. Il *frame* di lunghezza minima contiene tre campi per un totale di 32 bit e il protocollo usa una sequenza di 3 bit, quindi in ogni istante possiamo avere fino a 7 *frame* in attesa di *acknowledgement*. I *frame* possono essere di diversi tipi:

- ***frame* di informazione:** il campo *seq* è il numero di sequenza del *frame*; il bit P/F (*Poll/Find*) è usato quando un computer vuole interrogare un gruppo di terminali, per forzare l'altra macchina a inviare immediatamente un *frame* di tipo supervisione senza aspettare di avere del traffico all'indietro per eseguire *piggyback* o in connessione con i *frame* senza numero; il campo *next* è usato per il *piggyback* dell'*acknowledgement* del successivo *frame* non ancora ricevuto;

- ***frame* di supervisione:** si distinguono a seconda del valore del campo *type*. 0 è di tipo RECEIVE READY ovvero un *acknowledgement* usato per indicare il prossimo numero di sequenza atteso; 1 è di tipo REJECT ovvero un NAK che usato per indicare che è stato rilevato un errore di trasmissione e che la sorgente deve ritrasmettere tutti i *frame* a partire da quello indicato nel campo *next*; 2 è di tipo RECEIVE NOT READY e serve per generare l'*acknowledgement* di tutti i *frame* fino a quello indicato nel campo *next* (escluso); 3 è di tipo SELECTIVE REJECT e richiede la trasmissione solo del *frame* specificato;

- ***frame* senza numero:** viene usata a volte per informazioni di controllo ma può anche trasportare dati per conto di servizi senza connessione non affidabili.

### Lo strato *data link* in Internet

Internet consiste di una serie di macchine individuali (*host* e *router*) e nell'infrastruttura di comunicazione che le connette. Le comunicazioni punto-punto sono usate principalmente per connettere le varie sottoreti di cui è composta Internet e per la connessioni dei singoli individui a Internet attraverso la normale linea telefonica usando un *modem*.

**PPP (*Point-to-Point Protocol*) - il protocollo punto-punto.** Tre caratteristiche del PPP meritano una particolare menzione: 1) un metodo di *framing* che permette di delimitare in modo non ambiguo la fine e l'inizio di un *frame*, il cui formato permette anche di gestire la rilevazione degli errori; 2) un protocollo di collegamento per gestire la connessione, il test della linea, negoziare le opzioni di collegamento e gestire la disconnessione in modo pulito (LCP, *Link Control Protocol*); 3) una modalità per negoziare le opzioni relative allo strato *network* in modo indipendente dall'implementazione di tale strato (NCP, *Network Control Protocol*).
Per riassumere il PPP è un meccanismo di *framing* multiprotocollo adatto alla trasmissione dati via *modem*, linee seriali HDLC, SONET e altri strati fisici. Supporta la rilevazione degli errori, la negoziazione delle scelte, la compressione dell'intestazione e, opzionalmente, la trasmissione affidabile con un formato di *frame* tipo HDLC.

CAPITOLO 4

# IL SOTTOSTRATO MAC (*Medium Access Control*)

## 4.1 IL PROBLEMA DELL'ASSEGNAZIONE DEL CANALE

### Assegnazione statica del canale in LAN e MAN

Per l'assegnazione di un singolo canale tra più utenti si utilizza tradizionalmente il ***multiplexing* a divisione di frequenza** (FDM), quindi se ci sono *N* utenti la banda è divisa in *N* parti uguali e ogni utente ne riceve una. Se il numero di utenti è piccolo e costante e ognuno ha un volume di traffico pesante FDM è un meccanismo di assegnazione semplice ed efficiente, quando però il numero di utenti che trasmettono informazioni è molto elevato e variabile oppure quando il traffico è irregolare FDM presenta alcuni problemi. Dividere il singolo canale disponibile in sottocanali statici è una soluzione intrinsecamente inefficiente in quanto il problema di fondo è che quando alcuni utenti sono inattivi la loro banda è semplicemente persa.
Lo stesso ragionamento che si applica a FDM vale anche per la tecnica di ***multiplexing* a divisione di tempo** (TDM): ad ogni utente viene assegnato un *N*esimo intervallo temporale che, se non utilizzato, viene sprecato.

### Assegnazione dinamica del canale in LAN e MAN

Vi sono cinque premesse fondamentali al problema dell'allocazione:

- **modello della stazione:** il modello è composto da *N* stazioni (o terminali) indipendenti ognuno con un programma (o utente) che una volta generati i *frame* in trasmissione rimane bloccata fino a che il *frame* non è stato trasmesso con successo;
- **presupposto del canale singolo:** un solo canale assicura ogni comunicazione;
- **presupposto della collisione:** due *frame* trasmessi contemporaneamente si sovrappongono temporalmente e il segnale ne risulta distorto (collisione), questo evento è rilevabile da tutte le stazioni ed è l'unico che può causare errori;
- **tempo continuo:** la trasmissione di *frame* può iniziare in qualunque istante;
  **tempo diviso in intervalli:** il tempo è diviso in *slot* e la trasmissione di un *frame* coincide sempre con l'inizio dell'intervallo;
- **occupazione del canale verificabile:** prima di tentare la trasmissione le stazioni sono in grado di capire se il canale è occupato;
  **occupazione del canale non verificabile:** le stazioni non sono in grado di capire se il canale è occupato e si limitano a trasmettere il *frame*.

## 4.2 PROTOCOLLI AD ACCESSO MULTIPLO

### ALOHA

**ALOHA puro.** Questo è un sistema detto a contesa, in quanto più utenti possono condividere un canale in un modo che può causare conflitto. Si basa sull'idea di fondo di consentire agli utenti di trasmettere ogni volta che hanno dati da inviare. Per quanto riguarda il problema delle collisioni che potrebbero danneggiare i *frame*, questo può essere risolto grazie alla proprietà di *feedback* della trasmissione *broadcast*, ovvero un trasmettitore potrà scoprire ascoltando il canale se il suo *frame* è andato distrutto. Se non è possibile ascoltare il canale durante la trasmissione si deve adottare un sistema di *acknowledge* in modo che il trasmettitore resti in attesa per un intervallo di tempo casuale prima di ripetere la trasmissione. Altre caratteristiche del sistema ALOHA sono la dimensione uniforme dei *frame* e il fatto che ogni volta che due *frame* tentano di occupare contemporaneamente il canale si verifica una collisione che danneggia entrambi gli elementi.

***Slotted* ALOHA.** È un metodo che venne inventato per raddoppiare la capacità di un sistema ALOHA puro. Questo approccio consiste nel dividere il tempo in intervalli discreti, dove ogni intervallo corrisponde a un *frame*: un computer per inviare dati deve attendere l'inizio dello *slot* successivo, con la conseguenza che ora due trasmissioni o collidono completamente all'interno dello stesso *slot* o non collidono affatto.
Questo metodo venne utilizzato a pieno dopo l'invenzione dell'accesso ad Internet via cavo, quando si presentò il problema di allocare un canale condiviso da più utenti in competizione.

### Protocolli ad accesso multiplo con rilevamento della portante

**CSMA persistente e non persistente (*Carrier Sense Multiple Access)*.** Nel protocollo **CSMA 1-persistente** quando una stazione è pronta a trasmettere ascolta il canale e, se il canale è occupato, aspetta fino a quando non si libera. In caso di collisione la stazione rimane in attesa di un intervallo casuale prima di ritentare la trasmissione. Il principale problema di questo protocollo è dovuto al ritardo di propagazione, ovvero c'è la possibilità che subito dopo l'inizio di una trasmissione da parte di una stazione un'altra stazione sia pronta ad inviare dati e controlli il canale. Se il segnale della prima stazione non ha ancora raggiunto la seconda quest'ultima potrebbe ritenere il canale libero e iniziare a trasmettere i propri dati causando una collisione. Nel **CSMA non persistente**, invece, se il canale è occupato la stazione non esegue un controllo continuo per trasmettere subito il proprio *frame* ma attende un intervallo di tempo casuale prima di ripetere l'algoritmo. Questo meccanismo permette di utilizzare meglio il canale ma allunga i ritardi. Il protocollo **CSMA p-persistente** si applica ai canali divisi in intervalli temporali e si basa sul fatto che quando il canale è libero una stazione trasmette subito con una probabilità $p$ e rimanda fino all'intervallo successivo con probabilità $q$=1-$p$, il processo si ripete fino a che il *frame* non è stato trasmesso o un'altra stazione non inizia a trasmettere.

**CSMA con rilevamento delle collisioni.** Questo protocollo è detto **CSMA/CD** (*CSMA/Collision Detection*). Se due stazioni iniziano a trasmettere contemporaneamente, entrambe rileveranno la collisione quasi immediatamente. Invece di completare la trasmissione dei rispettivi *frame* ormai irrimediabilmente danneggiati le stazioni interrompono bruscamente la trasmissione, risparmiando tempo e banda, e attendono un periodo di tempo casuale prima di ritrasmettere i dati. Le collisioni possono essere rilevate confrontando con il segnale trasmesso la potenza o la larghezza d'impulso del segnale ricevuto, è quindi un processo analogico.

## Protocolli senza collisione

Una volta che la stazione ha acquisito il canale, con CSMA/CD non ci possono essere collisioni. Questo problema, però, può ancora presentarsi durante il periodo di contesa del canale. I protocolli senza collisione risolvono la contesa senza generare collisione.

**Un protocollo a mappa di bit.** Ogni periodo di contesa è composto da *N* intervalli e la stazione *j* può annunciare il possesso di un *frame* da inviare inserendo un bit 1 nello *slot j*. Una volta trascorsi tutti gli *N* intervalli tutti sanno quali sono le stazioni che vogliono trasmettere e può iniziare la trasmissione in ordine numerico. Poiché tutti sono d'accordo su chi sarà il prossimo non ci sarà mai nessuna collisione e, dopo che l'ultima stazione ha trasmesso il proprio *frame*, ha inizio un altro periodo di contesa di *N* bit.. Questo tipo di protocollo fa parte dei **protocolli a prenotazione**, in quanto il desiderio di trasmettere è comunicato a tutti prima di iniziare la trasmissione vera e propria.

**Conteggio binario.** Lo scopo di questo protocollo è di adattare il metodo della mappa di bit a reti composte da molte stazioni. Una stazione che desidera utilizzare il canale deve comunicare a tutti il proprio indirizzo sotto forma di stringa binaria partendo dal bit di ordine più elevato. I bit che occupano la stessa posizione negli indirizzi di stazioni diverse sono elaborate mediante l'operatore logico OR e, per evitare conflitti, la stazione rinuncia non appena si accorge che è stata sovrascritta da un 1 una posizione di bit che nel proprio indirizzo vale 0.

## Protocolli a contesa limitata

Questo tipo di protocolli combina le proprietà dei protocolli esaminati precedentemente in modo da crearne uno capace di usare il metodo della contesa per ottenere un ritardo limitato a basso carico e il metodo senza collisioni per raggiungere una buona efficienza di canale nelle situazioni a carico più elevato.

**Il protocollo *Adaptive Tree Walk*.** Nel primo intervallo di contesa che segue una trasmissione senza collisioni, l'intervallo 0, tutte le stazioni possono tentare di acquisire il controllo del canale. In caso di collisione durante l'algoritmo analizza l'intera struttura ad albero partendo dal basso per individuare tutte le stazioni pronte. Ogni intervallo di bit è associato a qualche particolare nodo dell'albero e, in caso di collisioni, la ricerca continua in modo ricorsivo con i figli posti a sinistra del nodo. Se un intervallo di bit è libero oppure se una sola stazione trasmette durante quel periodo la ricerca del suo nodo può interrompersi perché tutte le stazioni pronte sono state individuate.

## Protocolli LAN *wireless*

Il problema principale per questo tipo di reti è il fatto che prima di avviare una trasmissione una stazione avrebbe interesse a sapere se c'è attività attorno al ricevitore e CSMA permette solo di scoprire se c'è attività intorno alla stazione che sta analizzando la portante. Nel caso di un cavo tutti i segnali si propagano in tutte le stazioni, quindi nel sistema può avvenire una sola trasmissione alla volta, mentre al contrario in un sistema basato su onde radio a portata ridotta possono aver luogo contemporaneamente trasmissioni multiple se tutte hanno destinazioni diverse e se queste destinazioni sono oltra la portata l'una dell'altra.

**MACA e MACAW.** Nel **MACA** *(Multiple Access With Collision Avoidance)* il trasmettitore incita il ricevitore a trasmettere un *frame* RTS (*Request To Send*) di 30 byte in modo che le stazioni che si trovano nelle vicinanze rilevando questa trasmissione evitino di inviare dati durante la trasmissione

imminente del *frame* di dati più grande. La stazione ricevente risponde al mittente con un *frame* CTS (*Clear To Send*) in modo da incitarlo a continuare la trasmissione. In caso di collisione un trasmettitore che non ha avuto successo, ovvero che non riceve il CTS nel tempo previsto, aspetta per un intervallo di tempo casuale prima di ripetere l'operazione. Nel **MACAW** (MACA *for Wireless*) sono stati risolti alcuni problemi, come il fatto che in assenza di *acknowledge* nello strato *data link* i *frame* perduti non sono trasmessi fino a che lo strato di trasporto non rileva la loro assenza, introducendo un ACK dopo ogni *frame* dati trasmesso con successo e introducendo la capacità di rilevamento della portante in modo da impedire alle stazioni di trasmettere un *frame* RTS quando una stazione nelle vicinanze ne sta inviando un altro alla stessa destinazione. Inoltre è stato deciso di eseguire il controllo di *backoff* separatamente per ogni flusso dati invece che per ogni stazione ed è stato aggiunto un meccanismo che consente alle stazioni di scambiare informazioni sulla congestione.

## 4.3 *ETHERNET*

### Cablaggio *Ethernet*

Il cavo più vecchio è il modello **10base5**, chiamato anche *thick Ethernet*, la cui notazione indica che opera a 10 Mbps e utilizza un sistema di segnali a banda base che può supportare segnali lunghi fino a 500 metri. Le connessioni sono realizzate generalmente mediante spine a vampiro, spingendo uno spillo nel nucleo centrale del cavo coassiale. In questo caso un *transceiver*, contenente circuiti elettronici che gestiscono il rilevamento della portante e delle collisioni, è fissato intorno al cavo per mantenere stabile in contatto con il nucleo interno. Un cavo di discesa collega il trasmettitore a una scheda di interfaccia installata nel computer, che contiene un *chip controller* che trasmette e riceve *frame*, si occupa di assemblare i dati nel giusto formato, elaborare i codici di controllo dei *frame* in uscita e verificare i codici di controllo di quelli in ingresso.
Il secondo cavo in ordine di tempo è stato il cavo **10base2**, o *thin Ethernet*. Le connessioni, in questo caso, sono realizzate usando connettori BNC standard che formano giunzioni a "T". Questo nuovo cavo era molto più economico e semplice da installare rispetto al suo predecessore, ma ogni segmento poteva essere lungo al massimo 185 metri e supportare non più di 30 macchine, anche in questo caso comunque i circuiti elettronici si trovano nella scheda di controllo e ogni stazione ha il proprio *transceiver*. Con l'avvento di questa nuova tecnologia sono state sviluppate tecniche che aiutano a rintracciare i guasti, come la TDR (*Time Domain Reflectory*) in cui un impulso viene emesso nel cavo e se colpisce un ostacolo o raggiunge la fine del mezzo viene generato un eco. Cronometrando l'intervallo trascorso tra la trasmissione dell'impulso e la ricezione dell'eco è possibile individuare l'origine dell'eco stesso.
I problemi associati alla ricerca dei difetti nel cavo hanno condotto i sistemi verso un differente sistema di cablaggio, dove ogni stazione è collegata via cavo ad un concentratore centrale (*hub*) e vengono usati doppini telefonici e lo schema è detto **10base-T**. In questo caso non c'è nessun cavo condiviso, c'è solo l'*hub* al quale ogni stazione si collega attraverso un cavo dedicato. In questa configurazione è facile individuare le interruzioni della linea, ma lo svantaggio è rappresentato dalla lunghezza massima dei cavi che partono dall'*hub* che è di soli 100 metri.
Il quarto tipo di cavi è detto **10base-F** e utilizza le fibre ottiche. Questa è un'alternativa costosa ma offre un'eccellente immunità alle interferenze, e quindi una buona sicurezza, e consente di collegare edifici distanti o *hub* molto lontani.

## Codifica *Manchester*

Per risolvere il problema dei ricevitori di determinare senza ambiguità il punto iniziale, finale e centrale di ogni bit senza fare riferimento a impulsi esterni è stata inventata la **codifica *Manchester***. In questo tipo di codifica ogni periodo di bit è diviso in due intervalli uguali: l'1 binario è inviato scegliendo il livello di tensione alto durante il primo intervallo e un livello basso durante il secondo, mentre lo schema contrario è utilizzato per trasmettere lo 0 binario. Con questa codifica si ha la certezza che ogni periodo di bit ha una transizione nel punto centrale, caratteristica che aiuta il ricevitore a sincronizzarsi con il trasmettitore. Lo svantaggio della codifica *Manchester* è che occupa il doppio della banda della codifica elementare, perché gli impulsi sono larghi la metà. La **codifica *Manchester* differenziale** è una variante in cui il bit 1 è indicato dall'assenza di una transizione all'inizio dell'intervallo, al contrario del bit 0. Uno schema di questo tipo richiede dispositivi più complessi, ma offre una maggiore immunità ai rumori. Tutti i sistemi *Ethernet* adottano la codifica *Manchester* perché è più semplice.

### Il protocollo del sottostrato MAC *Ethernet*

Ogni *frame* comincia con un campo *preamble* di 8 byte in cui ognuno contiene lo schema di bit 10101010.
Il *frame* contiene due indirizzi da 6 byte che rappresentano rispettivamente la destinazione e la sorgente. Il bit più elevato nel campo *destination address* è 0 per gli indirizzi ordinari e 1 per gli indirizzi di gruppo (trasmissione *multicast*). Quando l'indirizzo è composto da tutti i bit a 1 è riservato per la trasmissione *broadcast*, ovvero è accettato da tutte le stazioni della rete. Per quanto riguarda il secondo bit più elevato, è utilizzato per distinguere gli indirizzi locali da quelli globali. Gli indirizzi locali sono assegnati da ogni amministratore di rete e non hanno alcun significato all'esterno della rete locale, mentre gli indirizzi globali sono assegnati centralmente da IEEE per garantire che nessuna stazione utilizzi lo stesso indirizzo globale di un'altra stazione.
Il campo successivo, *type*, indica al ricevitore cosa ne deve fare del *frame*.
Dopo a questo arriva il campo *data*, lungo dai 46 ai 1.500 byte. A questo campo viene imposta una lunghezza minima per due motivi: prima di tutto *Ethernet* richiede che i *frame* validi siano lunghi almeno 64 byte (il caso contrario il campo *pad* viene utilizzato per riempire il *frame*), e in secondo luogo per impedire ad una stazione di completare la trasmissione di un *frame* breve prima che il primo bit abbia raggiunto la fine del cavo e quindi per evitare collisioni. Per questo ultimo motivo al crescere della velocità di rete la lunghezza minima del *frame* deve aumentare o, proporzionalmente, deve diminuire la lunghezza massima del cavo.
L'ultimo campo *Ethernet* è il *checksum* ovvero un codice hash dei dati di 32 bit di tipo CRC (*Cyclic Redundancy Check*), che esegue solo il rilevamento degli errori e non si occupa della correzione.
Quando IEEE creò lo standard *Ethernet* il comitato apportò due modifiche al formato DIX: il campo *preamble* venne ridotto a 7 byte utilizzando come SoF (*Start of Frame*) l'ultimo byte per mantenere la compatibilità con gli standard precedenti e il campo *type* divenne il campo *lenght*. Con questa nuova struttura il ricevitore non riusciva a capire che cosa si sarebbe dovuto fare con il *frame* in arrivo, quindi venne aggiunta alla parte dei dati una piccola intestazione che fornisce quest'informazione.

## L'algoritmo di *backoff* esponenziale binario

Dopo una collisione il tempo è diviso in intervalli discreti la cui lunghezza è uguale al tempo di propagazione di andata e ritorno nel caso peggiore sul mezzo trasmissivo. In generale dopo $i$ collisioni viene scelto un numero casuale compreso tra 0 e $2^i$-1 e si salta quel numero di intervalli.

Questo algoritmo è chiamato ***backoff* esponenziale binario** ed è stato scelto perché si adatta dinamicamente al numero di stazioni che tentano di trasmettere. Poiché l'intervallo di scelta casuale cresce esponenzialmente con il numero di collisioni avvenute l'algoritmo assicura un basso ritardo quando poche stazioni collidono e garantisce la risoluzione della collisione in tempo ragionevole quando la collisione coinvolge molte stazioni.
Il CSMA/CD non fornisce alcun meccanismo di riconoscimento dell'errore e poiché la semplice assenza di collisioni non garantisce l'arrivo corretto dei bit, per rendere affidabile la comunicazione è necessario che la destinazione verifichi il codice di controllo e, se il valore è corretto, invii un *frame* di *scknowledge* alla sorgente. Per rendere più veloce il meccanismo di conferma non bisogna far altro che riservare alla stazione di destinazione il primo intervallo di contesa successivo alla trasmissione che ha avuto successo, ma sfortunatamente lo standard non lo prevede.

## Prestazioni di *Ethernet*

Aumentando la banda della rete o la distanza si riduce l'efficienza per una data dimensione di *frame*. Sfortunatamente gran parte della ricerca sull'*hardware* di rete punta proprio ad aumentare il prodotto tra queste due grandezze in quanto gli utenti vogliono ampiezze di banda più grandi su lunghe distanze. Questo suggerisce che lo standard *Ethernet* implementato in questa maniera potrebbe non essere il miglior sistema per queste applicazioni.

## *Ethernet* commutata

Gestire gli incrementi di carico sulle reti *Ethernet* è possibile attraverso uno ***switch***. Quando una stazione vuole trasmettere un *frame Ethernet* invia allo *switch* un *frame* standard. La scheda dello *switch* che riceve il *frame* può controllare se il pacchetto è destinato ad una delle stazioni collegate alla stessa scheda, in caso positivo il *frame* è copiato direttamente sul corrispondente connettore mentre in caso negativo è trasmesso verso la destinazione attraverso la scheda di *backplane* ad alta velocità.
Quando due macchine connesse alla stessa scheda di linea trasmettono *frame* nello stesso momento la gestione delle collisioni dipende interamente da come è stata costruita la scheda stessa. Quando tutte le porte della scheda sono collegate insieme a formare una LAN integrata le collisioni al suo interno vengono rilevate e gestite come le collisioni che avvengono su una rete CSMA/CD in quanto ogni scheda cosituisce il proprio dominio di collisione indipendente dagli altri. Con una sola stazione per dominio di collisione, le collisioni sono impossibili e le prestazioni migliorano. Con un secondo tipo di scheda ogni porta input dispone di un buffer di memoria, perciò i *frame* sono memorizzati al loro arrivo nella RAM integrata sulla scheda. Questa soluzione permette a tutte le porte input di ricevere e trasmettere *frame* contemporaneamente per garantire operazioni parallele *full duplex*, risultato che non si può ottenere con un CSMA/CD su un singolo canale. Tutto ciò garantisce che ogni porta sia un dominio separato, perciò non possono esserci collisioni.
Poiché il commutatore si aspetta di ricevere *frame Ethernet* standard su ogni porta è possibile utilizzare alcune porte come concentratori. Quando l'*hub* riceve i *frame* questi si contendono il mezzo di trasmissione e quelli che risultano vincitori arrivano al commutatore e sono commutati sulla linea di output corretta attraverso il *backplane* ad alta velocità.

## *Fast Ethernet*

Per aumentare la velocità della connessione a 10 Mbps differenti gruppi industriali proposero due nuove LAN ottiche basate su topologie ad anello: la **FDDI** (*Fiber Distributed Data Interface*) e la

***Fibre Channel***. Entrambe furono utilizzate come reti dorsali, ma a causa della complessa gestione della stazione e degli alti costi non raggiunsero mai il *desktop*. L'insuccesso delle LAN ottiche aprì un varco a varietà di *Ethernet* che supportavano velocità superiori a 10 Mbps.
Nel 1992 IEEE riunì il comitato 802.3 dando il mandato di creare una LAN più veloce e la proposta vincente fu quella di mantenere lo standard 802.3 invariato, aumentando solo la velocità. Il comitato decise di procedere verso la definizione di una *Ethernet* ottimizzata per alcuni motivi fondamentali: 1) mantenere la compatibilità con le LAN *Ethernet* esistenti, 2) il timore che un nuovo protocollo avrebbe potuto creare problemi imprevisti e 3) il desiderio di raggiungere l'obiettivo prima che la tecnologia cambiasse. Il risultato fu 802.3u, chiamato da tutti ***fast Ethernet***. L'idea alla base di questo protocollo era quella di mantenere tutti i vecchi formati di *frame*, interfacce e regole procedurali riducendo il tempo di bit da 100 nsec a 10 nsec.
Un'altra scelta fondamentale riguardava la tipologia di cavi da utilizzare. Lo schema chiamato **100Base-T4** utilizza una velocità di segnale di 25 MHz, solo il 25% più veloce rispetto ai 20 MHz di *Ethernet* standard, ma ha come vantaggio quello di utilizzare doppini di categoria 3. Dato che ogni ufficio disponeva di almeno quattro cavi di questo tipo collegati ad una centralina telefonica distante non più di 100 metri utilizzando questo tipo di cavi era possibile collegare i computer con *fast Ethernet* senza dover ricablare l'intero edificio, ma l'altra faccia della medagli era che un doppino di categoria 3 non riesce a trasportare segnali di 200 Mbaud (100 Mbps con codifica Manchester) per 100 metri. La seconda opzione, chiamata **100Base-TX**, era un modello basato su cavi di categoria 5 in cui le stazioni possono trasmettere e ricevere contemporaneamente a 100 Mbps. L'ultimo schema, **100Base-FX**, utilizza un cavo di fibra multimodale per ogni direzione per creare un sistema *full duplex* con 100 Mbps di velocità e distanza minima tra una stazione e l'*hub* che può raggiungere i 2 Km.
Con gli schemi 100Base-T è possibile utilizzare due tipi di connessione, *hub* e *switch*, mentre poiché i cavi 100Base-FX sono troppo lunghi per l'algoritmo di collisione *Ethernet* standard devono essere collegati a *switch* in modo tale che ognuno sia un dominio di collisione verso se stesso. Non sono consetiti *hub* con 100Base-FX.

## Gigabit *Ethernet*

Subito dopo la conclusione dello sviluppo di *fast Ethernet* il comitato 802 iniziò a lavorare su un'*Ethernet* ancora più veloce, 802.3z, denominata **gigabit *Ethernet***. Gli obiettivi del comitato erano: rendere *Ethernet* 10 volte più veloce mantenendo la compatibilità con tutti gli standard *Ethernet* esistenti offrendo servizi *datagram* senza *acknowledge* di tipo *multicast* e *unicast* adottando lo stesso schema di indirizzamento a 48 bit già in uso e mantenendo lo stesso formato di *frame*.
Tutte le configurazioni *Ethernet* gigabit sono punto-punto. Gigabit *Ethernet* supporta due modalità di operative: *full duplex* e *half duplex*. La modalità normale è quella ***full duplex***, utilizzata quando c'è uno *switch* centrale collegato ai computer del perimetro. In questa configurazione tutte le linee hanno un buffer quindi ogni computer o *switch* è libero di inviare *frame* quando vuole e, dato che non si verifica contesa, non si utilizza il protocollo CSMA/CD quindi la lunghezza del cavo dipende dalla forza del segnale. Gli *switch* sono liberi di gestire più velocità e l'autoconfigurazione è supportata proprio come in *fast Ethernet*. L'altra modalità operativa, ***half duplex***, è utilizzata quando i computer sono collegati a un *hub*. L'*hub* è sprovvisto di buffer dove memorizzare i *frame* in arrivo quindi il dispositivo al suo interno collega elettricamente tutte le linee per simulare il cavo *multidrop* utilizzato nella *Ethernet* classica. In questa modalità le collisioni sono ancora possibili, perciò è richiesto il protocollo CSMA/CD. Poiché il *frame* minimo può essere trasmesso 100 volte più velocemente che nella *Ethernet* classica la distanza minima risulta 100 volte più corta: 25 metri. Il comitato 802.3z ha considerato inaccettabile questo raggio e ha aggiunto allo standard due funzionalità che aumentano la portata: la prima è la ***carrier extension*** che dice all'*harddware* di

aggiungere dei dati di riempimento dopo il *frame* normale in modo da estendere la dimensione del pacchetto fino a 512 byte e la seconda, chiamata ***frame bursting***, permette al trasmittente di inviare una sequenza concatenata di più *frame* in una singola trasmissione. Le nuove funzionalità estendono il raggio della rete fino a 200 metri.
*Ethernet* gigabit supporta sia cavi di rame che in fibra. Le configurazioni **1000Base-SX** e **1000Base-LX** si appoggiano a fibre rispettivamente multimodali e mono/multimodali in cui la sorgente luminosa è un laser che può avere due diverse lunghezza d'onda. Per quanto riguarda invece i cavi in rame ci sono la configurazione **1000Base-CX**, che utilizza cavi di rame schermati che devono competere con la fibra per quanto riguarda le alte prestazioni e con gli UTP per quanto riguarda i bassi costi, e la **1000Base-T** che utilizza il capo UTP di categoria 5.
Questo standard utilizza nuove regole di codifica sulle fibre: la codifica Manchester è troppo difficile da realizzare dal punto di vista della banda, quindi è stato scelto un nuovo schema chiamato **8B/10B** basato su *fibre channel*. Ogni byte da 8 bit è codificato sulla fibra usando 10 bit e le *codeword* consentite sono selezionate seguendo due regole: nessun *codeword* può avere più di quattro bit identici consecutivi e più di sei 0 o sei 1. Queste scelte permettono di ottenere un numero sufficiente di transizioni che garantiscono che il ricevitore rimanga sincronizzato con il trasmettitore, oltre che per tenere il numero di 0 nella fibra più vicino possibile a quello degli 1.
*Ethernet* gigabit basata su 1000Base-T utilizza uno schema di codifica diverso, poiché sul cavo in rame è troppo difficile mantenere una temporizzazione dei dati di 1 nsec. Questa soluzione utilizza quattro doppini di categoria 5 per consentire la trasmissione in parallelo di quattro simboli codificati usando uno dei cinque livelli di tensione disponibili.
Questa variante di *Ethernet* supporta il controllo del flusso, che avviene quando l'apparato su un estremo della connessione trasmette un *frame* di controllo speciale all'altro dicendo di sospendere la trasmissione per un certo periodo di tempo. I *frame* di controllo sono normali *frame Ethernet* in cui i primi due byte del campo dati danno il comando mentre i byte successivi forniscono i parametri.

### Retrospettiva su *Ethernet*

*Ethernet* è utilizzata da più di 20 anni  e non ha alcun serio concorrente in vista, perciò è probabile che continuerà a dominare il mercato per molti anni. Probabilmente questa longevità dipende dalla semplicità e dalla flessibilità. "Semplice" si traduce in affidabile, economico e facile da mantenere in quanto non richiede l'installazione di alcun *software* e non dipende da tabelle di configurazione difficili da gestire.

## 4.4 LAN *WIRELESS*

### La pila di protocolli 802.11

I protocolli utilizzati da tutte le varianti di 802 hanno una struttura simile. Lo strato fisico corrisponde più o meno allo strato fisico OSI, ma in tutti i protocolli lo strato *data link* è diviso in due o più sottostrati: il sottostrato **MAC** (*Medium Access Control*) che stabilisce il metodo di allocazione del canale e il sottostrato **LLC** (*Logical Link Control*) che ha il compito di nascondere le differenze tra le varianti di 802 e renderle indistinguibili allo strato di rete.
802.11 specifica tre tecniche di trasmissione supportate nello strato fisico: il metodo a infrarossi e le tecniche FHSS e DSSS che utilizzano un sistema radio a bassa potenza. Per aumentare la banda disponibile sono state introdotte due nuove tecnologie, chiamate OFDM e HR-DSSS.

## Lo strato fisico di 802.11

Ognuna delle cinque tecniche di trasmissione consentite permette l'invio di un *frame* MAC da una stazione ad un'altra, ma le soluzioni differiscono sia per la tecnologia utilizzata sia per le velocità supportate.
La variante a infrarossi utilizza una trasmissione diffusa a 0,85 e 0,95 micron e supporta due velocità: 1 Mbps e 2 Mbps. Nonostante l'elevata sicurezza garantita dal fatto che i segnali infrarossi non possono attraversare i muri, questa tecnica non è molto popolare per colpa della banda scarsa e del fatto che la luce del sole interferisce con i segnali infrarossi. **FHSS** (*Frequency Hopping Spread Spectrum*) utilizza 79 canali, larghi 1 MHz ciascuno. Viene usato un generatore di numeri pseudocasuali per produrre la sequenza con cui le frequenze si susseguono in maniera tale che tutte le stazioni salteranno contemporaneamente nelle stesse frequenze a patto di usare lo stesso seme per generare i numeri casuali e di mantenere la sincronizzazione. La casualità di FHSS rappresenta un modo efficace di allocare lo spettro di banda e permette, inoltre, di ottenere un po' di sicurezza. Sulle lunghe distanze questa variante dimostra di essere abbastanza resistente al *multipath fading* e risulta relativamente insensibile anche alle onde radio, il suo svantaggio principale è la banda ridotta. Anche il terzo sistema di modulazione, **DSSS** (*Direct Sequence Spread Spectrum*) è limitato a 1 o 2 Mbps. Il suo schema somiglia a quello del CDMA.
La prima LAN *wireless* ad alta velocità, 802.11a, utilizza **OFDM** (*Orthogonal Frequency Division Multiplexing*) per distribuire fino a 54 Mbps nella banda dei 5 GHz. Questo sistema utilizza 52 frequenze, 48 delle quali per i dati e 4 per la sincronizzazione. La divisione del segnale in tante bande strette offre vantaggi vitali rispetto all'utilizzo di una singola banda larga, quali una migliore resistenza alle interferenze con banda stretta e la possibilità di usare bande contigue. Questa tecnica ha una buona efficienza di spettro in termini di bit/Hz e una buona resistenza al *multipath fading*. Il passo successivo è **HR-DSSS** (*High Rate*-DSSS) ed è un'altra tecnica a diffusione di spettro che raggiunge la velocità di 11 Mbps nella banda 2,4 GHz. Le velocità supportate dalla 802.11b sono 1, 2, 5,5 e 11 Mbps e mentre le due velocità più basse funzionano a 1 Mbaud usando la modulazione di fase per mantenere la compatibilità con DSSS, le ultime due velocità operano a 1,375 Mbaud. All'atto pratico la velocità operativa è quasi sempre 11 Mbps e, nonostante sia più lenta dell'802.11a, il campo d'azione è circa sette volte più ampio. Una versione evoluta, chiamata 802.11g, utilizza il metodo di modulazione OFDM come 802.11a ma opera nella più ristretta banda dei 2,4 GHz come 802.11b.

## Il protocollo del sottostrato MAC di 802.11

Con *Ethernet* una stazione si limita ad aspettare che il mezzo di trasmissione diventi silenzioso per iniziare a trasmettere. Con il *wireless* questa situazione non regge a causa del problema della stazione nascosta e della stazione esposta grazie ai quali viene messo in luce il fatto che non tutte le stazione sono all'interno del campo radio delle altre e le trasmissioni che avvengono in una parte della cella possono non essere ricevute in un'altra parte della stessa e viceversa. Inoltre la maggior parte delle radio è *half duplex*. Per tutti questi motivi 802.11 non utilizza CSMA/CD.
802.11 supporta due modalità operative: la principale è **DFC** (*Distributed Coordination Function*) e non utilizza alcun tipo di controllo centrale e la seconda, opzionale, è **PFC** (*Point Coordination Function*) che usa una stazione base per controllare tutta l'attività della cella. Quando adotta DFC, 802.11 utilizza un protocollo chiamato **CSMA/CA** (CSMA/*Collision Avoidance*) in cui sono controllati sia il canale fisico che quello virtuale. Questo protocollo permette due modalità operative. Nella prima la stazione controlla se il canale è libero e, in caso positivo, emette l'intero *frame* mentre in caso contrario il trasmettitore rimanda l'operazione fino a quando il canale diventa libero. In caso di collisione le stazioni coinvolte utilizzano l'algoritmo di *backoff* esponenziale binario. La seconda modalità si basa su MACAW e utilizza il controllo di presenza del canale virtuale. Quando una stazione si trova nel campo del trasmettitore riceve il *frame* RTS e, intuendo

che qualcuno sta per iniziare una trasmissione di dati, attende fino al termine dello scambio. Grazie alle informazioni passate attraverso l'RTS la stazione può calcolare la durata della sequenza perciò rivendica per sé una specie di canale virtuale, NAV (*Network Allocation Vector*). Viceversa una stazione che si trova nel campo del ricevente rileva il *frame* CTS e di conseguenza attiva il segnale NAV per sé.
A differenza delle reti cablate, le reti *wireless* sono rumorose e inaffidabili di conseguenza la probabilità che un *frame* arrivi a destinazione senza problemi diminuisce con la lunghezza del *frame*. Per risolvere il problema dei canali rumorosi questo standard ammette la frammentazione dei *frame* in parti più piccole, ognuna dotata del proprio *checksum* di controllo. I frammenti, la cui dimensione è fissata dalla stazione base, sono numerati e ricevono l'*acknowledge* individualmente con un protocollo *stop-and-wait*. Questa tecnica aumenta la capacità del canale perché permette di ritrasmettere solo i frammenti danneggiati invece dell'intero *frame*. Bisogna tenere presente che il meccanismo NAV tiene ferme le altre stazioni fino all'*acknowledge* successivo e per consentire la trasmissione senza interferenza di un intero *burst* di frammenti si usa un altro meccanismo. Tutto questo si applica alla modalità DFC, mentre per quanto riguarda la modalità PFC una stazione base sonda le altre stazioni chiedendo se hanno *frame* da trasmettere. Dato che l'ordine di trasmissione è completamente controllato dalla stazione base non avviene mai alcuna collisione. Il meccanismo principale adottato dalla stazione di base è quello della trasmissione *broadcast* periodica di un *frame* di segnalazione che contiene i parametri di sistema e invita le nuove stazioni a effettuare la registrazione al servizio di interrogazione. Appena una stazione aderisce per una specifica cadenza riceve una certa frazione di banda. Il tutto consente di dare garanzie sulla qualità del servizio.
802.11 si occupa anche della gestione dell'energia dei dispositivi *wireless*, in particolare la stazione base può obbligare una stazione mobile ad attivare la modalità di sospensione ed occorre quindi che memorizzi in un buffer i dati diretti alle stazioni se queste sono state disattivate.
DCF e PCF possono coesistere dentro la stessa cella e ciò è possibile solo grazie ad un'attenta definizione dell'intervallo tra i *frame*. L'intervallo più breve è chiamato **SIFS** (*Short InterFrame Spacing*) ed è utilizzato per concordare i turni tra le parti coinvolte nella singola conversazione, se la stazione autorizzata a rispondere non sfrutta questa possibilità e l'intervallo **PIFS** (*PCF InterFrame Spacing*) finisce la stazione base può inviare un *frame* di segnalazione o di interrogazione. Se la stazione base non ha nulla da dire e trascorre un intervallo **DIFS** (*DCF InterFrame Spacing*) qualunque stazione può tentare di acquisire il controllo del canale. L'ultimo intervallo, **EIFS** (*Extended InterFrame Spacing*) è utilizzato solo da una stazione che ha ricevuto un *frame* danneggiato o sconosciuto e serve per annunciarlo.

## Servizi

Lo standard 802.11 definisce nove servizi che ogni LAN *wireless* conforme deve fornire, divisi in due categorie. I cinque servizi di distribuzione riguardano la gestione dell'appartenenza alla cella e l'interazione con le stazioni poste al di fuori della cella.

1. **Associazione:** è utilizzato dalle stazioni mobili per effettuare la connessione alle stazioni base;
2. **Separazione:** la stazione mobile o quella base possono separarsi interrompendo la relazione;
3. **Riassociazione:** una stazione mobile può cambiare la propria stazione base preferita;
4. **Distribuzione:** stabilisce come saranno instradati i *frame* verso la stazione base;
5. **Integrazione:** gestisce la traduzione dal formato 802.11 al formato richiesto dalla destinazione.

I quattro servizi stazione si occupano, invece, dell'attività dentro la singola cella.

1. **Autenticazione:** solo le stazioni che si autenticano ricevono l'autorizzazione a trasmettere dati;
2. **Invalidamento:** per farle lasciare la rete una stazione precedentemente autenticata viene invalidata;
3. **Riservatezza:** gestisce la codifica e la decodifica dei dati che sono stati cifrati per garantire la riservatezza;
4. **Trasferimento dati:** modo per trasmettere e ricevere dati.

## 4.6 *BLUETOOTH*

Ericsson, IBM, Intel, Nokia e Toshiba si unirono a formare un SIG (*Special Interest Group*) per sviluppare uno standard *wireless* che avrebbe dovuto permettere il collegamento tra dispositivi vari mediante un sistema radio *wireless* a basso costo, a bassa potenza e portata ridotta. Il progetto venne chiamato ***Bluetooth*** e il suo ambito ben presto si espanse fino ad invadere il settore delle LAN *wireless*. Questo spostamento rende lo standard più utile ma crea concorrenza con l'802.11, tanto più che i due sistemi interferiscono elettricamente l'uno con l'altro.
Le specifiche *Bluetooth* descrivono un sistema completo, che parte dallo strato fisico e arriva allo strato applicazione. Il comitato IEEE 802.15, che aveva adottato il documento *Bluetooth* come punto di partenza e aveva iniziato a lavorare su di esso, sta standardizzando solo lo strato fisico e lo strato *data link* mentre il resto della pila dei protocolli non è contemplato. Le due versioni non sono identiche, ma si spera che presto convergano in uno standard.

### Architettura *Bluetooth*

L'unità base del sistema *Bluetooth* è la ***piconet***, composta da un nodo *master* e da non più di sette nodi *slave* attivi situati entro un raggio di 10 metri. Un insieme di *piconet* interconnesse è chiamato ***scatternet***. Oltre ai nodi *slave* la rete può contenere fino a 255 nodi sospesi (dispositivi a cui il *master* ha imposto di attivare uno stato di bassa alimentazione) che possono solamente rispondere a una richiesta di attivazione o ad un segnale trasmesso dal *master*. Esistono altri due stati di alimentazione intermedi detti *hold* e *sniff*.
Il cuore della *piconet* è costituito da un sistema TDM centralizzato in cui il *master* controlla il *clock* e decide quale dispositivo può comunicare in ogni intervallo temporale. Non è ammessa alcuna comunicazione diretta tra nodi *slave*.

### Applicazioni *Bluetooth*

La specifica *Bluetooth* v1.1 nomina 13 applicazioni specifiche, chiamati profili, supportate e fornisce a ogni applicazione una diversa pila di protocolli.

| Nome | Descrizione |
|---|---|
| Accesso generico | Procedure per la gestione del collegamento |
| Scoperta del servizio | Protocollo per scoprire i servizi offerti |
| Porta seriale | Sostituzione cavo seriale |
| Scambio di oggetti generico | Definisce la relazione *client*/*server* per lo spostamento degli oggetti |
| Accesso LAN | Protocollo tra un computer portatile e una LAN fissa |
| Accesso *dial-up* | Permette ad un computer portatile di chiamare attraverso un telefono mobile |
| Fax | Permette ad un apparecchio fax mobile di comunicare con un telefono mobile |
| Telefonia *cordless* | Collega un telefono alla sua base |
| Intercomunicanti | Ricetrasmittenti digitali |
| Auricolare *wireless* | Comunicazione vocale senza mani |
| Invio oggetto | Permette di scambiare semplici oggetti |
| Trasferimento *file* | Gestisce il trasferimento di *file* |
| Sincronizzazione | Permette a un PDA di sincronizzarsi con un altro computer |

## Lo strato radio di *Bluetooth*

Lo strato radio sposta i bit dal nodo *master* al nodo *slave* e viceversa. Questo sistema è a bassa potenza e ha una portata di 10 metri, opera sulla banda dei 2,4 GHz dividendola in 79 canali larghi 1 MHz. La modulazione è di tipo FSK (*Frequency Shift Keying*) con 1 bit per Hz, che raggiunge la velocità di 1 Mbps anche se gran parte dello spettro è consumato dal *checksum*. Per assegnare i canali in modo imparziale si utilizza la tecnica dello spettro distribuito a frequenza variabile, ovvero tutti i nodi della *piconet* eseguono contemporaneamente il cambio di frequenza seguendo la scelta del *master.*
Poiché 802.11 e *Bluetooth* operano sugli stessi canali possono interferire uno con l'altro. *Bluetooth* esegue i salti di frequenza molto più velocemente di 802.11, perciò è più probabile che un dispositivo di questo tipo danneggi una trasmissione 802.11 che non il contrario.

## Lo strato *baseband* di *Bluetooth*

Lo strato *baseband* di *Bluetooth* assomiglia molto ad un sottostrato MAC, infatti trasforma il flusso di bit grezzi in *frame* e definisce alcuni formati chiave. Nella sua forma più semplice il *master* di ogni *piconet* definisce una serie di intervalli temporali: le trasmissioni del *master* iniziano negli intervalli pari mentre quelle degli *slave* in quelli dispari. Questo è un meccanismo di *multiplexing* a divisione di tempo tradizionale in cui i *frame* possono occupare 1, 3 o 5 intervalli. Le temporizzazioni del sistema *frequency hopping* lasciano ai circuiti radio un tempo di assestamento per raggiungere la stabilità che può essere ridotto, ma a caro prezzo con la conseguenza che i *frame* più lunghi sono molto più efficienti di quelli che occupano un singolo intervallo.
Ogni *frame* è trasmesso attraverso un canale logico, chiamato *link*, stabilito tra il *master* e lo *slave*. Il *link* di tipo **ACL** (*Asynchronous ConnectionLess*) è utilizzato per dati a commutazione di pacchetto disponibili a intervalli irregolari. Il traffico di questo tipo è trasmesso in modo *best effort*, ovvero non viene offerta alcuna garanzia di consegna. Uno *slave* può avere un solo collegamento ACL con il proprio *master.* L'altro collegamento è di tipo **SCO** (*Synchronous Connection Oriented*), a questo tipo di canale è assegnato un intervallo fisso in ogni direzione. Questo *link* è utilizzato per i

dati in tempo reale, infatti i *frame* inviati attraverso collegamenti di questo tipo non vengono mai ritrasmessi ma si utilizza piuttosto un meccanismo di correzione degli errori in avanti per migliorare l'affidabilità. Un nodo *slave* può avere tre collegamenti SCO con il proprio *master.*

### Lo strato L2CAP di *Bluetooth*

Lo straato L2CAP svolge tre funzioni principali: accetta pacchetti grandi fino a 64 KB provenienti dalgi strati superiori e li divide in *frame* per la trasmissione, gestisce il *multiplexing* e il *demultiplexing* da più sorgenti di pacchetti e gestisce i requisiti di qualità del servizio durante l'impostazione dei collegamenti e durante le normali operazioni.

### La struttura del *frame* di *Bluetooth*

Il *frame* di *Bluetooth* inizia con un **codice di accesso** che identifica il nodo *master* e consente agli *slave* che si trovano nel raggio d'azione di due *master* di identificare il traffico di rispettiva competenza. Segue un'**intestazione** di 54 bit contenente i classici campi del sottostrato MAC. Poi c'è il campo **dati**, grande fino a 2.744 bit.
Per quanto riguarda l'intestazione questa contiene il campo **indirizzo** che identifica il destinatario del *frame* tra gli otto dispositivi attivi. A seguire il campo ***type*** che identifica il tipo di *frame*, di correzione degli errori utilizzato nel campo dati e il numero di intervalli occupati dal *frame*. Il bit ***flow*** è attivato da un nodo *slave* quando non può ricevere dati perché il buffer è pieno, il bit ***acknowledgement*** è utilizzato per aggiungere al *frame* un ACK, il bit ***sequence*** per numerare i *frame*. Seguono gli 8 bit utilizzati per il ***checksum***. L'intera intestazione è lunga 18 bit ed è ripetuta tre volte per formare l'intestazione di 54 bit. Dalla parte del ricevente un circuito esamina tutte e tre le copie di ogni bit: se le copie sono identiche il bit è accettato, altrimenti la maggioranza vince.
Per il campo dati dei *frame* ACL si usano più formati, mentre i *frame* SCO sono più semplici: il campo dati occupa sempre 240 bit, anche se sono definite tre varianti che consentono 80, 160 o 240 bit di carico utile reale mentre il resto è usato per la correzione degli errori.

## 4.7 COMMUTAZIONE NELLO STRATO *DATA LINK*

Per connettere diverse LAN tra di loro è necessario un dispositivo chiamato ***bridge***, che opera sullo strato *data link* esaminando gli indirizzi che provengono da questo strato per eseguire l'instradamento. Al contrario, i *router* esaminano gli indirizzi nei pacchetti e instradano i dati in base a queste informazioni.

### *Bridge* tra 802.x e 802.y

Immaginando che un *host* debba trasmettere ad un altro *host* che si trova su una rete di tipologia differente ma collegata alla LAN *wireless*, per prima cosa il pacchetto in questione scende fino al sottostrato LLC acquisendone l'intestazione. Il pacchetto passa quindi al sottostrato MAC dove riceve un'intestazione del tipo della rete da cui proviene. L'unità viene poi trasmessa attraverso lo spazio ed arriva alla stazione base, che si accorge che i dati devono essere trasferiti su una rete

differente da quella di partenza. Quando il pacchetto arriva al *bridge* che collega le due reti, il *frame* parte dallo strato fisico e si sposta verso l'alto. Nel sottostrato MAC del *bridge* l'intestazione della rete di provenienza viene strappata via e il pacchetto nudo viene trasferito nel sottostrato LLC.
I problemi fondamentali che si incontrano in questo tipo di connessioni sono innanzi tutto riguardanti il fatto che ogni LAN utilizza un diverso formato di *frame* e di conseguenza qualunque operazione di copia tra LAN diverse richiede una nuova formattazione che consuma capacità di calcolo del processore, esige l'elaborazione di un nuovo *checksum* e introduce la possibilità di errori non rilevati causati da bit difettosi presenti nella memoria del *bridge*. Il problema più grave che riguarda questo punto è relativo alle differenti lunghezze massime del *frame*: i *frame* che non possono essere inviati perché troppo grandi devono essere scartati per mantenere la trasparenza. Le LAN interconnesse, poi, non sempre funzionano alla stessa velocità. Questo significa che quando si invia una lunga sequenza di *frame* da una LAN veloce ad una più lenta il *bridge* non sarà in grado di trasmetterli alla stessa velocità con cui li ha ricevuti e quindi li deve memorizzare in un buffer sperando di non esaurire la memoria. Un altro punto importante riguarda la sicurezza, e in particolare il fatto che i servizi di cifratura disponibili sulle reti *wireless* vanno persi quando il traffico entra in una *Ethernet*. Peggio ancora, se una stazione *wireless* utilizza la crittografia nello strato *data link* la rete *Ethernet* non è in grado di decifrare i pacchetti in alcun modo, ma questo problema può essere risolto eseguendo la codifica a livello più alto. In ultimo abbiamo la qualità del servizio. *Ethernet* non ha alcuna nozione di qualità del servizio, quindi il traffico proveniente dalle altre reti perderà automaticamente la sua qualità quando entra in una rete di questo tipo.

## *Internetworking* locale

Un *bridge* trasparente opera in modalità promiscua, ossia accetta ogni *frame* trasmesso su ogni LAN a cui è collegato. Quando arriva un *frame* il *bridge* deve decidere se scartare i dati o inoltrarli e, nel secondo caso, deve stabilire su quale LAN immettere il *frame*. La decisione è presa confrontando l'indirizzo di destinazione con le informazioni riportate in una tabella *hash* memorizzata nel dispositivo, nella quale sono elencate tutte le possibili destinazioni e viene specificato a quale linea di output appartiene.
Quando *bridge* sono collegati alla rete per la prima volta tutte le tabelle sono vuote. Poiché non sanno dove si trovano le varie destinazioni, i *bridge* utilizzano un algoritmo di *flooding* detto **apprendimento all'indietro** in cui è previsto che ogni *frame* proveniente da una destinazione sconosciuta venga inviato a tutte le LAN connesse al *bridge* meno quella di input. Col tempo i *bridge* imparano dove si trovano le destinazioni e una volta scoperte le LAN a cui appartengono le stazioni di arrivo i *frame* destinati ad una destinazione saranno inviati solamente a quella LAN. La topologia della rete può cambiare a seconda che macchine e *bridge* vengano accesi, spenti o spostati. Per gestire topologie dinamiche ogni volta che il *bridge* crea una nuova voce nella tabella *hash* annota anche il tempo di arrivo del *frame*, in questo modo ogni volta che riceve un *frame* la cui origine è stata registrata nella tabella il *bridge* aggiorna la voce associata prendendo nota del tempo corrente e di conseguenza il tempo associato ad ogni voce indica quando è stato rilevato l'ultima volta un *frame* proveniente da quella macchina. In questo modo un computer scollegato alla sua LAN torna operativo senza necessità di un intervento manuale, ma proprio a causa di questo algoritmo se una stazione rimane silenziosa per alcuni minuti il traffico che le viene inviato dovrà essere ritrasmesso attraverso tutte le reti fino a quando questa non invierà un suo *frame*. La procedura di instradamento dipende dalle LAN sorgente e destinazione: se la destinazione e la sorgente coincidono il *frame* è scartato, se sono diverse il *frame* è inoltrato mentre se la destinazione è sconosciuta si utilizza la trasmissione su tutte le LAN.

## *Bridge spanning tree*

Per aumentare l'affidabilità alcuni siti installano due o più *bridge* in parallelo tra coppie di LAN, ma questa soluzione introduce nuovi problemi in quanto si vengono a creare degli anelli nella topologia. Per risolvere questo problema è necessario che i *bridge* comunichino tra di loro e coprano la topologia reale con un struttura ***spanning tree*** che raggiunga ogni LAN, ovvero alcune potenziali connessioni tra LAN vengono ignorate nell'interesse della costruzione di una topologia fittizia priva di anelli. Nella struttura ad albero un solo percorso collega ogni LAN ad un'altra e, una volta che i *bridge* hanno concordato lo *spanning tree* tutti gli inoltri tra LAN seguono la struttura. Per costruire lo *spanning tree* i *bridge* devono scegliere quale tra loro dovrà fungere da nodo principale. Questa scelta ricade sul dispositivo che ha il numero seriale più basso, e in base a lui viene costruita una struttura ad albero basata su percorsi più brevi che uniscono il nodo principale a ogni *bridge* e LAN. Se viene meno un componente della struttura si elabora uno nuovo *spanning tree* in quanto l'algoritmo continua a funzionare per rilevare automaticamente modifiche della tolopogia e aggiornare la struttura.

## *Bridge* remoti

Per raggiungere l'obiettivo di avere tutte le LAN interconnesse in modo che il sistema funzioni come una singola grande LAN è necessario installare un *bridge* in ogni LAN e collegare i *bridge* a coppie con una linea punto-punto.
Sulle linee punto-punto si possono usare molti protocolli. L'alternativa dei protocolli di collegamento standard per questo tipo di linee, inserendo poi nel carico utile i *frame* MAC completi, funziona meglio se tutte le LAN sono identiche in quanto l'unico problema sarebbe l'invio del *frame* alla LAN giusta. Per quanto riguarda, invece, l'ipotesi di trappare intestazione e coda MAC nel *bridge* sorgente e mettere ciò che rimane del carico utile del protocollo punto-punto si pone il problema che il *checksum* che arriva all'*host* di destinazione non è quello elaborato dall'*host* sorgente, perciò non è possibile rilevare gli errori causati da eventuali bit guasti nella memoria del *bridge*.

## Ripetitori, *hub*, *bridge*, *switch*, *router* e *gateway*

I **ripetitori** si trovano sul fondo, nello strato fisico, e sono dispositivi analogici collegati a due segmenti di cavo. Un segnale che appare su un segmento è amplificato e trasmesso sull'altro, i ripetitori si occupano quindi solamente di Volt. Poi ci sono gli ***hub***, che hanno diverse linee di input collegate elettricamente a formare un unico dominio di collisione. Tutte le linee di input devono operare alla stessa velocità.
Nello strato *data link* troviamo prima di tutto i ***bridge***, dei dispositivi che collegano una o più LAN. Quando arriva un *frame* il *software* che si trova nel dispositivo estrae l'indirizzo di destinazione dall'intestazione e lo confronta con le voci della sua tabella per scoprire la destinazione dei dati. Un *bridge* può avere schede di linea per reti e velocità diverse e, a differenza degli *hub*, ogni linea è un dominio di collisione indipendente. Gli ***switch*** sono simili ai *bridge* in quanto entrambi instradano i dati in base agli indirizzi dei *frame*, la differeenza principale è che quest'ultimo dispositivo è utilizzato prevalentemente per collegare singoli computer. A causa di questa sua caratteristica d'uso gli *switch* devono poter contenere un numero maggiore di schede di linea rispetto ai *bridge* e ogni scheda è in grado di memorizzare in un *buffer* i *frame* che arrivano attraverso le porte. Poiché ogni porta è un dominio di collisione indipendente gli *switch* non perdono mai i *frame* a causa delle collisioni ma iniziano ad esserci problemi quando lo spazio del buffer è esaurito. Per risolvere questo problema si abbandona la tecnica di commutazione *store-and-forward* per passare alla

*switch cut-through* in cui il dispositivo inizia a inoltrare i *frame* non appena arriva il campo destinazione dell'intestazione. Sempre in questo livello troviamo i ***router***, ovvero dei dispositivi che strappano intestazione e coda del *frame* e passano il pacchetto contenuto nel suo carico utile al *software* d'instradamento che sceglierà la linea di output. Questo *software* non vede gli indirizzi del *frame* e non conosce nemmeno il tipo di rete da cui arriva.
Al livello di trasporto troviamo i ***gateway***, che collegano due computer che usano protocolli di trasporto orientati a connessioni differenti. Un *gateway* di questo tipo può copiare i pacchetti provenienti da una connessione sull'altra modificando il formato secondo necessità, mentre un *gateway* di applicazione comprende il formato e il contenuto dei dati e traduce i messaggi da un formato ad un altro.

CAPITOLO 5

# STRATO *NETWORK*

Lo strato ***network*** si occupa del trasporto dei pacchetti lungo tutto il cammino percorso dall'origine alla destinazione finale. Per raggiungere i suoi obiettivi questo strato deve conoscere la topologia della sottorete di comunicazione e scegliere i percorsi più appropriati attraverso di essa e deve occuparsi dei nuovi problemi che nascono quando sorgente e destinazione si trovano su reti diverse.

## 5.1 Problemi dell'architettura dello strato *network*

### Commutazione di pacchetto *store-and-forward*

Nella **commutazione di pacchetto *store-and-forward*** un *host* con un pacchetto da trasmettere invia dati al *router* più vicino attraverso la sua stessa LAN o attraverso un collegamento punto-punto con l'operatore di telecomunicazioni. Qui il pacchetto viene memorizzato fino a quando non è interamente arrivato per verificare il *checksum*, quindi viene inoltrato al *router* successivo che si trova lungo il percorso fino a quando non raggiunge l'*host* di destinazione.

### Servizi forniti allo strato di trasporto

Lo strato *network* fornisce servizi allo strato di trasporto attraverso l'interfaccia tenendo conto dei seguenti obiettivi: i servizi non dovrebbero essere legati alla tecnologia del *router* quindi allo strato di trasposto dovrebbero essere nascosti dettagli al riguardo, e gli indirizzi di rete disponibili allo strato di trasporto dovrebbero utilizzare uno schema di numerazione uniforme.
Il nocciolo della questione è se lo strato di *network* dovrebbe fornire un servizio orientato alle connessioni oppure un servizio senza connessione. Una fazione afferma che il lavoro dei *router* è spostare i pacchetti da un punto all'altro e niente di più, arrivando alla conclusione che un servizio di rete non dovrebbe essere orientato alle connessioni e in particolare non si dovrebbe eseguire nessuna operazione di ordinamento dei pacchetti e di controllo di flusso in quanto gli *host* si occupano già di quest'aspetto. L'altra fazione afferma che la sottorete dovrebbe fornire un servizio affidabile orientato alle connessioni, e in questa visione la qualità del servizio è il fattore dominante.

### Implementazione del servizio senza connessione

Se il servizio è senza connessione i pacchetti (***datagram***) sono inoltrati alla sottorete (**sottorete di datagrammi**) individualmente e indipendentemente l'uno dall'altro.
Supponendo che un processo abbia un lungo messaggio per un secondo processo che si trova su un *host* differente, la sorgente porge il messaggio allo strato di trasporto il cui codice aggiunge un'intestazione di trasporto all'inizio del messaggio e passa al risultato allo strato di *network*. Supponendo che il messaggio sia lungo e debba quindi essere suddiviso il più pacchetti, questi vengono inviati uno dopo l'altro al primo *router* usando un protocollo punto-punto. A questo punto l'operatore di telecomunicazioni assume il controllo dell'operazione, ogni *router* ha una tabella

interna che indica dove vengono inviati i pacchetti diretti a ogni possibile destinazione e non appena i pacchetti raggiungono il primo *router* sono archiviati temporaneamente per verificarne il *checksum* prima che questi possano venire trasmessi secondo le scelte effettuate di volta il volta dall'**algoritmo di *routing*** agli altri *router* in maniera che il messaggio arrivi a destinazione.

### Implementazione del servizio orientato alla connessione

Se il servizio è orientato alle connessioni prima di inviare i pacchetti si deve stabilire un percorso che colleghi il *router* sorgente al *router* destinazione. Questa connessione è chiamata **CV** e la sottorete è detta **sottorete a circuito virtuale**.
L'idea che sta alla base di questo tipo di servizi è quella di evitare di dover scegliere una nuova strada per ogni pacchetto inviato in quanto il percorso viene scelto durante l'impostazione della connessione e archiviato nelle tabelle dei *router*. Il percorso in questione è utilizzato per tutto il traffico che scorre attraverso la connessione e nel momento in cui viene rilasciata anche il circuito viene terminato.
Con un servizio orientato alle connessioni ogni pacchetto contiene un identificatore che indica il circuito virtuale di appartenenza.

### Confronto tra sottoreti a circuito virtuale e a datagramma

I circuiti virtuali permettono ai pacchetti di utilizzare i numeri di circuito al posto degli indirizzi di destinazione completi. Se i pacchetti tendono ad essere corti un indirizzo di destinazione completo in ogni pacchetto può aumentare in modo significativo l'*overhead* e di conseguenza sprecare banda, ma il prezzo da pagare per l'utilizzo di circuiti virtuali è rappresentato dallo spazio delle tabelle dei *router*. L'utilizzo di circuiti virtuali richiede una fase di configurazione che esige tempo e consuma risorse ma dopo è facile capire che cosa si deve fare con un pacchetto dati in una sottorete di questo tipo, in con i datagrammi invece per individuare la voce relativa alla destinazione serve una procedura di ricerca complicata. Per quanto riguarda, invece, la quantità di spazio di tabella richiesto nella memoria del *router* una sottorete a datagrammi ha bisogno di una voce per ogni possibile destinazione mentre una sottorete a circuito virtuale si accontenta di una voce per ogni circuito.
I circuiti virtuali sono in vantaggio quando si tratta di dare garanzie sulla qualità del servizio ed evitare le congestioni all'interno della sottorete, dato che le risorse possono essere riservate al momento di stabilire la connessione. Per i sistemi che elaborano transizioni, invece, l'*overhead* richiesto per impostare e rilasciare un circuito virtuale può facilmente superare quello di utilizzo vero e proprio del circuito. Oltre a questo i circuiti virtuali hanno un problema di vulnerabilità in quanto se un *router* va in *crash* e perde i dati nella sua memoria tutti i circuiti virtuali che passano attraverso il dispositivo devono essere terminati, al contrario se un *router* per datagrammi si blocca ne soffrono solo gli utenti che in quel momento avevano pacchetti accodati nel *router*.

## 5.2 Algoritmi di *routing*

L'**algoritmo di *routing*** è quella parte del *software* dello strato *network* che si preoccupa di scegliere lungo la quale linea di uscita vanno instradati i pacchetti in arrivo. Se la sottorete utilizza

internamente i datagrammi questa decisione va ripetuta per ogni pacchetto di dati in arrivo, se la sottorete utilizza i circuiti virtuali invece le decisioni riguardano il *routing* vengono prese solo quando viene impostato un nuovo circuito virtuale.
Alcune proprietà dell'algoritmo di *routing* sono desiderabili indipendentemente dal fatto che i percorsi siano scelti indipendentemente per ogni pacchetto o stabiliti durante l'impostazione di una nuova connessione. Precisione e semplicità non richiedono commenti. Per quanto riguarda la robustezza, invece, possiamo dire che l'algoritmo dovrebbe essere in grado di far fronte ai cambiamenti di topologia e di traffico senza che sia necessario interrompere tutto il lavoro in tutti gli *host* e riavviare la rete ogni volta che un *router* si blocca. Anche la stabilità è un obiettivo importante per l'algoritmo di *routing*. Per quanto riguarda imparzialità e ottimizzazione possono sembrare qualità ovvie, ma spesso sono contradditorie.
Gli algoritmi di *routing* si possono raggruppare in due classi principali. Gli **algoritmi non adattativi** (o statici) non basano le loro decisioni su misure o stime del traffico e della topologia corrente mentre al contrario gli **algoritmi adattativi** cambiano le loro decisioni secondo le modifiche apportate alla topologia e al traffico. Gli algoritmi di questo tipo si distinguono per la fonte da cui traggono le loro informazioni, per il momento in cui modificano i percorsi e per il tipo di metrica utilizzata nell'operazione di ottimizzazione.

## Il principio di ottimalità

Risulta possibile fare un'ipotesi generale riguardo ai percorsi ottimali indipendentemente dalla topologia di rete o dal traffico. Questo assioma, conosciuto come **principio di ottimalità**, afferma che se il *router J* si trova sul percorso ottimale che collega *I* a *K* allora il percorso ottimale da *J* a *K* segue la stessa rotta.
Come diretta conseguenza si vede che la serie di percorsi ottimali che collegano tutte le sorgenti ad una data destinazione formano una struttura ad albero dove il nodo principale è la destinazione. In questo albero, chiamato ***sink tree***, la distanza è calcolata in base al numero di salti. Il *sink tree* non è necessariamente unico e l'obiettivo di tutti gli algoritmi di *routing* è scoprire e utilizzare gli alberi per tutti i *router.*

## *Routing* basato sul percorso più breve

L'idea di base è quella di costruire un grafo della sottorete dove ogni nodo rappresenta un *router* e ogni arco una linea di comunicazione. Per scegliere un percorso tra una coppia di *router* l'algoritmo cerca semplicemente la strada più corta che collega i due nodi del grafo.Si possono usare molte metriche per definire la lunghezza del percorso, tra cui la distanza fisica e il numero di salti. Nel caso generale le etichette sugli archi possono essere calcolate come una funzione della distanza, della banda, del traffico medio, del costo della comunicazione, della lunghezza media della coda, del ritardo e di altri fattori. Modificando la funzione che attribuisce i pesi l'algoritmo calcolerebbe il percorso più breve misurato secono qualunque numero di criteri o loro combinazione.
Nell'algoritmo chiamato ***Shortest Path First*** ogni nodo è associato ad un'etichetta che riporta la sua distanza dal nodo d'origine lungo il miglior percorso conosciuto. Inizialmente nessun percorso è noto e quindi tutti i nodi sono etichettati col simbolo dell'infinito. Mano a mano che l'algoritmo procede tutti i nodi sono etichettati e si trovano i percorsi, le etichette cambiano rispecchiando i percorsi migliori. Inizialmente tutte le etichette sono provvisorie, e quando si scopre che rappresenta il percorso più breve possibile dall'origine a quel nodo l'etichetta in questione viene resa permanente smettendo di modificarla.

## *Flooding*

L'algoritmo di ***flooding*** è un algoritmo statico in cui ogni pacchetto viene inviato a tutte le linee tranne a quella di provenienza. Questo meccanismo genera un numero infinito di pacchetti duplicati a meno che non si prendano delle misure per attenuare il processo. Una di queste potrebbe essere inserire all'interno dell'intestazione un contatore di salti (il cui valore iniziale corrisponde alla lunghezza del percorso dall'origine alla destinazione o il diametro dell'intera sottorete) che viene decrementato fino a che non raggiunge lo zero. Una tecnica di moderazione alternativa tiene traccia dei pacchetti trasmessi nel *flood* in modo da evitare una seconda trasmissione, ovvero il *router* di origine deve inserire in ogni pacchetto che riceve dai suoi *host* un numero di sequenza.
Una variante chiamata ***flooding* selettivo** invia ogni pacchetto solo attraverso le linee che vanno approssimativamente nella direzione giusta.

## *Routing* basato sul vettore delle distanze

Gli algoritmi di *routing* basati sul **vettore delle distanze** (*distance vector routing*) operano facendo in modo che ogni *router* conservi una tabella di *routing* indicizzata da ogni *router* della sottorete che definisce la migliore distanza conosciuta per ogni destinazione e la linea che conduce a tale destinazione. Queste tabelle sono aggiornate scambiando informazioni con i *router* vicini.
Questo algoritmo opera presumendo che il *router* conosca la distanza che lo separa da ognuno dei suoi vicini.

**Il problema del conto all'infinito.** In teoria il *routing* basato sul vettore delle distanze funziona, ma ha un serio difetto pratico: sebbene converga verso la risposta corretta può raggiungere l'obiettivo molto lentamente, questo perché reagisce rapidamente alle buone notizie ma troppo lentamente a quelle cattive in quanto nessun *router* ha mai un valore che supera di oltre una unità quello minimo di tutti i vicini. Lentamente i *router* trovano la loro strada verso l'infinito ma il numero di scambi richiesti dipende dal valore numerico utilizzato per rappresentare l'infinito stesso. Per questo motivo è meglio impostare questo valore in corrispondenza del percorso più lungo più uno. Sono stati fatti diversi tentativi di risolvere questo problema ma in generale nessuno di questi espedienti funziona bene in quanto il problema fondamentale è che quando *X* dice a *Y* che ha un percorso che punta da qualche parte, *Y* non ha modo di sapere se lui stesso fa parte del percorso.

## *Routing* basato sullo stato dei collegamenti

Il *routing* basato sul vettore delle distanze utilizzato in ARPANET venne presto sostituito per due motivi principali: poiché si basava sulla lunghezza della coda la metrica del ritardo non teneva conto della banda della linea nella scelta dei percorsi ma cambiando la metrica del ritardo l'algoritmo spesso impiegava troppo tempo a raggiungere la convergenza.
L'idea che stanno alla base del *routing* basato sullo **stato dei collegamenti** (*link state routing*) è quella di misurare sperimentalmente la topologia completa e tutti i ritardi per poi distribuirli ad ogni *router.*

**Scoperta dei vicini.** Quando viene acceso il *router* cerca di scoprire chi sono i suoi vicini inviando uno speciale pacchetto HELLO su ogni linea punto-punto e riceve come risposta l'identità del *router* all'altro capo della linea. Quando due o più *router* sono collegati ad una LAN, la LAN stessa viene considerata come un solo nodo.

**Misurazione del costo della linea.** Il modo più diretto per determinare questo ritardo consiste nell'inviare un pacchetto ECHO al quale l'altra parte deve rispondere immediatamente, misurando

il tempo di andata e ritorno e dividendo il risultato per due il *router* trasmittente può ottenere una stima ragionevole del ritardo. Per considerare anche il carico il cronometro che misura il tempo di andata e ritorno deve essere avviato nel momento in cui ECHO viene inserito nella coda, per ignorarlo invece dovrebbe essere avviato quando raggiunge il fronte della coda.
Entrambi i metodi sono discutibili. Includere i ritardi indotti dal traffico significa che quando dovrà scegliere tra due linee di pari banda una delle quali è caricata pesantemente il *router* considera il più breve il percorso che attraversa la linea non appesantita migliorando le prestazioni. Il rovescio della medaglia è rappresentato da tabelle di *routing* altamente instabili che causano instradamenti irregolari e molti potenziali problemi. Ignorando il carico e considerando solamente la banda questo problema non si pone. In alternativa il carico può essere distribuito su più linee, ma con la consapevolezza di non sfruttare al meglio il percorso migliore.

**Costruzione dei pacchetti che contengono lo stato dei collegamenti.** Il pacchetto in questione inizia con l'identità del trasmittente, seguita da un numero di sequenza, dall'età e da una lista di vicini per ognuno dei quali è riportato il ritardo misurato. Risulta semplice costruire questi pacchetti, la parte difficile è determinare se costruirli periodicamente o quando accade un evento significativo.

**Distribuzione dei pacchetti che contengono lo stato dei collegamenti.** L'idea fondamentale è quella di utilizzare il *flooding* per distribuire i pacchetti contenenti le informazioni sullo stato dei collegamenti e tenere sotto controllo il flusso dati inserendo un numero di sequenza, incrementato per ogni pacchetto, nell'intestazione. I *router* tengono traccia di tutte le coppie (*router* sorgente, sequenza) rilevate in modo che quando arriva un nuovo pacchetto confronta i dati con quelli già visti e se è nuovo il pacchetto è inoltrato.
Questo algoritmo ha alcuni difetti. Prima di tutto i numeri di sequenza ripetitivi potrebbero generare il caos, per questo si utilizzano numeri di sequenza lunghi 32 bit. In secondo luogo quando un *router* si blocca perde traccia dei suoi numeri di sequenza e se il conteggio ricomincia da 0 il paccchetto successivo viene rifiutato perché ritenuto un duplicato. Inoltre se un numero di sequenza arriva danneggiato i pacchetti successivi fino al numero di sequenza fallato saranno considerati obsoleti e quindi ignorati. Per risolvere questi due problemi è necessario inserire un campo età subito dopo il numero di sequenza e decrementare il suo valore una volta al secondo fino a che non si raggiunge il valore 0. Questo campo viene decrementato anche durante il processo di *flooding* iniziale per garantire che nessun pacchetto possa andare perduto e duri per un tempo indefinito.
Altre migliorie all'algoritmo sono il fatto che quando un pacchetto contiene i dati sullo stato dei collegamenti raggiunge un *router* per il *flooding* questo non viene accodato immediatamente per la trasmissione ma i dati vengono prima inseriti in un'area di mantenimento. Se prima della trasmissione arriva dalla stessa sorgente un altro pacchetto contenente dati sullo stato dei collegamenti il *router* confronta i numeri di sequenza: se i valori sono uguali il duplicato viene scartato, altrimenti viene eliminato il più vecchio. Inoltre, per prevenire errori sulle linee da *router* a *router* tutti i pacchetti contenenti informazioni relative allo stato dei collegamenti ricevono un *acknowledgement*.

## *Routing* gerarchico

Quando la dimensione delle tabelle di *routing*, che cresce proporzionalmente alla dimensione della rete, consuma non soltanto la memoria del computer ma aumenta il tempo che la CPU impiega ad analizzare i dati e la banda necessaria per inviare informazioni sullo stato è necessario cambiare algoritmo di *routing*.
Nel ***routing* gerarchico** i *router* sono divisi in regioni: ogni *router* conosce i dettagli relativi al *routing* dei pacchetti diretti a destinazioni nella stessa regione ma non sa nulla della struttura interna della regione stessa. Per reti di dimensioni maggiori può non essere sufficiente una struttura

gerarchica a due livelli quindi le regioni sono raggruppate in *cluster*, i *cluster* in zone, le zone in gruppi e così via. Quando si connettono reti diverse è naturale considerare ogni rete come una regione separata. Sfortunatamente questi risparmi di spazio hanno come prezzo da pagare la crescita della lunghezza dei percorsi.
Per quanto riguarda il numero delle livelli all'interno di una rete, il loro numero ottimale per una sottorete di *N router* è uguale a ln*N*, per un totale di *e* voci per *router*. Insieme a questo si è potuto dimostrare che l'aumento della lunghezza del percorso medio reale causato da un *routing* di questo tipo è sufficientemente piccolo da renderlo il più delle volte accettabile.

### *Routing broadcast*

Un metodo di trasmissione *broadcast* che non richiede l'implementazione di alcuna funzionalità speciale nella sottorete è quello in cui la sorgente si invia un pacchetto distinto ad ogni destinazione. Questo metodo non solo spreca banda ma obbliga anche la sorgente a possedere una lista completa di tutte le destinazioni.
Un altro candidato è il meccanismo di ***flooding***. Il problema di questa tecnica è che genera troppi pacchetti e consuma troppa banda.
Un terzo algoritmo è quello del ***multidestination routing***, in cui gni pacchetto contiene una lista delle destinazioni o una mappa di bit che indica le destinazioni desiderate. Quando riceve un pacchetto il *router* controlla tutte le destinazioni per determinare l'insieme di linee di trasmissione richieste e genera una nuova copia del pacchetto per ogni linea di output usata, includendo in ogni pacchetto solo quelle destinazioni che si trovano sulla linea. Dopo un numero sufficiente di salti ogni pacchetto conterrà una sola destinazione e potrà essere trattato come percorso normale.
Un'altra tecnica utilizza uno ***spanning tree***, ovvero un sottoinsieme che comprende tutti i *router* ma che non contiene cicli. Se ogni *router* sa quali delle sue linee appartengono allo *spanning tree* allora può copiare un pacchetto *broadcast* in arrivo su tutte le linee dell'albero, esclusa quella d'ingresso. Questo metodo usa in modo eccellente la banda e genera in assoluto il minimo numero di pacchetti necessari per svolgenere il lavoro, ma per rendere applicabile il metodo ogni *router* deve conoscere uno *spanning tree*.
L'ulimo algoritmo, chiamato ***reverse path forwarding***, tenta di approssimare il comportamente dell'algoritmo precedente anche quando i *router* non sanno nulla degli *spanning tree*. Quando riceve un pacchetto *broadcast* il *router* verifica se il pacchetto è giunto attraverso la linea che normalmente è utilizzata per inviare pacchetti alla sorgente della trasmissione *broadcast*. In caso affermativo c'è una forte probabilità che il pacchetto stesso abbia seguito il percorso migliore del *router* e, in questo caso, il *router* inoltra copie del pacchetto attraverso tutte le linee esclusa quella di input. In caso contrario, invece, il pacchetto è scartato in quanto è probabile che si tratti di un duplicato. Il vantaggio principale di questo algoritmo è che si tratta di un sistema efficiente e facile da implementare.

## 5.3 ALGORITMI PER IL CONTROLLO DELLA CONGESTIONE

Quando troppi pacchetti sono presenti in una porzione della sottorete le prestazioni degradano. Questa situazione è chiamata **congestione**. Se il traffico aumenta in modo eccessivo i *router* non riescono a far fronte alla situazione e cominciano a perdere pacchetti, con un traffico elevatissimo le prestazioni crollano completamente e quasi nessun pacchetto viene inoltrato.
La congestione può essere causata da più fattori. Quando inizia a formarsi coda, per esempio, se la

memoria del *router* non è sufficiente non sarà possibile conservare tutti i dati, perciò alcuni pacchetti andranno persi. Nel caso in cui i *router* avessero una memoria infinita la congestione peggiora, poiché quando raggiungeranno il fronte della coda i pacchetti saranno già scaduti e quindi saranno già stati trasmessi dei duplicati. Infine anche i processori lenti possono causare congestioni. Il più delle volte il vero problema è un cattivo adattamento tra le parti del sistema, che perdura fino a quando non si raggiunge un equilibrio tra tutti i componenti.
Controllare la congestione significa garantire che la sottorete sia in grado di trasportare il traffico immesso. Questo problema è globale, coinvolge il comportamento di tutti i componenti della rete. Il controllo del flusso, invece, ha a che fare con il traffico punto-punto tra trasmettitore e ricevitore: il suo compito è quello di evitare che una sorgente veloce trasmetta continuamente una quantità di dati maggiore di quella che il ricevitore è in grado di assorbire, e coinvolge spesso una retroazione. Il motivo per cui spesso il controllo della congestione viene confuso con il controllo del flusso è che alcuni algoritmi di controllo della congestione inviano alle sorgenti messaggi che dicono di rallentare la trasmissione quando la rete ha problemi.

## Principi generali del controllo della congestione

Quando i problemi delle reti di computer vengono considerati dal punto di vista della teoria del controllo si è portati a dividere tutte le soluzioni in due categorie: cicli aperti e cicli chiusi. Le **soluzioni a ciclo aperto** tentano di risolvere il problema mediante un buon progetto che renda in primo luogo improbabile la manifestazione del problema stesso e, una volta che il sistema è stato attivato non viene eseguita alcuna correzione in corsa. Tutti questi approcci hanno in comune il fatto che le decisioni sono prese senza tener conto dello stato corrente della rete. Al contrario le **soluzioni a ciclo chiuso** si basano sul concetto di retroazione. Questo approccio è composto da tre parti: il controllo del sistema per rilevare dove e quando si presenta la congestione, il passaggio di queste informazioni ai punti dove si possono eseguire le azioni di correzioni e la regolazione del funzionamento del sistema.
Si conoscono molti algoritmi di controllo della congestione, che vengono appunto divisi in soluzioni a ciclo aperto e chiuso. Gli algoritmi a ciclo aperto si dividono a loro volta in soluzioni che agiscono sulla sorgente e soluzioni che lavorano sulla destinazione. Gli algoritmi a ciclo chiuso invece vengono distinti in algoritmi a retroazione esplicita, ovvero in cui per avvisare la sorgete vengono spediti indietro i pacchetti dal punto della congestione, e algoritmi a retroazione implicita, in cui la sorgente deduce l'esistenza della congestione effettuando osservazioni locali.
La presenza di congestione indica che il carico è momentaneamente più grande di quello che può essere gestito. Le due soluzioni più immediate sono aumentare le risorse o diminuire il carico. Quando non è possibile aumentare la capacità il carico può essere ridotto negando servizi ad alcuni utenti, degradando il servizio ad alcuni o a tutti gli utenti e facendo in modo che gli utenti programmino le richieste in modo prevedibile. Per le sottoreti che utilizzano internamente circuiti virtuali questi metodi possono essere implementati sullo strato di *network*, per le sottoreti a datagramma a volte si possono egualmente implementare sulle connessioni dello strato di trasporto.

## Criteri per prevenire la congestione

I sistemi a ciclo aperto, che sono stati progettati in primo luogo per ridurre la possibilità di una congestione e non per risolvere il problema quando si presenta, tentano di raggiungere il loro obiettivo usando opportuni criteri su più livelli.

| Strato | Criteri |
|---|---|
| Trasporto | • Ritrasmissione<br>• *Catching* fuori sequenza<br>• *Acknowledgement*<br>• Controllo di flusso<br>• Determinazione dei *timeout* (se l'intervallo di tempo è troppo breve verranno trasmessi inutilmente pacchetti aggiuntivi, se è troppo lungo la congestione verrà ridotta ma il tempo di risposta ne soffrirà ogni volta che un pacchetto si perde) |
| *Network* | • Scelta fra circuiti virtuali e datagrammi nella sottorete (molti algoritmi di controllo della congestione funzionano solo con le sottoreti a circuito virtuale)<br>• Accodamento e servizio dei pacchetti (riguarda il numero di code presenti sui *router* - una per ogni linea di input, una per ogni linea di output o entrambe - e l'ordine in cui i pacchetti sono elaborati)<br>• Eliminazione dei pacchetti (quale pacchetto scartare se non c'è più spazio)<br>• Algoritmo di *routing* (può aiutare a evitare la congestione sparpagliando il traffico attravero tutte le linee)<br>• Gestione della vita utile del pacchetto (se la vita è troppo lunga i pacchetti persi possono intasare la rete per molto tempo, se è troppo breve possono scadere prima di raggiungere le loro destinazioni causando ritrasmissioni) |
| *Data link* | • Ritrasmissione (riguarda la velocità con il quale un trasmettitore gestisce la scadenza dei pacchetti e cosa trasmette dopo il *timeout*)<br>• *Catching* fuori sequenza (se i ricevitori scartano sistematicamente tutti i pacchetti fuori sequenza questi verranno successivamente ritrasmessi creando carico)<br>• *Acknowledgement* (see ogni pacchetto riceve immediatamente *scknowledgement* questi pacchetti generano traffico aggiuntivo, se al contrario vengono trasportati dal traffico inverso si possono avere ritrasmissioni e scadenze aggiuntive)<br>• Controllo di flusso |

## Controllo della congestione nelle sottoreti a circuito virtuale

Una tecnica largamente utilizzata è il controllo di ammissione, ovvero una volta che la congestione è stata segnalata nessun circuito virtuale viene più impostato fino a quando il problema non viene risolto. Questo approccio è decisamente rozzo, ma facile da implementare. Una seconda strategia consiste nella creazione di nuovi circuiti virtuali intradandoli attentamente in modo da aggirare le aree congestionate. Come alternativa è possibile adottare una tecnica che consiste nel negoziare un accordo tra l'*host* e la sottorete durante l'impostazione del circuito virtuale, specificando volume e forma del traffico, qualità del servizio richiesta e altri parametri. In questo modo è poco probabile che la congestione si presenti sui nuovi circuiti virtuali in quanto tutte le risorse necessarie sono garantite e disponibili. L'utilizzo costante di questa tecnica ha, però, uno svantaggio: tende a sprecare risorse e di conseguenza la banda risulta sprecata.

## Controllo della congestione nelle sottoreti a datagrammi

***Chocke packet.*** In questo approccio il *router* invia all'*host* sorgente un *chocke packet* dandogli la destinazione trovata nel pacchetto e il pacchetto originale viene etichettato in modo da impedire la generazione di altri *chocke packet* lungo il percorso prima di essere inoltrato. Quando riceve il *chocke packet* la sorgente devi ridurre dell'*X* per cento il traffico inviato alla destinazione specificata e dato che altri pacchetti puntano alla stessa destinazione per un intervallo di tempo prefissato l'*host* dovrebbe ignorare i *choche packet* che si riferiscono a quella destinazione. Trascorso questo intervallo di tempo l'*host* rimane in ascolto di uteriori *chocke packet*: se ne arrivano la linea è ancora congestionata e l'*host* riduce il flusso ancora un po', invece se non ne arrivano può aumentare nuovamente il flusso.
In alcune varianti di questo algoritmo i *router* possono mantenere diverse soglie di guardia e in base alla soglia che viene oltrepassata il *chocke packet* può contenere un avviso, un allarme o un ultimatum. In un'altra variante invece come segnale di attivazione si utilizza la lunghezza della coda o l'utilizzo del buffer al posto dell'utilizzatore della linea.

***Chocke packet hop-by-hop.*** Ad alte velocità o lunghe distanze l'invio dei *chocke packet* agli *host* sorgente non funziona bene perché la reazione è lenta. In un approccio alternativo il *chocke packet* ha effetto su tutti i salti attraversati. Questo schema ha l'effetto di alleviare rapidamente la congestione nel punto dove si forma al prezzo di un incremento dell'utilizzo dei buffer di trasmissione nella parte del percorso precedente, in questo modo la congestione può essere stroncata sul nascere senza perdere alcun pacchetto.

## *Load shedding*

La tecnica ***load shedding*** è un modo per indicare l'azione di eliminazione intrapresa dai *router* inondati da troppi pacchetti. Un *router* sommerso da pacchetti può scartare qualche pacchetto a caso, ma la scelta del pacchetto giusto può dipendere dalle applicazioni in esecuzione: per il trasferimento di file un pacchetto vecchio è più preziooso di un pacchetto nuovo (*wine*), mentre nel caso di dati multimediali un pacchetto nuovo è più importante di uno vecchio (*milk*). Un ulteriore passo prevede la cooperazione dei trasmettitori.
Per implementare un criterio di eliminazione intelligente le applicazioni devono contrassegnare i loro pacchetti in classi di priorità in modo da indicare la loro importanza. Se lo fanno i *router* possono scartare i pacchetti partendo da quelli di classe più bassa. La strategia più efficiente al riguardo consente agli *host* di superare il limiti specificati nell'accordo negoziato durante l'impostazione del circuito virtuale a condizione che essi accettino di trasmettere tutto il traffico in eccesso come traffico a bassa priorità. Questo rende più efficiente l'utilizzo delle risorse poiché permette gli *host* di adoperarle quando nessun altro è interessato, senza stabilire un diritto di utilizzo quando la situazione si fa difficile.

**RED (*Random Early Detection*).** Risulta più semplice gestire la congestione appena viene rilevata piuttosto che cercare di porvi rimedio. Questa osservazione conduce all'idea di scartare i pacchetti prima che tutto lo spazio del buffer sia completamente esaurito. Per stabilire quando è il momento giusto per iniziare a scartare i pacchetti i *router* mantengono una media mobile delle lunghezze delle code e quando la lunghezza della coda su una linea supera una soglia di guardia la linea è considerata congestionata e viene intrapresa l'azione di eliminazione. Scegliere un pacchetto a caso dalla coda in questione è il massimo che si possa fare. Non occorre nemmeno rendere nota l'operazione, in quanto la sorgente alla fine noterà l'assenza del pacchetto di *acknowledgement* e prenderà un'iniziativa. Questa forma implicita di *feedback* funziona solo quando le sorgenti rispondono ai pacchetti perduti rallentando la loro velocità di trasmissione.

## Controllo del *jitter*

Un ***jitter*** è la variazione nel tempo di arrivo di un pacchetto. Il *jitter* può essere delimitato calcolando il tempo di transito atteso per ogni salto lungo il percorso. Quando il pacchetto raggiunge il *router* questo controlla di quanto il pacchetto è in anticipo o in ritardo rispetto alla sua programmazione e questa informazione viene memorizzata nel pacchetto e aggiornata ad ogni salto. Se è in anticipo il pacchetto è trattenuto quanto basta per rientrare nella pianificazione, mentre se è in ritardo il *router* tenta di trasmetterlo il prima possibile.
In alcune applicazioni, come quelle basate sui video *on demand*, il *jitter* può essere eliminato memorizzando i dati in un buffer del computer ricevente mentre in altre, come quelle che richiedono interazione in tempo reale, il ritardo inerente alla memorizzazione non è accettabile.

# 5.4 QUALITÀ DEL SERVIZIO

## Requisiti

Le esigenze di ogni flusso possono essere caratterizzate da quattro parametri primari: affidabilità, ritardo, *jitter* e banda. L'insieme di questi parametri determina il **QoS** (*Quality of Service*).

| Applicazione | Affidabilità | Ritardo | *Jitter* | Banda |
|---|---|---|---|---|
| Posta elettronica | H | L | L | L |
| Trasferimento file | H | L | L | A |
| Accesso al web | H | A | L | A |
| Login remoto | H | A | A | L |
| Audio *on demand* | L | L | H | A |
| Video *on demand* | L | L | H | H |
| Telefonia | L | H | H | L |
| Videoconferenza | L | H | H | H |

Le reti ATM classificano i flussi in quattro categorie a seconda delle Qos richieste: velocità costante (tenta di simulare un cavo fornendo una banda e un ritardo uniformi), velocità variabile in tempo reale o non reale (usata in presenza di video compressi) e velocità disponibile (per le applicazioni che non sono sensibili né al ritardo né allo *jitter*).

## Tecniche per ottenere una buona qualità di servizio

**Sovradimensionamento.** Consiste nel fornire così tanta capacità del *router*, spazio di buffer e banda da far transitare facilmente i pacchetti. Il problema di questa soluzione è che è costosa.

**Utilizzo del buffer.** I flussi possono essere memorizzati dal dispositivo ricevente in un buffer prima di essere consegnati, questo non influenza né l'affidabilità né la banda ma aumenta il ritardo ed elimina lo *jitter*. Nel caso dell'audio e del video *on demand* lo *jitter* è il problema principale quindi questa tecnica risulta particolarmente utile.

***Traffic shaping*.** Un output non uniforme è frequente quando il server gestisce molti flussi nello stesso momento e supporta altre azioni. Rendendo uniforme la trasmissione del server la qualità del servizio risulterebbe migliore. Per fare questo quando una connessione viene impostata l'utente e la sottorete si mettono d'accordo su un particolare modello di traffico (*service level agreement*) e fintanto che il cliente si attiene alla sua parte dell'accordo e invia solo pacchetti secondo il contratto concordato l'operatore di telecomunicazioni promette di consegnare i dati in modo tempestivo. Il *traffic shaping* riduce la congestione regolando la velocità media di trasmissione e aiuta così l'operatore a tener fede alle sue promesse. La supervisione del flusso di traffico è chiamata *traffic policing*.

**L'algoritmo *leaky bucket*.** Concettualmente l'*host* collegato alla rete da un'interfaccia che contiene un *leaky bucket*, ossia una coda interna infinita. L'*host* è autorizzato ad inserire nella rete un solo pacchetto per ogni ciclo di *clock*, e il meccanismo trasforma un flusso irregolare di pacchetti proveniente da processi utente dell'*host* in un flusso regolare di pacchetti immesso nella rete. Quando si usano pacchetti a dimensione variabile è meglio consentire un numero fisso di byte per ogni ciclo di *clock* invece che limitarsi ad un unico pacchetto.

**L'algoritmo *token bucket*.** Il *leaky bucket* contiene *token*, generati da un *clock*, che devono essere catturati e distrutti da un pacchetto perché questo possa essere inviato. Con questa tecnica è permesso agli *host* inattivi di risparmiare permessi, fino a riempire la dimensione massima del secchio, per inviare successivamente grandi raffiche di dati. Un'altra differenza con l'algoritmo precedente è che il *token bucket* getta via i *token* quando il secchio si riempie, ma non scarta mai i pacchetti.

## 5.5 COLLEGAMENTO TRA RETI

L'obiettivo dell'interconnessione tra reti è quello di consentire agli utenti di ogni rete di comunicare con gli utenti delle altre oltre che di permettere agli utenti di ciascuna di accedere ai dati delle altre. Per raggiungere questo obiettivo è necessario l'invio di pacchetti da una rete all'altra, ma poiché queste hanno spesso caratteristiche diverse riuscire a trasportare i pacchetti non è sempre immediato.

### Differenze tra le reti

Quando i pacchetti inviati da una sorgente su una rete devono attraversare una o più reti straniere prima di raggiungere la rete di destinazione possono presentarsi molti problemi sulle interfacce che collegano le diverse reti. Prima di tutto se i pacchetti provenienti da una rete orientata alle connessioni devono attraversare una rete senza connessione può essere necessario riordinare i dati. Spesso sono richieste conversioni di protocollo, che possono risultare difficili se ula funzionalità richiesta non può essere espressa. Altra cosa necessaria è convertire gli indirizzi. Il passaggio di pacchetti *multicast* all'interno di reti che non supportano la trasmissione *multicast* richiede la creazione di pacchetti separati per ogni destinazione. La dimensione massima dei pacchetti, inoltre, cambia da un tipo di rete all'altro. Anche le diverse qualità del servizio rappresentano un problema. Il controllo degli errori, del flusso e delle congestionin spesso cambiano in base alla rete. Altri potenziali problemi potrebbero nascere dai diversi meccanismi di protezione, dalle impostazioni dei parametri, dalle regole per il conteggio dei costi e persino dalle leggi sulla privacy.

## Connessione tra le reti

Le reti possono essere collegate tra loro attraverso dispositivi di diversto tipo. Nello strato fisico possono essere connesse mediante ripetitori o *hub*, cioè apparecchi che spostano i bit da una rete ad un'altra rete identica senza modifiche. Nello strato *data link* operano *bridge* e *switch*. Questi apparecchi possono accettare *frame*, esaminare gli indirizzi MAC e inoltrare *frame* su reti diverse eseguendo nel contempo una limitata conversione di protocollo. Nello strato *network* due reti possono essere collegate tramite *router*, e se il loro strati rete sono differenti il *router* può essere capace di tradurre i formati dei pacchetti. Nello strato di trasporto si trovano i *gateway*, che possono interfacciare due connessioni di trasporto. Infine nello strato di applicazione i *gateway* applicativi traducono la semantica dei messaggi
Una differenza sostanziale tra il caso basato sullo *switch* e il caso basato sul *router* è che con uno *switch* (o *bridge*) si trasporta l'intero *frame* sulla base del suo indirizzo MAC, mentre con il *router* il pacchetto è estratto dal *frame* e l'indirizzo nel pacchetto è utilizzato per decidere dove inviarlo. Gli *switch* non devono comprendere il protocollo dello strato di *network* utilizzato per commutare i pacchetti, i *router* invece sì.

## Circuiti virtuali concatenati

In questo modello viene impostata una connessione all'*host* di una rete lontana seguendo le consuete modalità. La sottorete si accorge che la destinazione è remota e costruisce un circuito virtuale diretto al *router* più vicino alla rete di destinazione e quindi crea un circuito virtuale da quel *router* a un *gateway* esterno (*router* multiprotocollo). Questo *gateway* registra l'esistenza del circuito virtuale nelle sue tabelle e procede costruendo un altro circuito virtuale diretto a un *router* nella sottorete successiva. Questo processo continua fino a quando non viene raggiunto l'*host* di destinazione. Una volta che i pacchetti iniziano a scorrere lungo il percorso ogni *gateway* inoltra i pacchetti in arrivo, convertendo i formati dei pacchetti e i numeri di circuito virtuale in base alle necessità.
Ricapitolando, il modello a circuiti virtuali concatenati offre gli stessi vantaggi offerti dai circuiti virtuali usati in una singola sottorete: permette di riservare in anticipo i buffer, garantisce l'ordine dei pacchetti, permette di utilizzare intestazioni corte ed evita i problemi causati dagli stessi pacchetti duplicati trasmessi in ritardo. Questa soluzione ha anche gli stessi svantaggi: ogni connessione aperta occupa spazio nelle tabelle dei *router*, non è possibile utilizzare percorsi alternativi per evitare le zone congestionate e il sistema è vulnerabile ai guasti dei *router* che si trovano sul percorso. Risulta inoltre difficile, se non impossibile, implementare questo metodo se una delle reti coinvolte è a datagrammi inaffidabile.

## Collegamento tra reti senza connessione

In questo modello l'unico servizio che lo strato *network* offre allo strato trasporto è la capacità di introdurre datagrammi nella sottorete. Questo modello non richiede che tutti i pacchetti che appartengano a una connessione debbano attraversare la stessa sequenza di *gateway* in quanto la decisione di *routing* è presa separatamente per ogni pacchetto. Questa strategia può utilizzare diversi percorsi e quindi ottenere una banda maggiore rispetto al modello basato sui circuiti virtuali concatenati, ma d'altra parte non c'è nessuna garanzia che i pacchetti raggiungano la destinazione in ordine.
I problemi maggiori di questo modello sono che se ogni rete utilizza uno strato *network* diverso un pacchetto proveniente da una rete non può passare in un'altra rete. Oltre a questo abbiamo il problema dell'indirizzamento, e per di più cambia il concetto di ciò che è indirizzabile. Nella

migliore delle ipotesi qualcuno dovrebbe gestire un database che tenga traccia di ogni possibile associazione, ma questa soluzione rappresenterebbe una fonte costante di problemi. Una seconda idea è quella di progettare un pacchetto "internet" universale, facendo in modo che tutti i *router* lo riconoscano (approccio di IP). Le proprietà di questo approccio sono più o meno le stesse delle sottoreti a datagrammi: maggiore probabilità di congestioni ma anche più strumenti per aggirarle, robustezza nonostante i guasti dei *router* e necessità di intestazioni più lunghe. In una internet, inoltre, si possono adottare diversi algoritmi di *routing* adattativi. Un altro grande vantaggio di questo approccio è che può essere utilizzato su sottoreti che non usano al loro interno circuiti virtuali.

### *Routing* in una *internetwork*

Il *routing* attraverso una *internetwork* è simile al *routing* eseguito in una singola sottorete. Dopo aver costruito il grafo si possono applicare all'insieme di *router* multiprotocollo gli algoritmi di *routing* già noti, si ottiene quindi un algoritmo di *routing* a due livelli: dentro ogni rete viene utilizzato un **protocollo di *gateway* interno** (*interior gateway protocol*), ma tra le reti è adottato un **protocollo di *gateway* esterno** (*exterior gateway protocol*). Dato che nella *internetwork* ogni rete è indipendente da tutte le altre spesso le reti sono chiamate ***Autonoumos System*** (AS).
Un classico pacchetto internet nasce nella propria LAN, dove viene indirizzato al *router* multiprotocollo locale. Raggiunto il *router* il codice dello strato di *network* decide, usando le sue tabelle di *routing*, a quale *router* multiprotocollo deve essere inoltrato il pacchetto. Se quel *router* può essere raggiunto usando il protocollo di rete nativo del pacchetto questo è inoltrato direttamente, altrimenti i dati vengono trasmessi via *tunneling* (incapsulati nel protocollo richiesto dalla rete intermedia). Questo processo si ripete fino a quando il pacchetto non raggiunge la rete di destinazione.
Una delle differenze tra il *routing* nelle *internetwork* e quello interno alle reti è che il primo può richiedere l'attraversamento di confini internazionali, quindi improvvisamente entrano in gioco diverse leggi.. Una seconda differenza riguarda il costo: dentro una singola rete normalmente si applica un solo algoritmo di addebito, mentre reti diverse possono essere gestite da enti diversi e un percorso può risultare meno costoso di un altro. In modo analogo anche la qualità del servizio offerto dalle diverse reti può cambiare, e da questo può dipendere la scelta di un percorso piuttosto che di un altro.

## 5.6 LO STRATO *NETWORK* IN INTERNET

Nello strato *network* la rete di Internet può essere vista come un insieme di AS interconnessi. Non esiste alcuna vera struttura, esistono solo dorsali principali, formate da linee a banda larga e *router* veloci, alle quali si collegano reti di livello intermedio a cui si collegano a loro volta LAN. La colla che tiene unito Internet è il protocollo dello strato *network* **IP** (*Internet Protocol*), il cui compito è quello di fornire il *best effort* per trasportare i datagrammi inviati da una sorgente ad una destinazione senza tener conto della posizione delle macchine e della eventuale presenza di sottoreti.
La comunicazione Internet funziona in questo modo. Lo strato trasporto prende i flussi dati e li divide in datagrammi, ognuno dei quali viene frammentato in unità più piccole. Quando tutti i pezzi raggiungono la destinazione lo strato *network* ricostruisce il datagramma originale, che sarà passato allo strato di trasporto che lo inserisce nel flusso di input del processo ricevente.

## Il protocollo IP

Un datagramma IP è costituito da una parte di intestazione e da una parte di testo.
L'intestazione ha una parte fissa di 20 byte e una parte opzionale di lunghezza variabile per un totale di 60 byte. Viene trasmessa da sinistra a destra a partire dal bit di ordine più elevato del campo *version*, che tiene traccia della versione del protocollo usato per il datagramma per favorire la retrocompatibilità. Poiché la lunghezza non è costante, l'intestazione contiene un campo *IHL* che indica la lunghezza dell'intestazione stessa espressa in parole di 32 bit (minimo 5, massimo 15). Il campo *options* non può occupare più di 40 byte.
Il campo *type of service* era originariamente usato per distinguere le diverse classi di servizizo, per le quali sono ammesse diverse combinazioni di affidabilità e velocità, in pratica i *router* correnti spesso ignorano del tutto questo campo. *Total lenght* tiene conto di tutto il contenuto del datagramma e la sua lunghezza massima è 65.535 byte. Il campo *identification* serve all'*host* di destinazione per determinare a quale datagramma appartiene il frammento appena arrivato. Subito dopo c'è un bit inutilizzato e poi due campi lunghi 1 bit: DF è l'acronimo di *Don't Fragment* mentre MF di *More Fragments* (tutti i frammenti tranne l'ultimo hanno questo bit impostato a 1). *Fragment offset*, poi, indica la posizione del frammento nel datagramma corrente. Il campo *time to live* è un contatore utilizzato per limitare la vita di un pacchetto. Il campo *protocol* indica quale processo di trasporto è in attesa di quei dati. Iil campo *header checksum* verifica solo l'intestazione. *Source address* e *destination address* indicano il numero di rete e quello degli *host*. Le opzioni, infine, sono di lunghezza variabile: ognuna inizia con un codice di 1 byte che la identifica, alcune sono seguite da un campo *option lenght* di 1 byte che precede 1 o più byte di dati. Questo campo è riempito con multipli di 4 byte.

## Indirizzi IP

Ogni *host* e *router* di Internet ha un indirizzo IP che codifica il suo indirizzo di rete e il suo numero di *host*. Questa combinazione è unica e, per evitare conflitti, i numeri di rete sono gestiti da un ente no profit chiamato **ICANN** (*Internet Corporation for Assigned Names and Numbers*) il quale a sua volta affida la gestione di alcune parti dello spazio di indirizzamento ad autorità regionali.
Gli indirizzi di rete, rappresentati da numeri a 32 bit, sono di solito scritti in notazione decimale a punti, ovvero ciascuno dei 4 byte di cui è composto è rappresentato in forma decimale con un numero che varia tra 0 e 255. Il valore 0 indica la rete corrente (0.0.0.0), 1 invece è utilizzato come indirizzo *broadcast* (1.1.1.1). Tutti glil indirizzi espressi nella forma 127.*xx.yy.zz* sono riservati per le prove di *loopback*, ovvero i pacchetti diretti a quell'indirizzo non sono immessi nel cavo ma elaborati localmente e trattati come un pacchetto in arrivo senza che il trasmettitore conosca il suo numero.

| Classe | Struttura | Intervallo di indirizzi di *host* |
|---|---|---|
| A | 0 - Rete - *Host* | Da 1.0.0.0 a 127.255.255.255 |
| B | 10 - Rete - *Host* | Da 128.0.0.0 a 191.255.255.255 |
| C | 110 - Rete - *Host* | Da 192.0.0.0 a 223.255.255.255 |
| D | 1110 - Indirizzi *multicast* | Da 224.0.0.0 a 239.255.255.255 |
| E | 111 - Riservati per usi futuri | Da 240.0.0.0 a 255.255.255.255 |

**Sottoreti.** Tutti gli *host* di una rete devono avere lo stesso numero di rete. Questa proprietà dell'indirizzamento IP può causare dei problemi, ma la soluzione sta nel dividere internamente la rete in più parti (sottoreti) facendo modo che il mondo esterno veda ancora una singola rete.

Quando il pacchetto raggiunge il *router* principale per identificare la sottorete di destinazione dei dati è possibile memorizzare nel *router* principale una tabella contenente 65.536 voci che indicano il *router* da utilizzare per ogni *host*, ma questo richiederebbe l'implementazione nel *router* principale di una tabella molto grande e impegnativa da gestire. Per questo motivo alcuni bit sono tolti al numero di *host* per creare il numero di sottorete. Per implementare le sottoreti il *router* principale ha bisogno di una ***subnet mask*** che indichi il punto di demarcazione tra il numero di rete (incluso quello di sottorete) e quello di *host*. Anche le *subnet mask* sono scritte in notazione decimale a punti, oppure sono segnate da una barra seguita dal numero di bit della parte che rappresenta la rete più la sottorete.
Ogni *router* ha una tabella che elenca i numeri di indirizzi IP del tipo (rete, 0) che si riferiscono a reti distanti e numeri di indirizzi IP del tipo (questa rete, *host*) che riguardando *host* locali. Ad ogni tabella è associata la scheda di rete da utilizzare per raggiungere la destinazione. Quando riceve un pacchetto IP il *router* cerca il suo indirizzo di destinazione nella tabella di *routing* e se è indirizzato ad una rete distante il pacchetto viene inoltrato al *router* successivo, mentre se la destinazione è un *host* locale il pacchetto viene inviato direttamente alla destinazione. Se la rete non viene trovata sulla tabella il pacchetto viene inoltrato ad un *router* predefinito contenente tabelle più estese. Quando si introduce la tecnica della divisione in sottoreti le tabelle di *routing* cambiano, vengono aggiunte voci espresse nella forma (questa rete, sottorete, 0) e (questa rete, questa sottorete, *host*). Un *router* su una sottorete *k*, quindi, sa come raggiungere tutte le altre sottoreti e sa anche come raggiungere tutti gli *host* della sottorete *k*.

**CIDR (*Classless InterDomain Routing*).** Grazie alla crescita esponenziale di Internet IP sta esaurendo gli indirizzi. Inizialmente esistevano più di due miliardi di indirizzi, ma il sistema di suddivisione basato sulle classi ne ha sprecati milioni. In particolare il vero problema è rappresentato dalla rete di classe B: per la maggior parte delle società una rete di classe A (16 milioni di indirizzi) è troppo grande e una rete di classe C (256 indirizzi) è troppo piccola. In realtà un indirizzo di classe B è di gran lunga troppo grande per la maggior parte delle aziende, ma senza dubbio ognuna di quelle che ha chiesto un indirizzo in questa classe pensava che un giorno gli 8 bit del campo *host* non sarebbero più stati sufficienti. Se fossero stati assegnati 20 bit al numero di rete di classe B sarebbe comunque emerso il problema dell'esplosione delle tabelle di *routing*. L'archiviazione fisica delle tabelle è probabilmente un operazione fattibile, anche se costosa. Il problema serio riguarda gli algoritmi che si occupano della gestione delle tabelle. Inoltre vari algoritmi di *routing* richiedono che ogni *router* trasmetta periodicamente le proprie tabelle, che più sono grandi e più è probabile che arrivino incomplete. Questo problema potrebbe essere risolto tramite una gerarchia più ramificata, ma questa soluzione richiederebbe un numero di bit maggiore dei 32 utilizzati dagli indirizzi IP.
La soluzione implementata per dare ad Internet un po' di respiro si chiama **CIDR**, e consiste nell'assegnare gli indirizzi IP rimanenti in blocchi di dimensioni variabili senza tener conto delle classi. Adesso ogni voce della tabella di *routing* è integrata da una maschera di 32 bit, perciò c'è una singola tabella di *routing* per tutte le reti costituita da una matrice di valori nella forma (indirizzo IP, maschera di sottorete, linea di output). Quando arriva il pacchetto il *router* estrae prima di tutti il suo indirizzo IP di destinazione e poi esamina la tabella di *routing* voce per voce mascherando l'indirizzo di destinazione e confrontando le voci della tabella. Nel caso più voci con lunghezze di maschera di sottorete diverse coincidano con il valore cercato è utilizzata la maschera più lunga.

**NAT (*Network Address Traslation*).** La scarsità degli indirizzi IP ha come soluzione a lungo termine quella che tutta Internet passi ad IPv6, protocollo che adotta indirizzi a 128 bit. Questa transazione sta procedendo lentamente e, nel frattempo, è stata adottata la soluzione a breve termine **NAT**. L'idea di base è di assegnare ad ogni azienda un singolo indirizzo IP per il traffico di Internet: all'interno di ogni azienda ogni computer riceve un indirizzo IP unico utilizzato per instradare il traffico interno alla rete locale, ma quando il pacchetto lascia l'azienda viene eseguita una

traduzione di indirizzo. Per rendere fattibile questo schema sono stati dichiarati privati tre intervalli di indirizzi IP (che non devono essere contenuti in nessun pacchetto che appare su Internet): 10.0.0.0 - 10.255.255.255/8, 172.16.0.0 - 172.31.255.255/12 e 192.168.0.0 - 192.168.255.255/16. Dentro un'azienda ogni macchina ha un unico indirizzo espresso nella forma 10.*x*.*y*.*z*, ma quando un pacchetto lascia l'azienda passa attraverso un dispositivo NAT che converte l'indirizzo IP interno nel vero indirizzo IP assegnato all'azienda. Il dispositivo NAT è spesso abbinato ad un *firewall* che protegge la rete locale controllando il flusso di dati in entrata e in uscita dalla LAN. Quando un processo desidera stabilire una connessione TCP/UDP con un processo remoto, questo si lega ad una porta inutilizzata presente nella sua macchina (porta sorgente) e indica al codice TCP/UDP dove devono essere inviati i pacchetti in arrivo appartenenti alla connessione. Il processo fornisce anche una porta di destinazione che indica chi deve ricevere i pacchetti sulla parte remota. Quando il pacchetto trasmesso dall'ISP raggiunge il dispositivo NAT, la *source port* nell'intestazione TCP/UDP viene estratta e utilizzata come indice nella tabella di mappatura del dispositivo NAT. Dalla voce individuata il dispositivo estrapola l'indirizzo IP interno e la *source port* originale, quindi inserisce entrambe le informazioni nel pacchetto, ricalcola e inserisce sia il *checksum* IP che quello TCP/UDP. Alla fine il pacchetto viene passato al *router* aziendale per il normale inolto.

## Protocolli di controllo Internet

**ICMP (*Internet Control Message Protocol*).** Viene usato per testare Internet e per comunicare quando avviene qualcosa di imprevisto.

| Messaggio | Descrizione |
|---|---|
| *DESTINATION UNREACHABLE* | Destinazione irraggiungibile, il pacchetto potrebbe non essere inoltrato |
| *TIME EXCEEDED* | Tempo superato, il campo *time to live* ha raggiunto il valore 0 |
| *PARAMETER PROBLEM* | Problema di parametri, campo dell'intestazione non vallido |
| *SOURCE QUENCH* | Spegnimento della sorgente, pacchetto di interruzione |
| *REDIRECT* | Reindirizzamento, insegna al *router* la geografia |
| *ECHO* | Eco |
| *ECHO REPLY* | Risposta a eco |
| *TIMESTAMP REQUEST* | Richiesta di cotnrassegno temporale |
| *TIMESTAMP REPLY* | Risposta alla richiesta di contrassegno temporale |

**ARP (*Address Resolution Protocol*).** Anche se ogni macchina di Internet ha uno o più indirizzi IP in realtà questi non possono essere utilizzati per inviare pacchetti perché l'*harware* che opera sullo strato *data link* non comprende gli indirizzi Internet. Il problema potrebbe essere risolto mediante un file di configurazione collocato da qualche parte nel sistema che associ gli indirizzi IP agli indirizzi *Ethernet*, ma la gestione di questo file potrebbe facilmente creare errori oltre che essere molto onerosa. La soluzione migliore è utilizzare l'**ARP** per interrogare in maniera *broadcast* tutte

le mecchine della *Ethernet* sull'identità dell'*host* con un determinato IP, richiedendo come risposta l'indirizzo *Ethernet* corrispondente. Alcune ottimizzazioni permettono ad ARP di funzionare in modo più efficiente. Tanto per cominciare una volta eseguita l'operazione ARP il computer memorizza in *cache* il risultato e includa la propria associazione IP-*Ethernet* nel pacchetto ARP. Inoltre ogni computer trasmette *broadcast* la propria associazione durante l'accensione sotto forma di ricerca ARP mirata al proprio indirizzo.

**DHCP (*Dynamic Host Configuration Protocol*).** DHCP permette l'assegnazione manuale o automatica degli indirizzi IP e si basa sull'idea di un server speciale che assegna gli indirizzi IP gli *host* che ne richiedono uno. Poiché il server DHCP potrebbe non essere raggiunto dalle trasmissioni *broadcast* è necessario installare in ogni LAN un agente di inoltro DHCP, incaricato di inviare in modalità *unicast* al server la richiesta dell'*host*. Se un *host* abbandona la rete senza restituire il proprio indirizzo IP al server questo andrà definitivamente perso, per impedire questo è stata pensata una tecnica di *leasing* per cui viene fissata una scadenza temporale dell'indirizzo e se non viene rinnovata la richiesta da parte dell'*host* viene riconsegnato al server.

## OSPF - Il protocollo di *routing* per i *gateway* interni

Inizialmente il protocollo di *routing* per *gateway* interni di Intenet era un protocollo basato sul vettore delle distanze (RIP), ma numerosi problemi hanno portato allo sviluppo di **OSPF** (*Open Shortest Path First*) che è diventato il principale protocollo in questo ambito. OSPF supporta tre tipi di connessioni e di reti: linee punto-punto tra due *router*, reti multiaccesso con e senza trasmissione *broadcast*. Questo protocollo opera riassumento l'insieme di reti reali, *router* e linee in un grafo orientato in cui ad ogni arco è assegnato un costo, e quindi calcola il percorso più breve.
Dato che molti AS di Internet sono a loro volta complessi, OSPF permette di dividere tali AS in più aree dove ogni area è una rete o un insieme di reti contigue. Le aree non si sovrappongono e non hanno bisogno di essere complete, inoltre non sono visibili dall'esterno. Ogni area ha una dorsale chiamata **area 0** a cui sono collegate tutte le altre aree, questa comprende tutti i *router* collegati a due o più aree. Anche la topologia della dorsale è invisibile dall'esterno della stessa. dentro un'area ogni *router* ha lo stesso database degli stati dei collegamenti e segue lo stesso algoritmo per l'individuazione del percorso più breve.
Il *flooding* permette ad ogni *router* di comunicare a tutti gli altri *router* della sua area chi sono i vicini e i costi utilizzati. Queste informazioni permettono ad ogni *router* di costruire un grafo dell'area e di calcolare il percorso più corto. I *router* della dorsale fanno lo stesso, e inoltre accettano informazioni dai *router* di confine d'area in modo da poter calcolare il percorso migliore diretto da ogni *router* di dorsale ad ogni altro *router*. Queste informazioni si propagano all'indietro all'interno dell'area e utilizzando questi dati un *router* che sta per inviare un pacchetto da un'area all'altra può selezionare il *router* di uscita migliore verso la dorsale.

## BGP - Il protocollo di *routing* per i *gateway* esterni

Tra AS si utilizza un protocollo chiamato **BGP** (*Border Gateway Protocol*), che si occupa non solo di spostare i pacchetti ma deve anche rispettare certi criteri politici, di sicurezza o economici che sono configurati manualmente in ogni *router* BGP (o inseriti usando qualche *script*) che non fanno parte del protocollo stesso. Le reti sono raggruppate in tre categorie: le ***stub networks*** che hanno una sola connessione al grafo BGP e non possono essere utilizzate per il traffico di passaggio, le ***multiconnected networks*** che potrebbero essere utilizzate per il traffico di passaggio se non si rifiutano, e le ***transit networks*** che sono pronte a gestire pacchetti di terze parti.
BGP è un protocollo basato sul vettore delle distanze che invece di conservare solo il costo di ogni

destinazione tiene traccia del percorso utilizzato e, analogamente, invece di comunicare periodicamente ad ogni vicino il costo stimato associato ad ogni possibile destinazione comunica l'esatto percorso in uso.

## Intenet *multicasting*

IP supporta la trasmissione *multicast* utilizzando gli indirizzi di classe D. Ogni indirizzo identifica un gruppo di *host* e 28 bit sono utilizzati per identificare i gruppi. Quando un processo invia un pacchetto ad un indirizzo di classe D viene fatto un tentativo *best effort* per trasmettere i dati a tutti i membri del gruppo di destinazione, ma non viene data alcuna garanzia. Sono supportati due tipi di indirizzi di gruppo: permanenti e temporanei.
La trasmissione *multicast* è implementata da speciali *router multicast* che eventualmente possono essere abbinati a *router standard*. Circa una volta al minuto ogni *router multicast* genera una trasmissione *hardware* che chiede alle macchine di indicare i gruppi di appartenenza dei loro processi. Ogni *host* risponde comunicando tutti gli indirizzi di classe D che gli interessano. Questi pacchetti di interrogazione e risposta utilizzano un protocollo chiamato **IGMP** (*Intenert Group Management Protocol*) che assomiglia vagamente all'ICMP. Il *routing multicast* è eseguito usando le strutture *spanning tree*: ogni *router* scambia informazioni con i suoi vicini usando un protocollo basato sul vettore delle distanze e questo consente a ogni *router* di costruire per ogni gruppo uno *spanning tree* che copre tutti i membri del gruppo. Il protocollo usa in modo pesante i tunnel per evitare di infastidire i nodi che non fanno parte dello *spanning tree*.

CAPITOLO 6

# LO STRATO DI TRASPORTO

## 6.2 Gli elementi dei protocolli di trasporto

### Stabilire la connessione

L'***handshake* a tre vie** non richiede a entrambe le parti di iniziare la trasmissione partendo dallo stesso numero di sequenza, pertanto può essere utilizzato con metodi di sincronizzazione diversi dal metodo dell'orologio globale.
La normale procedura per stabilire una connessione quando l'*host* 1 viene inizializzato prevede che quest'ultimo scelga il numero di sequenza *x* e invii una TPDU *CONNECTION REQUEST* contenente *x* all'*host* 2. L'*host* 2 risponde con una TPDU *ACK* che riconosce *x* e annuncia il suo numero di sequenza iniziale, *y*. Per finire l'*host* 1 riconosce la scelta del numero di sequenza iniziale dell'*host* 2 nella prima TPDU dati inviata. In caso di TPDU duplicate o ritardate la prima TDPU REQUEST si riferisce ad una vecchia connessione e arriva all'*host* 1 senza che l'*host* 2 lo sappia. L'*host* 2 reagisce inviando all'*host* 1 una TPDU *ACK* e, quando questo rifiuta il tentativo di stabilire una connessione abbandona l'intento.
Non esiste combinazione di vecchie TDPU che può provocare il fallimento del protocollo o l'instaurarsi accidentale di una connessione indesiderata.

### Il rilascio della connessione

Esistono due modi per terminare una connessione. Nel **rilascio asimmetrico** quando una delle due parti riattacca la connessione viene interrotta, questa disconnessione è improvvisa e può causare la perdita di dati. Il **rilascio simmetrico** tratta la connessione come se fosse composta da due connessioni unidirezionali e chiede il rilascio separato di ognuna di esse, svolge il suo compito quando ogni processo presenta una quantità fissa di dati da inviare e sa chiaramente quando li ha inviati in quanto determinare queste due variabili non è un'operazione così ovvia.
Uno degli utenti invia una TPDU *DISCONNECTION REQUEST* per iniziare il rilascio di una connessione. Quando arriva il destinatario restituisce una TPDU DR e avvia un timer. All'arrivo di questa DR il mittente originale invia una TPDU *ACK* e rilascia la connessione, quando questa arriva il ricevitore rilascia a sua volta la connessione. Questo protocollo è generalmente sufficiente, ma in teoria può fallire se vengono perse la DR iniziale e le *N* ritrasmissioni: il mittente abbandonerà i tentativi e rilascierà la connessione mentra l'altro lato non saprà nulla di tutti i tentativi di disconnessione e sarà pienamente attivo. Questa situazione provoca una connessione aperta a metà. Un modo per interrompere tutte le connessioni aperte a metà può essere la definizione di una regola del tipo: se non arrivano TPDU per un certo numero di secondi la connessione viene automaticamente terminata.

## 6.4 I protocolli di trasporto Internet: UDP

### Introduzione a UDP

La suite di protocolli Internet supporta un protocollo di trasporto senza connessione, vale a dire **UDP** (*User Datagram Protocol*). UDP trasmette segmenti costruiti da un'intestazione idi 8 byte seguita dal carico utile. All'interno di questi segmenti sono presenti due campi in cui vengono specificate le porte che servono per identificare i punti finali all'interno dei computer di origine e destinazione, quando arriva un pacchetto infatti UDP il suo carico utile è consegnato al processo associato alla porta di destinazione mentre la porta di orgine serve principalmente quando si deve inviare una risposta all'origine.
Il protocollo in questione non si occupa del controllo del flusso, degli errori o della ritrasmissione dopo la ricezione di un segmento errato. Questo compiti sono lasciati ai processi utente. UDP si occupa invece di fornire un'interfaccia al protocollo IP, con la caratteristica aggiunta del *demultiplexing* di più processi utilizzando le porte.

## 6.5 IL PROTOCOLLO DI TRASPORTO INTERNET: TCP

### Introduzione a TCP

Il **TCP** (*Transmission Control Protocol*) è stato progettato per fornire un flusso di byte affidabile *ent-to-end* su una *internetwork* inaffidabile. Ogni computer che supporta TCP dispone di un'entità di trasporto TCP che gestisce i flussi TCP e le interfacce verso lo strato IP. Un'entità di questo tipo accetta dai processi locali i flussi dati dell'utente, li suddivide in pezzi e invia ogni pezzo in un datagramma IP autonomo. Questi arrivano al computer e vengono consegnati all'entità TCP che ricostruisce i flussi di byte originali.
Lo strato IP non garantisce la consegna senza errori dei datagrammi, pertanto è TCP che deve eseguire il *timeout* e la ritrasmissione secondo necessità. I datagrammi potrebbero anche arrivare nell'ordine sbagliato, è sempre compito di TCP riassemblare i messaggi nella giusta sequenza.

### Il modello di servizio TCP

Il servizio TCP è ottenuto con la creazione di punti finali da parte i mittente e ricevente, che vengono chiamati ***socket***. Ogni *socket* possiede un indirizzo composto dall'indirizzo IP dell'*host* e da un numero di 16 bit locale all'*host* chiamato porta (nome TCP per TSAP) e supporta più connessioni contemporaneamente. I numeri di porta minori di 1.024 identificano le ***well-known ports*** e sono riservati ai servizi standard. Per ottenere il servizio TCP si deve stabilire esplicitamente una connessione tra un *socket* sulla macchina di invio e uno sulla macchina ricevente.
In genere si preferisce avviare un singolo *daemon* FTP (che in UNIX si chiama **inetd**, Internet *daemon*) associato a più porte per attendere la prima connessione in ingresso. Quando questa si verifica inetd genera un nuovo processo e vi esegue il *daemon* appropriato consentendogli di gestire la richiesta, in questo modo i *daemon* diversi da inetd sono attivi solo quando hanno il loro lavoro da compiere. Inetd apprende quali porte deve utilizzare da un file di configurazione, di conseguenza l'amministratore di sistema può configurare il computer perché disponga di *daemon* permanenti

sulle porte più usate mentre inetd si occupa delle altre.
Le caratteristiche principali del TCP sono che tutte le connessioni ono ti tipo *full-duplex* punto-punto e non supportano il *multicasting* o il *broadcasting*. Oltre a questo è importante sapere che una connessione TCP è un flusso di byte, e non di messaggi. Questo comporta che i confini dei messaggi non vengano conservati da una parte all'altra.
Quando un'applicazione passa i dati a TCP, questo a sua discrezione può inviarli immediatamente o inserirli in un buffer per raccoglierne una quantità maggiore da inviare tutta insieme. Per forzare l'uscita di dati le applicazioni possono usare il flag *PUSH* per comunicare a TCP di non ritardare la trasmissione. Allo stesso modo, quanto un utente interattivo preme i tasti CANC o CTRL+C per interrompere un'elaborazione remota che ha già avuto inizio l'applicazione trasmittente inserisce alcune informazioni di controllo nel flusso dei dati e le consegna a TCP con il flag *URGENT*, in maniera da obbligare TCP a interrompere l'accumulazione dei dati e trasmettere immediatamente tutto ciò che ha a disposizione per quella connessione. La fine dei dati urgenti è contrassegnata, in modo che l'applicazione sappia dove terminano, al contrario l'inizio dei dati invece non è contrassegnato ed è compito dell'applicazione calcolarlo.

## Il protocollo TCP

Una funzionalità vitale di TCP consiste nel fatto che ogni byte in una connessione di questo tipo ha un proprio numero di sequenza di 32 bit. Le entità di invio e ricezione scambiano dati sotto forma di segmenti, che consistono di un'intestazione fissa di 20 byte seguita da zero o più byte dati. Il software TCP decide la dimensione dei segmenti, e può accumulare in un segmento dati provenienti da più scritture oppure dividere i dati di una scrittura in più segmenti. I limiti di dimensione sono due: ogni segmento, intestazione compresa, deve essere contenuto in un carico utile di 65.515 byte di IP e dato che ogni rete ha una MTU (*Maximum Transfer Unit*), che è generalmente lunga 1.500 byte, ogni segmento deve esserci contenuto.
Il protocollo di base utilizzato da TCP è lo ***sliding window***. Quando un mittente trasmette un segmento avvia anche un timer. Quando il segmento arriva a destinazione l'entità TCP ricevente invia un segmento contrassegnato da un numero di *acknowledgement* uguale al numero di sequenza successivo che prevede di ricevere. Se il timer del mittente scade prima della ricezione dell'*acknowledgement* il mittente ritrasmette il segmento.
I principali problemi di questo protocollo sono che i segmenti possono arrivare nell'ordine sbagliato in quanto durante il transito i segmenti possono essere ritardati così tanto che il mittente va in timeout ed è costretto a ripetere la trasmissione, costringendo ad un'attenta gestione per tenere traccia di quali byte sono stati ricevuti correttamente a ogni istante. Questa operazione può essere svolta tranquillamente, in quanto ogni byte nel flusso ha un offset univoco.

## L'intestazione del segmento TCP

Ogni segmento inizia con un'intestazione di 20 byte con formato fisso, che può essere seguita da alcune opzioni dell'intestazione. Dopo le opzioni seguono fino a 65.535 - 20 - 20 = 65.495 byte di dati, dove i primi 20 fanno riferimento all'intestazione IP gli altri a quella TCP.
I campi *source port* e *destination port* identificano gli estremi locali della connessione. Il campo TCP *header lenght* indica quante parole da 32 bit sono contenute nell'intestazione TCP, segue un campo di 6 bit inutilizzato. Seguono i sei flag di 1 bit: URG è impostato quando si usa *urgent pointer*, che indica l'offset in cui si trovano i dati urgenti, il bit ACK è impostato a 1 per indicare che *acknowledgement number* è valido, il PSH segnala la presenza di dati *PUSH*, RST viene utilizzato per reimpostare una connessione che è diventata incongruente, SYN viene utilizzato per stabilire connessioni e FIN per rilasciarle. Il campo *window size* indica quanti byte possono essere

inviati a partire da quello che ha ricevuto *acknowledgement*. Per una maggiore flessibilità viene anche fornito un campo *checksum*. Il campo *options*, infine, è un modo per aggiungere caratteristiche extra non disponibili nella normale intestazione, l'opzione più importante è quella che permette a ogni *host* di specificare il carico utile massimo che TCP potrà accettare.

## Costituzione della connessione TCP

Le connessioni TCP vengono stabilite mediante l'*handshake* a tre vie. Un lato attende in modo passivo una connessione in ingresso eseguendo le primitive *LISTEN* e *ACCEPT* indicando un'origine specifica oppure nessuna in particolare, l'altro lato invece esegue la primitiva *CONNECT* specificando l'indirizzo IP e la porta a cui vuole connettersi, la dimensione massima del pacchetto e facoltativamente altri dati utente. La primitiva *CONNECT* invia un segmento TCP con il bit SYN a 1 e ACK a 0 e poi attende una risposta. Quando questo segmento arriva a destinazione l'entità TCP controlla se esiste un processo che ha eseguito un *LISTEN* sulla porta indicata e in caso negativo invia una risposta con il bit RST a 1 per rifiutare la connessione, se invece un processo è in ascolto sulla porta riceve il segmento in ingresso e può accettare o rifiutare la connessione. Nel caso in cui accetti viene restituito un segmento di *acknowledgement*.
Nel caso due *host* tentino contemporaneamente di stabilire una connessione tra gli stessi due *socket* il risultato è la costituzione di una sola connessione, e non due. Questo perché le connessioni sono identificate dai loro punti finali. Quando, infine, un *host* subisce un *crash* potrebbe non essere riavviato per un tempo pari a quello di vita massimo del pacchetto, in modo da garantire che non vi siano in circolazione su Internet pacchetti riferiti a connessioni precedenti.

## Il rilascio della conenssione TCP

Per rilasciare una connessione entrambe le parti possono inviare un segmento TCP con bit FIN impostato, per indicare che non hanno più dati da trasmettere. Quando FIN riceve l'*acknowledgement* la direzione viene chiusa, tuttavia i dati possono continuare a fluire indefinitamente nella direzione opposta. Quando entrambe le direzioni saranno state chiuse la connessione sarà rilasciata. Se una risposta a FIN non arriva entro la durata massima di due pacchetti, il mittente rilascia la connessione e l'altro andrà in timeout.
Normalmente sono necessari quattro segmenti per rilasciare una connessione: un FINe un ACK per ogni direzione. Se il primo FIN e il secondo ACK vengono inseriti nello stesso segmento, tuttavia, è possibile ridurre il totale a tre.

CAPITOLO 7

# LO STRATO APPLICAZIONE

## 7.1 DNS: il sistema dei nomi di dominio

L'essenza del **DNS** (*Domain Name System*) è uno schema di denominazione gerarchico basato su dominio, e di un sistema di database distribuito per l'implementazione di questo schema di denominazione. Il suo utilizzo principale è quello di associare nomi di *host* e destinazioni di posta elettronica agli indirizzi IP, ma ci si può ricorrere anche per antri scopi.
Per associare un nome ad un indirizzo IP un programma applicativo chiama una procedura di libreria chiamata **risolutore** passando il nome come parametro. Il risolutore invia un pacchetto UDP ad un server DNS locale, che quindi cerca il nome e restituisce l'indirizzo IP al risolutore, che a sua volta lo restituisce al chiamante. Armato di indirizzo IP il programma può quindi stabilire una connessione TCP con la destinazione oppure inviarle pacchetti UDP.

### Lo spazio dei nomi DNS

Internet è divisa in **donini** di primo livello, dove ogni dominio copre molti *host*. Ogni dominio è partizionato in sottodomini, che sono a loro volta divisi. Tutti questi domini possono essere rappresentati in una struttura ad albero dove le foglie rappresentano i domini che non hanno sottodomini: ogni foglia può contenere uno o più *host*.
I domini di primo livello sono di due tipi: generici (originariamente *com*, *edu*, *int*, *mil*, *net*, *org*) e per nazioni. Per ottenere domini di secondo livello è necessario contattare un ***registrator*** (autorità di registrazione domini) per il dominio di primo livello corrispondente per controllare se il nome desiderato è disponibile e se altri non possiedono il marchio, se non ci sono problemi il richiedente paga una tariffa annuale e ottiene il nome. Ogni dominio controlla come allocare i domini sottostanti: per creare un nuovo dominio è necessaria l'autorizzazione del dominio in cui sarà incluso.
Ogni dominio è denominato da un percorso compreso tra esso e la radice, non denominata, i cui componenti sono separati da punti. I nomi di dominio sono *case insensitive* e possono essere assoluti o relativi: un nome di dominio assoluto termina sempre con un punto, al contrario di un nome relativo che deve essere interpretato in un contesto per determinare univocamente il reale significato.

### I record delle risorse

Ad ogni dominio è associato un insieme di ***resource records***. Quando un risolutore fornisce il nome di dominio a DNS ciò che ottiene sono i record delle risorse associati a tale nome, di conseguenza la principale funzione di DNS è associare i nomi di dominio ai record di risorse.
Un record di risorse è una quintupla di tipo `Domain_name Time_to_live Class Type Value`. `Domain_name` indica il dominio a cui si riferisce il record, `Time_to_live` offre un indicatore di stabilità del record stesso, `Class` indica il tipo di informazioni contenute, `Type` specifica il tipo di record e `Value`, che a seconda del tipo di record può contenere un numero, un nome di dominio o una stringa ASCII.

## I server dei nomi

Lo spazio dei nomi DNS viene diviso in **zone** non sovrapposte, che contengono alcune parti della struttura e i server dei nomi col le informazioni della zona in questione. Di norma una zona avrà un server dei nomi primario che ottiene le sue informazioni da un file su disco e uno o più server dei nomi secondari che ottengono le loro informazioni dal server dei nomi primario, per migliorare l'affidabilità alcuni server per una zona possono essere situati all'esterno della zona stessa. La posizione dei confini di zona all'interno della zona è una scelta lasciata all'amministratore.
Quando un risolutore ha un'interrogazione su un nome di dominio passa l'interrogazione ad uno dei server dei nomi locali. Se un dominio è all'interno della giurisdizione del server dei nomi resituisce record autorevoli delle risorse (ovvero un recordo fornito dall'autorità che gestisce il record ed è pertanto sempre corretto, si contrappone ai record archiviati nella *cache* che potrebbero non essere aggiornati). Se il dominio è remoto e non sono disponibili localmente informazioni su di esso il server dei nomi invia un messaggio d'interrogazione al server dei nomi di primo livello per il dominio richiesto. Questo metodo è conosciuto come **interrogazione ricorsiva**, perché ogni server che non possiede le informazioni richieste deve trovarle altrove e poi restituirle. Una forma alternativa prevede che quando un'interrogazione non può essere soddisfatta localmente e quindi fallisce viene restituito il nome del successivo server da provare ad usare, quando un *client* DNS non riesce ad ottenere una risposta prima della scadenza del suo *timer* la volta successiva proverà ad usare un altro *server*.
Anche se DNS è molto importante per il corretto funzionamento di Internet il suo unico compito è quello di associare nomi simbolici per le macchine ai loro indirizzi IP. **LDAP** (*Lightweight Directory Access Protocol*) fornisce invece una struttura ad albero che supporta la ricerca per diversi elementi, come una sorta di rubrica.

CAPITOLO 8

# SICUREZZA DELLE RETI

La sicurezza è un argomento ampio che copre una moltitudine di problemi, e nella sua forma più semplice riguarda come fare in modo che intrusi non riescano a leggere/modificare i messaggi destinati a terzi. Si occupa inoltre di impedire che determinate persone possano accedere a diversi servizi remoti che non sono autorizzate ad usare, di come accertarsi dell'identità dei mittenti dei messaggi, di come impedire l'intercettazione e la ripetizione di messaggi legittimi catturati sulla rete e di come perseguire chi afferma di non aver mai spedito certi messaggi.
Nello **strato fisico** l'ascolto del canale può essere evitato racchiudento le linee di trasmissione dentro ad un tubo sigillato che contiene del gas ad alta pressione, in modo che ogni tantativo di forare il tubo rilascerà il gas riducendo la pressione e facendo quindi scattare un allarme. Nello **strato *data link*** i pacchetti di una linea punto-punto possono essere cifrati quando lasciano un computer e poi decifrati appena entrano in un altro in maniera tale che tutti i dettagli siano gestiti dallo strato in questione e che quelli superiori non si accorgano di queste operazioni (soluzione non attuabile quando i pacchetti devono attraversare più *router* perché questi dovrebbero essere decifrati all'interno di ogni *router* e sarebbero quindi vulnerabili ad attacchi all'interno del *router* stesso). Nello **strato *network*** si possono installare *firewall* per filtrare i pacchetti buoni e respingere quelli cattivi. Nello **strato trasporto** l'intera connessione può essere cifrata da capo a capo, ovvero da processo sorgente a processo ricevente. Nello **strato applicativo** possono essere trattati problemi come l'autenticazione degli utenti e il ripudio/non-ripudio.

## 8.1 CRITTOGRAFIA

**C**rittografia deriva da due parole in greco antico che significano "scrittura segreta". Con il termine **cifrario** si intende una trasformazione carattere per carattere (o bit per bit), senza considerare la struttura linguistica del messaggio, mentre un **codice** rimpiazza ogni parola con un'altra parola o un simbolo. I codici non vengono più utilizzati.

### Introduzione alla crittografia

I messaggi da cifrare (**testi in chiaro**) sono trasformati tramite una funziona parametrica da una **chiave**. L'output del processo di cifratura (**testo cifrato**) viene generalmente trasmesso. Al contrario del destinatario legittimo un eventuale **intruso** non conosce la chiave di decifrazione e quindi non riesce facilmente a decifrare il testo cifrato. L'arte di decriptare i cifrari, chiamata **criptoanalisi**, e l'arte di inventarli, ovvero la **crittografia**, sono note sotto il nome collettivo di **crittologia**.
**Decriptare** è l'attività di decifrazione da parte del criptoanalista (intruso), mentre **decifrare** è l'operazione legittima di lettura del messaggio cifrato.
Una regola fondamentale della crittografia, nota come **Principio di *Kerkrckhoff***, afferma che bisogna sempre assumere che il criptoanalista conosca il metodo usato per la cifratura e la decifrazione. Qui entrano in gioco le chiavi crittografiche, ovvero stringhe relativamente corte che identifica una particolare cifratura fra molte possibilità. Il nostro modello base è dato, quindi, da un modello generale stabile e noto pubblicamente parametrizzato da una chiave segreta che può essere facilmente cambiata.

**Principio di *Kerckhoff*: tutti gli algoritmi devono essere pubblici, solo le chiavi sono segrete.**
Cercare di tenere segreto l'algoritmo, metodo noto come sicurezza per occultamento, non funziona mai. Pubblicizzando l'algoritmo il crittografo ottiene, inoltre, la consulenza gratuita di un gran numero di accademici desiderosi di forzare il sistema. Visto che la vera segretezza sta nella chiave, la sua lunghezza è un argomento importante nella progettazione di un sistema crittografico: più lunga è la chiave e più è alto il fattore lavoro che il criptoanalista deve mettere in conto, in quanto questo cresce esponenzialmente con la lunghezza della chiave.
Dal punto di vista del criptoanalista i problemi di criptoanalisi hanno tre variazioni principali: quando si ha solo il testo cifrato e nessun testo chiaro si parla di un problema con solo testo cifrato, quando invece si ha a disposizione del testo cifrato con il corrispondente testo in chiaro si parla di problemi con testo in chiaro noto e, infine, quando il criptoanalista ha la possibilità di cifrare dei pezzi di testo in chiaro a scelta parliamo di problema con testo in chiaro a scelta.

## Cifrari a sostituzione

In un **cifrario a sostituzione** ogni lettera, o gruppo di lettere, viene rimpiazzato da un'altra lettera, o gruppo di lettere, per mascherare il messaggio. Il sistema generale per la sostituzione simbolo a simbolo viene chiamato **sostituzione monoalfabetica**, dove la chiave è la stringa di lettere che corrisponde all'intero alfabeto.
L'attacco base fa leva sulle proprietà statistiche dei linguaggi naturali, quindi un criptoanalista che voglia forzare un cifrario simile comincerebbe contando la frequenza relativa di tutte le lettere del testo cifrato e poi procedendo per tentativi oppure indovinare una parola/frase che probabilmente si trovano dentro al testo.

## Cifrari a trasposizione

In un **cifrario a trasposizione** vengono riordinate le lettere, ma non vengono mascherate. Lo scopo della chiave è quello di riordinare le colonne.
Per forzare un cifrario di questo tipo il criptoanalista per prima cosa deve sapere che si tratta di un cifrario a trasposizione. Il passo successivo consiste nel fare ipotesi sul numero di colonne, mentre il passo conclusivo consiste nel trovare l'ordine delle colonne.

## Blocchi monouso

Il metodo per costruire un cifrario imbattibile è noto come **blocco monouso** (*one-time pad*). Per prima cosa si prende una chiave formata da una stringa di bit generati a caso e poi si converte il testo in chiaro in un'altra stringa di bit, infine si fa la XOR delle due stringhe, bit per bit. Il testo cifrato risultante non può essere forzato. Questo metodo è immune a tutti gli attacchi presenti e futuri indipendentemente da quanta potenza di calcolo l'intruso abbia a disposizione in quanto il messaggio cifrato non contiene nessuna informazione ma contiene tutti i possibili testi in chiaro di una data lunghezza con eguale probabilità.
I blocchi monouso sono ottimi in teoria, ma in pratica presentano i seguenti svantaggi. La chiave non può essere imparata a memoria, quindi sia mittente che destinatario devono averne una copia. Inoltre la quantità di dati che può essere trasmessa è limitata alla lunghezza della chiave. L'ultimo problema è la debolezza nel trattare i caratteri persi o inseriti per sbaglio.

## Due principi crittografici fondamentali

**Ridondanza.** Il destinatario, dopo aver decifrato il messaggio, deve essere in grado di stabilirne la validità con un semplice esame del contenuto o un breve calcolo. Questo tipo di ridondanza è necessario per prevenire gli attacchi degli intrusi attivi, che consistono nell'inviare messaggi generati a caso per ingannare il destinatario che li decifra come fossero validi. Il rovescio della medaglia è che la stessa ridondanza rende più semplice per gli intrusi passivi il compito di decifrare i messaggi. Inoltre la ridondanza non dovrebbe mai presentarsi nella forma di *n* valori a zero posti all'inizo o alla fine del messaggio.
La soluzione migliore consiste nell'usare dei codici *hash* crittografici.

**Attualità.** Il secondo principio afferma che è necessario avere la possibilità di verificare che ogni messaggio ricevuto sia attuale, ovvero trasmesso di recente. Questo serve per evitare che gli intrusi attivi possano inviare messaggi vecchi spacciandoli per nuovi.

# 8.2 Algoritmi a chiave simmetrica

Gli **algoritmi a chiave simmetrica** utilizzano la stessa chiave per cifrare e decifrare i messaggi. In particolare, i **cifrari a blocco** prendono un blocco di *n* bit dal testo in chiaro e li trasformano utilizzando una chiave di *n* bit cifrati.

## DES (*Data Encryption Standard*)

Il **DES** è un cifrario prodotto e sviluppato da IBM che venne adottato come standard per le informazioni non riservate nel gennaio 1977 dal governo degli Stati Uniti.
Il testo in chiaro viene cifrato in blocchi da 64 bit, che generano 64 bit di testo cifrato. L'algoritmo è parametrizzato da una chiave a 56 bit e ha 19 stadi distinti. Il primo stadio è indipendente dalla chiave e consiste nella trasposizione dei 64 bit del testo in chiaro. L'ultimo stadio è l'inverso del primo. Il penultimo stadio consiste nello scambiare i 32 bit più a sinistra con i 32 bit più a destra. I rimanenti 16 stadi sono funzionalmente identici, ma sono parametrizzati da diverse funzioni chiave. L'algoritmo è stato progettato per permettere la decifrazione con la stessa chiave usata per la cifratura, quindi i passi vengono semplicemente percorsi in ordine inverso nei due casi. Una delle tecniche utilizzate per rafforzare il DES si chiama **sbiancamento** (*whitening*), consiste nell'utilizzare l'operatore XOR di una chiave casuale a 64 bit su ogni blocco di dati del testo in chiaro, usare il risultato come input per il DES e infine eseguire lo XOR di una seconda chiave a 64 bit. Lo sbiancamento si rimuove facilmente eseguendo le operazioni inverse.

## DES triplo

Fin dal 1979 IBM si rese conto che la chiave di DES era troppo corta e trovò un modo per incrementarla in modo efficacie utilizzando una cifratura tripla: vengono usati due chiavi e tre stadi. Nel prima stadio il testo in chiaro viene cifrato con DES nel modo solito utilizzando la chiave $K_1$, nel secondo DES viene usato in modalità decifrazione usando la chiave $K_2$ e infine viene fatta un'altra cifratura con la chiave $K_1$.

## AES (*Advanced Encryption Standard*)

Nel gennaio 1997 **NIST** (*National Institute of Standards and Tecnology*) invitò i ricercatori di tutto il mondo ad inviare proposte per un nuovo standard che avrebbe preso il nome di **AES**. Le regole del concorso erano: 1) l'algoritmo doveva essere un cifrario simmetrico a blocchi, 2) la struttura doveva essere interamente pubblica, 3) doveva supportare chiavi di lunghezza di 128, 192 e 256 bit, 4) dovevano essere possibili implementazioni *software* e *hardware* e 5) l'algoritmo doveva essere pubblico o rilasciato senza vincoli di alcun tipo.
Nel novembre 2001 **Rijndael** divenne uno standard del governo degli Stati Uniti.

## Modalità di cifratura

**Modalità ECB (*Electronic Code Book*).** Questo metodo consiste nel prendere il testo in chiaro e suddividerlo in blocchi di 8 *byte* (64 bit) che vengono poi cifrati uno dopo l'altro usando la stessa chiave. L'ultimo pezzo di testo in chiaro, se necessario, viene riempito fino a fargli raggiungere la lunghezza di 64 bit. Per evitare attacchi i blocchi cifrati possono essere collegati in diverse maniere.

**Modalità *cipher block chaining*.** Ogni blocco di testo in chiaro è messo in XOR con il precedente blocco cifrato prima di eseguire la cifratura vera e propria, così facendo a blocchi di testo in chiaro uguali non corrispondono più blocchi di testo cifrato identici e la cifratura non è più costituita da un cifrario a sostituzione monoalfabetica. Per il primo blocco lo XOR viene calcolato con un blocco di dati casuali, detto **IV** (*Inizialization Vector*), che è trasmesso in chiaro insieme al testo cifrato. Questa tecnica ha lo svantaggio di richiedere che un intero blocco di 64 bit di testo cifrato arrivi a destinazione prima che la decifrazione possa cominciare.

**Modalità *cipher feedback*.** Questa tecnica è utilizzata con la cifratura byte per byte. Quando arriva il *byte n* del testo in chiaro l'algoritmo DES (AES) agisce sui 64 (128) bit del registro di *shift* per generare 64 (128) bit di testo cifrato. Il *byte* più a sinistra del testo cifrato viene estratto, si calcola lo XOR con $P_n$ e il risultato è trasmesso sulla linea di comunicazione. Inoltre il registro di *shift* viene spostato a sinistra di 8 bit, quindi $C_{n-8}$ esce dal registro mentre $C_n$ entra nella posizione lasciata vacante da $C_{n-1}$. Anche in questo caso serve un vettore di inizializzazione per iniziare la comunicazione.
Un problema di questa modalità si nota quando un bit di testo cifrato viene accidentalmente invertito durante la trasmissione: gli 8 byte che vengono decifrati mentre nel registro di *shift* si trova il byte corrotto saranno pure loro corrotti.

**Modalità *stream cipher*.** Questo metodo è utilizzato per evitare gli errori che potrebbero insorgere con la modalità precedente. Funziona cifrando un vettore di inizializzazione con una chiave crittografica per ottenere un blocco in uscita, che viene poi cifrato per produrre un secondo blocco e così via. La sequenza di blocchi cifrati in uscita, chiamata ***keystream***, viene utilizzata come un blocco monouso e applicata in XOR sul testo in chiaro per ottenere il testo cifrato. Notiamo che il *keystream* è indipendente dai dati, così che può essere calcolato in anticipo, se necessario, ed è anche immune da errori di trasmissione.
È necessario che in questa modalità non venga mai riutilizzata la coppia (chiave, IV), perché questo vorrebbe dire generare più volte lo stesso *keystream* e esporre il testo cifrato ad attacchi di tipop *keystream* riutilizzato.

**Modalità contatore.** Questa modalità risolve un problema presente in tutte le modalità eccetto la ECB: l'impossibilità di accesso casuale ai dati. In questo caso il testo in chiaro non viene cifrato direttamente, ma si cifra invece un vettore d'inizializzazione con una costante e il risultato è messo in XOR con il testo in chiaro. Incrementando di 1 il vettore d'inizializzazione ad ogni blocco diventa facile riuscire a decifrare un blocco in qualunque posizione si trovi, senza dover decifrare

prima tutti i suoi predecessori.
Risulta necessario scegliere vettore d'inizializzazione è chiave indipendentemente e in maniera casuale, in modo che anche se la stessa chiave viene usata due volte accidentalmente quando l'IV è differente nei due casi il testo in chiaro risulta salvo.

### Criptoanalisi

Il primo sviluppo della criptoanalisi è la **criptoanalisi differenziale**, una tecnica che può essere utilizzata per attaccare i cifrari a blocco con attacchi di tipo probabilistico. Il secondo sviluppo è la **criptoanalisi lineare**, utilizzata per ridurre il fattore lavoro necessario a forzare il cifrario. Il terzo sviluppo consiste nell'analizzare il consumo dell'alimentazione elettrica per trovare le chiavi segrete, questo tipo di criptoanalisi può essere sconfitta solo con una programmazione attenta a livello di *assembly*. Il quarto sviluppo è l'analisi dei tempi che, utilizzata simultaneamente a quella del consumo possono semplificare il lavoro di decriptazione delle chiavi.

## 8.3 ALGORITMI A CHIAVE PUBBLICA

Nel 1976 due ricercatori proposero un sistema crittografico radicalmente innovativo, in cui le chiavi di cifratura e decifrazione erano differenti e la chiave di decifrazione non era facilmente derivabile da quella di cifratura. La crittogradia a chiave pubblica richiede che ciascun utente abbia due chiavi: una chiave pubblica usata per cifrare i messaggi da mandare a quell'utente e una chiave privata che l'utente usa per decifrare i messaggi.

### RSA

RSA si basa su alcuni principi della teoria dei numeri. Vengono scelti due numeri primi, $p$ e $q$, tipicamente di 1.024 bit e viene calcolato $n = pq$ e $z = (p - 1)(q - 1)$. Viene scelto, poi, un numero primo relativamente a $z$, detto $d$ e calcolato $ed = 1 \bmod z$. La sicurezza di questo metodo si basa sulla difficoltà di scomporre in fattori i numeri molto grandi, infatti il suo svantaggio è che richiede chiavi di almeno 1.024 bit per poter offrire una buona sicurezza.

### Altri algoritmi a chiave pubblica

Esistono anche degli altri schemi a chiave pubblica, come quelli basati sulle curve ellittiche, ma le due categorie principali si basano sulla difficoltà di fattorizzare i numeri primi e sul calcolo dei logaritmi discreti modulo un grande numero primo.

## 8.5 Firme digitali

### Firme a chiave pubblica

Ad un certo punto risulta necessario un metodo per firmare i documenti che non richieda l'autorità fidata. La crittografia a chiave pubblica per la firma digitale è uno schema elegante, che però presenta alcuni problemi legati principalmente al contesto in cui viene utilizzata piuttosto che agli algoritmi utilizzati. Nel 1991 NIST propose una variante dell'algoritmo a chiave pubblica per il nuovo **DSS** (*Digital Signature Standard*) che si basa sulla difficoltà di calcolare i logaritmi discreti.

### *Message digest*

Spesso ai metodi di firma si rivolge la critica di riunire due funzioni distinte: l'autenticazione e la segretezza. Questo schema consiste nel costruire su una funzione di tipo *hash* monodirezionale, che prende come input un testo chiaro di lunghezza arbitraria e calcola una stringa di bit di lunghezza fissa. La funziona *hash* detta ***message digest*** è molto più veloce da calcolare che non cifrare il testo stesso con un algoritmo a chiave pubblica, per questo motivo i *message diget* possono essere usati per velocizzare gli algoritmi di firma digitale.

**MD5.** Questo *message digest* mescola i bit di ingresso in una maniera sufficientemente complicata da ottenere che ogni bit in uscita sia influenzato da tutti i bit d'ingresso. Sono state trovate alcune vulnerabilità manon è ancora stato forzato.

**SHA-1 (*Secure Hash Algorithm*).** Come MD5 anche questa funzione elabora gli input in blocchi di 512 bit, però al contrario genera dei *message digest* di 160 bit. SHA-1 è ampiamente usato per trasmettere i messaggi dove l'integrità è importante ma i contenuti non sono segreti: con un costo di calcolo relativamente basso questo schema riesce a garantire una probabilità molto alta di rilevare le modifiche al testo in transito.

### L'attacco del compleanno

L'**attacco del compleanno** è un approccio che spiega come per forzare un *message digest* di *m* bit possono bastare $2^{m/2}$ operazioni.

## 8.5 GESTIONE DELLE CHIAVI PUBBLICHE

La crittografia a chiave pubblica consente la comunicazione sicura fra persone che non condividono una chiave comune e permette la firma dei messaggi senza la presenza di una terza parte fidata. I *message digest* firmati, infine, abilitano e facilitano la verifica dell'integrità dei messaggi.

## Certificati

La funzione principale di un **certificato** è quella di legare una chiave pubblica con un protagonista, oppure ad un particolare attributo. Un'organizzazione che certifica le chiavi pubbliche è chiamata una **CA** (*Certificate Authority*): la CA emette un certificato, ne calcola l'*hash* SHA-1 e lo firma con una chiave privata di CA.

## X.509

Se tutte le persone interessate alla firma elettronica andassera dalla CA con un tipo diverso di certificato, l'operazione di gestire i diversi formati diventerebbe presto proibitiva. Per risolvere questo problema è stato sviluppato uno standard per i certificati, poi approvato dalla ITU, chiamato **X.509**. La funzione principale di questo standard è quella di descrivere i certificati, che sono codificati con OSI **ASN.1** (*Abstract Syntax Notation*), a partire dalla V3 è permesso utilizzare i nomi DNS al posto di quelli X.500.

## Infrastrutture a chiave pubblica

Affidarsi ad una sola CA per emettere tutti i certificati è una soluzione che non può funzionare, e anche avere diverse CA tutte gestite dalla stessa organizzazione e tutte con la stessa chiave privata usata per firmare i certificati sarebbe un problema per la sicurezza. Per questo motivo si è evoluto un metodo differente per certificare le chiavi pubbliche: **PKI** (*Public Key Infrastructure*). La funzione della PKI è quella di fornire la struttura per tutti i componenti del processo e definire gli standard per i diversi tipi di documenti e protocolli.
La CA di livello più alto, la *root*, certifica le CA di secondo livello, **RA** (*Regional Authorities*), che a loro volta certificano le vere e proprie CA, che emettono i certificati X.509 alle organizzazioni e ai privati. Quando la *root* autorizza una nuova RA genera un certificato X.509 che ne sancisce l'avvenuta approvazione e che ne include la chiave pubblica della nuova RA, quindi lo firma e lo passa alla RA. Analogamente quando la RA approva una nuova CA produce e firma un certificato che riporta l'avvenuta approvazione e che contiene la chiave pubblica della CA. Una catena di certificati che risale fino alla *root* è detta **catena di fiducia** o **percorso di certificazione**.
Per il problema di gestione della *root* la soluzione consiste nel non avere una singola *root* ma di averne tante, ognuna con le proprie RA e CA: i *browser* moderni contengono già al loro interno le chiavi pubbliche di oltre 100 *root* (***trust anchors***), e permettono agli utenti di controllarle e di cancellare quelle che sembrano sospette.

# 8.6 Sicurezza delle comunicazioni

## IPsec

Il progetto **IPsec** (*IP security*) definisce un'infrastruttura per fornire molteplici servizi, algoritmi e granularità. La ragione per avere molteplici servizi è che non tutti vogliono pagare il prezzo per avere tutti i servizi disponibili in ogni istante, quindi alcuni di questi vengono resi disponibili su richiesta. I servizi principali sono: segretezza, integrità dei dati e protezione dagli attacchi di tipo

ripetizione. Tutti questi servizi sono basati sulla crittografia a chiave simmetrica. Avengo più algoritmi, invece, rendiamo IPsec indipendente dall'algoritmo in sé e quindi l'infrastruttura può sopravvivere anche se un particolare algoritmo viene forzato. Prevedendo diverse granularità, infine, è possibile proteggere una singola connessione TCP come tutto il traffico fra due *host*, tutto il traffico fra due *router* sicuri o altre possibilità.
IPsec è orientato alla connessione, anche se si trova sullo strato IP. Una connessione nel contesto IPsec è detta **SA** (*Security Association*), ed è una connessione *simplex* tra due estremi a cui è associato di un identificatore di sicurezza. Se è necessario stabilire un traffico sicuro nelle due direzioni saranno necessarie sue SA. Gli identificatori di sicurezza sono trasportati da pacchetti che viaggiano su queste connessioni sicure e vengono usate all'arrivo dei pacchetti sicuri per la ricerca delle chiavi e di altre informazioni rilevanti.
IPsec può essere utilizzato in due modalità. Nella **modalità trasporto** l'intestazione IPsec viene inserita subito dopo quella dell'IP e il campo *Protocol* dell'intestazione IP è cambiato in modo da indicare che l'intestazione IPsec segue quella solita dell'IP. L'intestazione IPsec contiene le informazioni di sicurezza: l'indicatore SA, un nuovo numero di sequenza ed eventualmente un controllo di integrità del campo *payload*. Nella **modalità tunnel** l'intero pacchetto IP, intestazione compresa, viene incapsulato nel corpo di un nuovo pacchetto IP con un'intestazione IP completamente diversa. Questa modalità è utile quando il tunnel arriva in un posto diverso dalla sua destinazione finale oppure quando si vuole aggregare un'insieme di connessioni TCP per poi gestirle come un unico flusso cifrato. Lo studio della distribuzione del flusso dei pacchetti, anche se cifrati, è chiamato analisi del traffico. Lo svantaggio della modalità tunnel è dato dal fatto che aggiunge delle ulteriori intestazioni IP e quindi aumenta in modo sostanziale la dimensione del pacchetto, al contrario della modalità trasporto.
La prima ulteriore intestazione è detta **AH** (*Authentication Header*) e garantisce il controllo dell'integrità e la sicurezza contro gli attacchi di ricezione, ma non la segretezza in quanto non ha cifratura. Il campo *Next header field* serve per memorizzare il valore originale che aveva il campo IP *Protocol* prima di essere rimpiazzato, il campo *Payload lenght* contiene il numero di *word* a 32 bit nell'intestazione AH meno 2, il campo *Security parameter index* è l'identificatore di connessione, il campo *Sequence number* viene usato per attribuire un numero a tutti i pacchetti inviati in un SA e infine *Authentication data* che è un campo a lunghezza variabile che contiene la firma digitale del *payload*. Uno schema di questo tipo è chiamato **HMAC** (*Hashed Message Authentication Code*) ed è molto più veloce da calcolare della strategia alternativa, che consiste nell'usare prima SHA-1 per poi passare ad elaborare il risultato con RSA.
L'intestazione IPsec alternativa è **ESP** (*Encapsulating Security Protocol*) e consiste in due *word* a 32 bit che costituiscono i campi *Security parameters index* e *Sequence number*. Una terza *word*, *Inizialization vector*, è usata per la cifratura dei dati. In questo caso **HMAC** può essere calcolato mentre i bit sono trasmessi verso l'interfaccia di rete e poi aggiunto alla fine, mentre con AH il pacchetto va messo in un *buffer* e la firma calcolata prima di poterlo trasmettere.

## *Firewall*

Il ***firewall*** ha due componenti: due *router* che fanno il filtraggio dei pacchetti e un *gateway* applicativo. Esistono anche delle configurazioni più semplici, ma questa struttura ha il vantaggio di obbligare ogni pacchetto a transitare attraverso due filtri e un *gateway* applicativo sia per entrare che per uscire.
Ogni ***packet filter*** è un *router* standard equipaggiato con funzionalità extra che permettono di ispezionare ogni pacchetto in arrivo o in uscita, i pacchetti che rispondono a certi criteri vengono fatti passare normalmente mentre quelli che falliscono il test vengono scartati. I *packet filter* sono gestiti tramite tabelle configurate dall'amministratore di sistema, che elencano le sorgenti e destinazioni accettabili e quelle bloccate e le regole predefinite riguardo a cosa fare con i pacchetti

che vengono o vanno verso altre macchine. La seconda parte del *firewall* è costituita dal ***gateway* applicativo**, che non guarda più i pacchetti in quanto tali ma opera sullo strato applicativo. Anche quando il *firewall* è configurato perfettamente rimangono ancora molti problemi di sicurezza, per esempio gli attacchi che hanno come scopo quello di fermare l'operatività dell'obiettivo e non quello di rubare dei dati, chiamati **DoS** (*Denial of Service*). Una variante più pericolosa è possibile quando un intruso è già penetrato in centinaia di macchine sparse per il mondo e ordina a tutte queste macchine di attaccare simultaneamente un solo obiettivo, attacchi di questo tipo sono detti **DDoS** (*Distributed DoS*).

### Sicurezza *wireless*

Molti dei problemi di sicurezza delle reti *wireless* sono dovuti al fatto che i produttori delle stazioni di trasmissione *wireless*, gli *access point*, vogliono rendere questi prodotti semplici da usare per gli utenti finali. Questo si ottiene al costo di una scarsa sicurezza e del pericolo di vedere i propri dati trasmessi a tutti quelli che sitravano nel raggio di portata dell'apparecchio.

**Sicurezza di 802.11.** Lo standard 802.11 prescrive un protocollo di sicurezza per lo strato *data link* chiamato **WEP** (*Wired Equivalent Privacy*), che è stato progettato per portare sicurezza in una rete LAN *wireless* allo stesso livello di quella delle reti cablate. Quando la sicurezza di 802.11 viene attivata il risultato è che ogni stazione stabilisce una chiave segreta condivisa con la stazione base. La cifratura WEP usa uno stream *cipher* basata sull'algoritmo RC4, utilizzato per generae un *keystream* che viene applicato in XOR al testo in chiaro per produrre testo cifrato. Il *payload* di ogni pacchetto viene cifrato e l'IV utilizzato per inizializzare il *keystream* RC4 viene inviato insieme al testo cifrato. Il destinatario quando riceve il pacchetto per prima cosa estrae il *payload* cifrato, poi genera il *keystream* a partire dalla chiave segreta condivisa e dall'IV che ha ricevuto, quindi calcola lo XOR fra il testo cifrato e il *keystream*. In questo modo ottiene il testo in chiaro e può procedere a verificarne il *checksum*.

**Sicurezza *Bluetooth*.** *Bluetooth* ha tre modalità di sicurezza, che vanno da nessuna sicurezza a una completa cifratura dei dati e controllo dell'integrità. Quando un nuovo dipositivo *slave* chiede un canale al *master* si suppone che i due dispositivi abbiano già effettuato uno stambio di chiavi segrete condivise (***passkey***). Per stabilire la connessione *master* e *slave* effettuano il controllo per vedere se l'altro conosce la *passkey* e, in caso affermativo, negoziano se il canale dev'essere cifrato e se bisogna effettuare il controllo d'integrità o entrambe le cose. Quindi scelgono una chiave di sessione casuale a 128 bit, di cui alcuni bit possono essere resi pubblici. La cifratura utilizza uno *stream cipher* detto $\mathbf{E_0}$, il controllo d'integrità usa **SAFER+**.

**Sicurezza del WAP 2.0.** Visto che WAP 2.0 è basato su IP supporta in pieno l'uso di **IPsec** nello strato *network*, mentre nello strato trasporto le connessioni TCP si possono proteggere con **TLS**. Ad un livello più alto WAP 2.0 usa la connessione *client* **http**. Librerie crittografiche allo strato applicativo forniscono il controllo d'integrità e di non-ripudio.

## 8.7 Protocolli di autenticazione

L'**autenticazione** è la tecnica usata dai processi per verificare che la loro controparte nella comunicazione sia veramente chi dice di essere.

Il modello generale utilizzato da tutti i protocolli di autenticazione comincia inviando un messaggio ad un **KDC** (*Key Distribution Center*) e continua con uno scambio di messaggi in varie direzioni. In molti protocolli viene stabilita tra i comunicanti una chiave segreta detta ***session key***, da usare per continuare la conversazione. Nella realtà per motivi di prestazioni tutti il traffico è cifrato con la crittografia a chiave simmestrica, mentre la crittografia a chiave pubblica è usata largamente nei protocolli di autenticazione per stabilire le chiavi di sessione.

## Come stabilire una chiave condivisa: lo scambio di chiave di *Diffie-Hellman*

Il protocollo che permette agli sconosciuti di scambiarsi una chiave segreta è chiamato **scambio di chiave di *Diffie-Hellman***. Questo protocollo richiede che i due estremi si mettano d'accordo su due numeri grandi $n$ e $g$ tali che $n$ è un numero primo, $(n - 1)/2$ è pure primo e $g$ soddisfa certe particolari condizioni. Questi numeri possono essere pubblici. Il primo estremo invia all'altro un messaggio che contiene $(n, g, g^x \bmod n)$ e l'altro risponde inviando un messaggio che contiene $g^y \bmod n$. Il primo estremo a questo punto calcola $(g^y \bmod n)^x \bmod n$ e il secondo $(g^x \bmod n)^y \bmod n$. Per le regole dell'aritmetica modulare entrambe le espressioni valgono $g^{xy} \bmod n$, che diventerà quindi la chiave segreta condivisa.
Questo algoritmo è vulnerabile ad attacchi di tipo ***bucket brigade***, detti anche ***man-in-the-middle***.