Ottimizzazione Combinatoria
Appunti

Giovanni "Qua' Qua' dancer" Palma e Alex Basta

Contents

Chapter 1

Introduzione

Soccia che bononia

Chapter 2

Introduzione alle Reti di Calcolatori

2.1 Cos'è una Rete di Calcolatori?

Definition 2.1.1: Rete di Calcolatori

Un insieme di dispositivi di calcolo, autonomi e interconnessi, in grado di scambiare informazioni. L'autonomia è una caratteristica cruciale: ogni dispositivo, o *host*, deve essere capace di svolgere compiti di calcolo e comunicazione in modo indipendente, senza un controllo centralizzato esterno per ogni singola operazione. Ad esempio, i terminali di una rete telefonica tradizionale non sono autonomi, e quindi non costituiscono una rete di calcolatori.

Le reti nascono per soddisfare esigenze fondamentali, evolutesi nel tempo:

- **Supporto alla comunicazione tra utenti**: La motivazione originaria. Ha portato allo sviluppo di servizi rivoluzionari come la posta elettronica (e-mail) e il World Wide Web (WWW).
- **Condivisione di informazioni**: Accesso a database, pagine web e file remoti, rendendo l'informazione distribuita e universalmente accessibile.
- **Condivisione di risorse e dispositivi**: Permette a più utenti di utilizzare dispositivi costosi o centralizzati, come stampanti ad alte prestazioni o grandi sistemi di archiviazione (storage), ottimizzandone l'uso.
- **Accesso a calcolatori remoti**: Possibilità di utilizzare la potenza di calcolo di un computer senza essere fisicamente presenti.
- **Calcolo distribuito e sistemi scalabili**: Le reti consentono di suddividere calcoli complessi tra più macchine, eseguendoli in parallelo e aumentando enormemente le prestazioni. Questo rende possibile la creazione di sistemi scalabili.

Note:

Tutte queste reti interconnesse tra loro formano quella che oggi conosciamo come Internet, la "RETE delle RETI", un agglomerato globale di reti eterogenee che comunicano grazie a protocolli standard.

Definition 2.1.2: Sistema Scalabile

Un sistema è definito **scalabile** quando i suoi tempi di risposta rimangono accettabili anche a fronte di un aumento massiccio della dimensione del problema o del numero di utenti. Internet è l'esempio per eccellenza di un sistema scalabile.

2.2 Classificazione delle Reti

Le reti di calcolatori possono essere classificate in base alla loro estensione geografica, ovvero l'area coperta dai dispositivi connessi.

- **PAN (Personal Area Network)**: Reti personali che connettono dispositivi molto vicini, tipicamente entro una stanza o su una scrivania. Sono gestite dal singolo utente.

  Example 2.2.1 (Esempi di PAN)
  Connessioni Bluetooth tra uno smartphone, auricolari wireless e uno smartwatch.

- **LAN (Local Area Network)**: Reti locali che coprono un'area limitata come un ufficio, un edificio o un campus universitario (raggio di circa 100-200 metri). Sono generalmente gestite da un'unica organizzazione (azienda, università).

- **MAN (Metropolitan Area Network)**: Reti metropolitane che si estendono su un'intera area urbana, con un raggio di decine di chilometri. Sono gestite da provider di servizi di comunicazione.

  Example 2.2.2 (Esempio di MAN)
  La rete ALMAWIFI dell'Università di Bologna, che collega diversi edifici sparsi per la città.

- **WAN (Wide Area Network)**: Reti geografiche che coprono aree molto vaste, come nazioni o interi continenti. Possono utilizzare tecnologie complesse e integrate, incluse le comunicazioni satellitari.

- **Internet**: È la rete globale, un'interconnessione di innumerevoli PAN, LAN, MAN e WAN. La sua esistenza è resa possibile dall'adozione di un insieme comune di regole di comunicazione: i protocolli di Internet.

2.3 Evoluzione e Costi di Internet

Lo sviluppo di Internet è stato un processo incrementale, reso possibile dalla distribuzione dei costi di realizzazione e gestione tra innumerevoli entità.

- **1969**: La prima rete progenitrice di Internet (ARPANET) nasce come un esperimento connettendo solo 4 calcolatori di 4 università americane. L'idea era trasmettere bit di informazione su linee telefoniche usando un dispositivo intermedio, il **MODEM** (Modulatore-Demodulatore), che converte i segnali digitali del computer in segnali analogici per la linea telefonica e viceversa.

- **Inizio 2003**: Internet conta già oltre 172 milioni di computer connessi.

- **Oggi (2020)**: Si stimano oltre 4 miliardi di dispositivi con indirizzo IP e oltre 25 miliardi includendo i dispositivi dell'Internet of Things (IoT).

- **Previsione 2022-2025**: Si stima che si raggiungeranno oltre 60 miliardi di dispositivi connessi.

Note:
Il grafico mostra una crescita esponenziale del numero di host (dispositivi con nome logico nei servizi DNS), che tende a stabilizzarsi intorno al 2017. In realtà, questa metrica non cattura l'esplosione dei dispositivi IoT, che continua a far crescere esponenzialmente il numero totale di nodi connessi.

La gestione dei costi è distribuita su diverse scale:

- **Scala Globale**: Le grandi infrastrutture (WAN, dorsali oceaniche) sono finanziate e gestite da consorzi multinazionali e grandi provider di servizi, che sostengono costi elevatissimi.

- **Scala Locale**: La maggior parte dell'infrastruttura capillare è costituita da piccole reti (LAN) gestite da singole aziende, enti o privati, con costi limitati e distribuiti.

- **Costo per l'utente**: L'utente finale paga per l'accesso alla rete, con tariffe che possono essere basate sul tempo di connessione, sulla quantità di dati scambiati ("a consumo") o forfettarie ("flat" o "tutto incluso").

2.4 Prestazioni di una Rete

Per un utente, le prestazioni di una rete sono definite principalmente da due indici fondamentali.

- **Capacità di Trasmissione (Bandwidth)**: Comunemente ma impropriamente chiamata "velocità della rete", indica la quantità di dati (misurata in bit o Byte, dove 1 Byte = 8 bit) che possono essere trasmessi in un secondo. Si misura in bit/sec o byte/sec, con i relativi multipli (Kilo, Mega, Giga, Tera).

  Note:
  Pensando a una rete come a un tubo, la capacità di trasmissione è il diametro del tubo: più è largo, più dati possono passare contemporaneamente.

- **Ritardo di Collegamento (Latency)**: Indica il tempo necessario affinché un dato transiti dal mittente al destinatario. È determinato non solo dalla distanza fisica (la velocità della luce è un limite invalicabile), ma anche dai tempi di elaborazione e gestione che i dati subiscono nei nodi intermedi della rete (router, switch), a causa dei protocolli di comunicazione.

  Note:
  Continuando l'analogia, il ritardo è il tempo che i dati impiegano per percorrere l'intera lunghezza del tubo.

Example 2.4.1 (Capacità vs. Ritardo)
Immaginiamo di dover trasferire una grande quantità di dati (es. 1 Terabyte) da Milano a Roma.

- **Opzione A (Rete)**: Utilizzare una connessione in fibra ottica con capacità di 1 Gbit/s.
- **Opzione B (Furgone)**: Caricare i dati su un hard disk e trasportarlo con un furgone.

Il furgone ha una "capacità di trasmissione" enorme (trasporta 1 TB in un colpo solo), ma ha un ritardo molto alto (diverse ore per il viaggio). La fibra ottica ha una capacità inferiore ma un ritardo quasi istantaneo. La scelta migliore dipende dall'esigenza specifica: per applicazioni interattive (come il gaming online), un basso ritardo è cruciale.

Esistono anche indici di prestazione più complessi e specifici per determinate applicazioni:

- **Jitter**: È la variazione statistica del ritardo di rete. Un jitter elevato significa che i pacchetti di dati arrivano a intervalli irregolari. È un parametro critico per le applicazioni di streaming (audio/video), dove un flusso costante è essenziale per evitare interruzioni o "scatti". Per compensare il jitter, i dispositivi usano un *buffer di ricezione*, una memoria temporanea che immagazzina i pacchetti in arrivo e li riproduce a un ritmo costante.
- **RTT (Round Trip Time)**: È il tempo totale di andata e ritorno, ovvero il tempo che intercorre tra l'invio di un pacchetto e la ricezione della relativa risposta. È fondamentale nelle applicazioni interattive come i giochi online.

Example 2.4.2 (Importanza dell'RTT nel Gaming)
Due giocatori, A e B, si sfidano online.

- Il giocatore A ha un RTT di 10 ms verso il server di gioco.
- Il giocatore B ha un RTT di 25 ms.

Quando entrambi si vedono sullo schermo nello stesso momento e sparano, l'azione del giocatore A raggiunge il server 15 ms prima di quella del giocatore B. Di conseguenza, il server registrerà che A ha sparato per primo, e B "muore prima", anche se dal suo punto di vista ha sparato simultaneamente. Questo dimostra come una bassa latenza sia più importante della capacità di trasmissione in questo contesto.

2.5 Componenti per la Connessione di Rete

Per connettere un calcolatore a una rete sono necessari componenti hardware e software specifici.

- **Dispositivo o Scheda di Rete (Hardware)**: È il componente fisico (spesso integrato nella scheda madre) che si occupa di codificare i dati digitali del computer in segnali trasmissibili sul mezzo fisico (in trasmissione) e di decodificare i segnali ricevuti in dati digitali (in ricezione).
- **Mezzo di Trasmissione**: È il supporto fisico attraverso cui si propagano i segnali (es. cavi di rame, fibre ottiche, onde radio). Realizza l'infrastruttura fisica della rete.
- **Connettore di Rete**: È l'interfaccia standard (es. la porta RJ45 per i cavi Ethernet) che collega fisicamente la scheda di rete al mezzo di trasmissione.
- **Protocolli di Rete (Software)**: Sono un insieme di regole e procedure, implementate a livello software (spesso nel sistema operativo), che gestiscono ogni aspetto della comunicazione per garantirne il corretto funzionamento e la compatibilità tra dispositivi diversi.
- **Driver (Software)**: È un software di basso livello che permette al sistema operativo di comunicare e utilizzare le funzionalità hardware della scheda di rete. Se una scheda non funziona, spesso è a causa di driver mancanti o non corretti.

Note:

La standardizzazione di connettori, schede e protocolli è ciò che garantisce la compatibilità e l'interoperabilità, permettendo a dispositivi di produttori diversi di comunicare senza problemi.

2.6 Collegamenti e Infrastrutture di Rete

Definition 2.6.1: Infrastruttura di Rete

L'infrastruttura di rete descrive la struttura complessiva dei collegamenti fisici tra tutti i dispositivi (host) di una rete. La comunicazione tra due host è possibile se esiste un collegamento diretto o una sequenza di collegamenti intermedi, detta **cammino**.

Esistono diversi schemi di connessione (infrastrutture):

- **Punto a Punto**: La connessione più semplice, collega direttamente solo due host. È facile da gestire e poco costosa. Esempi comuni sono le connessioni Bluetooth o il collegamento di un modem domestico al provider.
- **Completamente Connessa**: Ogni host è collegato direttamente a tutti gli altri host della rete.
  - **Vantaggi**: Altissima **resilienza** (se un collegamento si rompe, ne esistono molti altri alternativi), possibilità di **parallelismo** (più comunicazioni possono avvenire simultaneamente su cavi diversi, senza collisioni) e prestazioni elevate.
  - **Svantaggi**: Estremamente **costosa** e complessa da cablare. Il numero di collegamenti cresce esponenzialmente con il numero di host.
- **Parzialmente Connessa (o Minimamente Connessa)**: Esiste almeno un cammino tra ogni coppia di host, ma non tutti sono collegati direttamente. È un compromesso tra costo e connettività.
  - **Vantaggi**: Costo e complessità ridotti rispetto a una rete completamente connessa.
  - **Svantaggi**: Meno resiliente. Il guasto di un singolo collegamento critico può portare a una **partizione di rete**.
- **Partizione di Rete**: Si verifica quando, a seguito di un guasto, la rete si divide in due o più sottogruppi di host isolati, incapaci di comunicare tra loro.

Note:

Per comunicare, è necessario identificare univocamente ogni scheda di rete. A questo scopo, ogni scheda possiede un indirizzo fisico unico al mondo, chiamato indirizzo **MAC (Media Access Control)**, lungo 48 bit (6 byte).

2.7 Topologie di Rete

Definition 2.7.1: Topologia di Rete

La topologia descrive lo schema geometrico delle connessioni in una rete. Le topologie standard rappresentano un buon compromesso tra costo, complessità e benefici, e sono tipicamente usate in reti piccole (LAN e PAN).

Le principali topologie sono:

- **Anello (Ring)**: Ogni host è connesso solo ai due vicini, formando un anello chiuso. I dati viaggiano in una direzione (oraria o antioraria). È più robusta di una rete minimamente connessa: se un collegamento si interrompe, i dati possono viaggiare nel verso opposto per raggiungere la destinazione, mantenendo la rete connessa.
- **Stella (Star)**: Tutti gli host sono collegati a un dispositivo centrale (un **hub** o uno **switch**). È la topologia più utilizzata oggi nelle LAN.
- **Bus**: Tutti gli host sono collegati a un unico cavo condiviso, detto bus. È una topologia semplice ed economica, ma la gestione delle collisioni è critica.
- **Albero (Tree)**: È una struttura gerarchica, simile a una stella estesa, dove un nodo centrale si collega ad altri nodi che, a loro volta, possono fungere da centro per altri dispositivi.
- **Maglia (Mesh)**: È una topologia complessa e non strutturata, tipica delle grandi reti (come Internet), dove i nodi hanno connessioni multiple e ridondanti, creando un grafo complesso.

2.7.1 Dettagli sulla Topologia ad Anello e a Stella

Topologia ad Anello e il Token Ring Per evitare collisioni in una topologia ad anello, si usa un protocollo di accesso al mezzo chiamato **Token Ring**.

- Un pacchetto speciale di bit, il **token** (testimone), circola continuamente sull'anello.
- Solo l'host che possiede il token ha il diritto di trasmettere dati.
- Una volta terminata la trasmissione, l'host rilascia il token, passandolo all'host successivo.
- Questo meccanismo **garantisce l'assenza di collisioni**.
- **Svantaggio critico**: Se il token viene perso o danneggiato, nessun host può più trasmettere, paralizzando la rete.

Topologia a Stella: Hub vs. Switch Il dispositivo centrale in una topologia a stella può essere un hub o uno switch, che operano a livelli diversi e hanno comportamenti molto differenti.

- **Hub (Livello 1 - Fisico)**: È un "ripetitore multiporta". Quando riceve un segnale su una porta, semplicemente lo rigenera e lo ritrasmette **a tutte le altre porte**. Tutti gli host connessi a un hub condividono lo stesso **dominio di collisione**, il che significa che se due host trasmettono contemporaneamente, si verifica una collisione.
- **Switch (Livello 2 - MAC/LLC)**: È un dispositivo più intelligente. Analizza i pacchetti di dati in arrivo, legge l'indirizzo MAC di destinazione e inoltra il pacchetto **solo sulla porta a cui è collegato il destinatario**.

  - **Vantaggi**: Riduce drasticamente le collisioni (ogni porta è un dominio di collisione separato), aumenta le prestazioni permettendo comunicazioni multiple e simultanee (es. A può parlare con B mentre C parla con D), ed evita di congestionare la rete con traffico inutile.
  - **Costo**: È più costoso di un hub, ma offre prestazioni notevolmente superiori.
- **Criticità**: In entrambi i casi, se il dispositivo centrale (hub o switch) si guasta, l'intera rete collegata ad esso smette di funzionare.

2.8 Il Mezzo Fisico di Trasmissione

Il mezzo fisico è il canale attraverso cui viaggiano i segnali. Ne esistono tre tipi principali.

1. **Cavi Metallici (es. rame)**: Trasmettono segnali elettrici (variazioni di tensione e corrente).
   - **Tipi**: Doppino intrecciato (cavi Ethernet), cavo coassiale.
   - **Caratteristiche**: È il metodo più diffuso per il buon rapporto costo/prestazioni.
   - **Problemi**: **Attenuazione** (il segnale perde energia con la distanza) e **interferenza** da rumore elettromagnetico esterno.
2. **Fibre Ottiche**: Trasmettono segnali luminosi (fotoni) vincolati all'interno di sottilissime fibre di vetro.
   - **Caratteristiche**: Offrono le prestazioni migliori in termini di capacità (migliaia di Gbit/s) e sono immuni alle interferenze elettromagnetiche.
   - **Problemi**: Il costo dell'infrastruttura e soprattutto della giunzione (collegamento) delle fibre è molto elevato.
3. **Senza Fili (Wireless)**: Utilizzano la propagazione di onde elettromagnetiche nello spazio.
   - **Tipi**: Onde radio (Wi-Fi, Bluetooth), infrarossi.
   - **Caratteristiche**: Permettono la mobilità dei dispositivi e riducono la necessità di cablaggio.
   - **Problemi**: Il collegamento non è sempre affidabile, la capacità è generalmente inferiore a quella cablata, la distanza di copertura è limitata e il segnale si attenua fortemente con la distanza (l'energia necessaria per raddoppiare la distanza di comunicazione quadruplica).

2.9 Il Dispositivo di Rete (Scheda di Rete)

Definition 2.9.1: Scheda di Rete

È il componente hardware che funge da intermediario tra il calcolatore e il mezzo di trasmissione. Le sue funzioni principali sono memorizzare temporaneamente i dati, codificarli in segnali per la trasmissione, decodificare i segnali in ricezione e gestire l'accesso al mezzo fisico.

Le sue caratteristiche principali sono:

- Prende il nome dal protocollo standard che utilizza (es. scheda **Ethernet**, scheda **Wi-Fi**).
- Dipende strettamente dal mezzo di trasmissione a cui deve collegarsi.
- Possiede un indirizzo fisico univoco a livello mondiale, non modificabile, detto **indirizzo MAC (Medium Access Control)**, assegnato dal costruttore.

2.10 Canali di Comunicazione della Rete

Definition 2.10.1: Canale di Comunicazione

È una visione astratta del mezzo di trasmissione, che può essere considerato come un insieme di "tubi virtuali" separati. Un singolo mezzo fisico può supportare più canali di comunicazione contemporaneamente (ad esempio, usando frequenze diverse per le onde radio).

Esistono due tipi fondamentali di canali:

- **Canale Punto a Punto**: Un canale riservato e utilizzato esclusivamente da due dispositivi, un mittente e un destinatario. Il problema dell'arbitraggio è banale (es. "prima parlo io, poi parli tu").
- **Canale ad Accesso Multiplo (o Broadcast)**: Un canale condiviso su cui tutti i dispositivi connessi possono trasmettere e ricevere. È il tipo di canale più comune nelle reti locali.

I canali ad accesso multiplo introducono due problemi critici che devono essere risolti dai protocolli:

1. **Rischio di Collisione**: Se due o più dispositivi trasmettono contemporaneamente sullo stesso canale, i loro segnali si sovrappongono e si "distruggono" a vicenda, rendendo l'informazione illeggibile.
2. **Arbitraggio**: È necessario definire regole precise (protocolli) per decidere **chi** può trasmettere e **quando**, al fine di evitare o gestire le collisioni.
3. **Indirizzamento**: Poiché tutti ricevono la trasmissione, ogni messaggio deve contenere l'indirizzo (MAC address) del destinatario specifico, in modo che solo quel dispositivo processi il messaggio e tutti gli altri lo ignorino.

Note:

Esiste un indirizzo MAC speciale, detto **indirizzo di broadcast** (composto da tutti 1), che, se usato come destinazione, indica che il messaggio è rivolto a tutte le schede di rete connesse al canale.

2.11 Reti a Commutazione di Circuito

Una volta stabiliti i canali di comunicazione, si possono realizzare servizi di trasferimento dati su vasta scala secondo due modalità principali: a commutazione di circuito e a commutazione di pacchetto.

Definition 2.11.1: Commutazione di Circuito

Una modalità di comunicazione in cui viene stabilito e riservato un cammino fisico dedicato (un **circuito**) di canali di comunicazione punto a punto, che si estende dal mittente al destinatario, *prima* che la trasmissione dei dati abbia inizio. Questo circuito rimane ad uso esclusivo dei due interlocutori per tutta la durata della comunicazione. L'esempio classico è la rete telefonica tradizionale.

La creazione del circuito avviene attraverso una fase iniziale di *setup*, durante la quale il protocollo negozia e prenota ogni singolo canale lungo il percorso. Una volta stabilito, il circuito si comporta come se ci fosse un collegamento fisico diretto tra mittente e destinatario.

- **Vantaggi**:
  - **Prestazioni Garantite**: Una volta creato il circuito, non c'è rischio di collisioni o di contesa per le risorse.
  - **Ritardo Basso e Costante**: Poiché il percorso è predefinito, i nodi intermedi non devono prendere decisioni di instradamento. I dati fluiscono senza interruzioni, garantendo una latenza minima e un jitter (variazione del ritardo) quasi nullo. Questo la rende ideale per applicazioni in tempo reale come le chiamate vocali.
  - **Semplicità dei Dati**: Non è necessario includere indirizzi di mittente e destinatario in ogni frammento di dato, poiché il percorso è già fissato.

- **Svantaggi**:
  - **Ritardo Iniziale Elevato**: La fase di setup per la prenotazione del canale può richiedere un tempo significativo prima che la comunicazione possa iniziare.
  - **Inefficienza e Spreco di Risorse**: Il canale rimane riservato e pagato per tutta la durata della connessione, anche nei momenti in cui non vengono trasmessi dati. Questo porta a un basso utilizzo delle risorse di rete.
  - **Costo Elevato**: Tipicamente il servizio si paga in base al tempo di connessione, indipendentemente dalla quantità di dati scambiati.

2.12 Reti a Commutazione di Pacchetto

È l'alternativa alla commutazione di circuito e costituisce la base di quasi tutte le reti di calcolatori moderne, inclusa Internet.

Definition 2.12.1: Commutazione di Pacchetto

Una modalità di comunicazione in cui i dati da trasmettere vengono suddivisi in blocchi di dimensioni definite, detti **pacchetti**. Ogni pacchetto è un'entità indipendente, contenente, oltre a una porzione dei dati, informazioni di controllo come l'indirizzo del mittente e del destinatario. I pacchetti vengono poi inviati sulla rete senza che venga riservato un percorso predefinito.

In questo modello, i canali di comunicazione (specialmente quelli ad accesso multiplo) sono condivisi tra molti utenti. I pacchetti di diverse comunicazioni vengono intercalati (*multiplexati*) sullo stesso mezzo fisico, ottimizzando l'uso delle risorse.

- **Vantaggi**:
  - **Utilizzo Efficiente delle Risorse**: Il canale viene utilizzato solo quando c'è effettivamente un pacchetto da trasmettere, permettendo a più utenti di condividerlo simultaneamente e riducendo gli sprechi.
  - **Costo Basato sul Consumo**: L'utente paga in base alla quantità di dati trasmessi, non per il tempo di connessione.
- **Svantaggi**:
  - **Ritardo Maggiore e Variabile (Jitter)**: I pacchetti possono dover attendere nei nodi intermedi (router) se la rete è congestionata, introducendo ritardi non costanti.
  - **Nessuna Garanzia di Consegna**: I pacchetti possono essere persi (ad esempio, a causa della congestione) o arrivare a destinazione in un ordine diverso da quello di invio.

2.13 Servizi di Trasmissione a Pacchetto

Il fatto che la commutazione di pacchetto sia intrinsecamente inaffidabile non significa che la comunicazione debba esserlo. I protocolli di rete possono costruire, al di sopra di questa infrastruttura, due tipi di servizi per l'utente.

2.13.1 Servizi orientati alla connessione (Connection-Oriented)

Definition 2.13.1: Servizio Connection-Oriented

Un servizio di trasmissione che **garantisce** la consegna affidabile e ordinata di tutti i pacchetti dal mittente al destinatario. Simula il comportamento di un circuito dedicato, creando un cosiddetto "circuito virtuale".

Per ottenere questa affidabilità, i protocolli implementano meccanismi sofisticati:

- **Numerazione dei Pacchetti**: Ogni pacchetto viene etichettato con un numero di sequenza.

- **Riordino alla Destinazione**: Il destinatario utilizza una memoria temporanea (*buffer*) per memorizzare i pacchetti ricevuti e riordinarli secondo il loro numero di sequenza, ricostruendo il messaggio originale.
- **Conferma di Ricezione (Acknowledgement - ACK)**: Per ogni pacchetto ricevuto correttamente, il destinatario invia al mittente un piccolo messaggio di conferma, detto ACK.
- **Ritrasmissione**: Il mittente avvia un timer per ogni pacchetto inviato. Se non riceve il corrispondente ACK entro la scadenza del timer (*timeout*), assume che il pacchetto sia andato perso e lo **ritrasmette**.

Note:

Il protocollo per eccellenza che fornisce questo servizio su Internet è il **TCP (Transmission Control Protocol)**.

2.13.2 Servizi non orientati alla connessione (Connectionless)

Definition 2.13.2: Servizio Connectionless

Un servizio di trasmissione che **non offre alcuna garanzia** sulla consegna, sull'ordine o sulla non-duplicazione dei pacchetti. Opera secondo un principio di "massimo sforzo" (*best-effort*): la rete fa del suo meglio per consegnare i pacchetti, ma senza alcuna certezza. È analogo a spedire una serie di lettere con la posta ordinaria.

Questo servizio viene utilizzato da applicazioni che privilegiano la velocità e un basso ritardo rispetto all'affidabilità totale, come lo streaming video o audio e i giochi online, dove la perdita occasionale di un pacchetto è preferibile all'attesa di una sua ritrasmissione.

Note:

Il protocollo che fornisce questo servizio su Internet è l' **UDP (User Datagram Protocol)**.

Definition 2.13.3: Congestione di Rete

La congestione si verifica quando un nodo della rete, tipicamente un router, riceve pacchetti a una velocità superiore a quella con cui può inoltrarli. Le code di attesa (buffer) del router si riempiono, e i pacchetti in arrivo vengono scartati (*dropped*), causando perdita di dati.

Chapter 3

Architetture e Protocolli di Rete

3.1 L'Architettura a Livelli

La gestione della comunicazione in una rete è un problema estremamente complesso. Per dominarlo, si adotta un approccio basato sulla **separazione delle competenze**, organizzando i protocolli in una gerarchia di **livelli** sovrapposti (*stack*). Questa architettura, detta *a livelli*, si basa su alcuni principi fondamentali:

- Ogni livello è progettato per risolvere una specifica classe di problemi della comunicazione.
- Ogni livello fornisce dei **servizi** al livello immediatamente superiore, nascondendo i dettagli di come tali servizi vengono realizzati. Ad esempio, il livello superiore non deve preoccuparsi di come i bit vengono trasformati in segnali elettrici.
- Ogni livello utilizza i servizi offerti dal livello immediatamente inferiore per svolgere le proprie funzioni.
- La comunicazione tra livelli adiacenti avviene attraverso interfacce ben definite.

Example 3.1.1 (Analogia della Comunicazione a Livelli)

Due innamorati, un italiano e una giapponese, devono comunicare superando una serie di vincoli: parlano solo la loro lingua madre, l'unico mezzo di trasmissione è un FAX e l'unica macchina da scrivere disponibile usa l'alfabeto cirillico, adatto per la lingua russa.
Creano un'architettura a 4 livelli:

1. **Livello 4 (Dialogo)**: L'italiano formula la sua dichiarazione in italiano. Il suo unico compito è creare il messaggio.
2. **Livello 3 (Traduzione)**: Il livello dialogo passa il messaggio al livello traduzione, chiedendo il servizio di "traduzione in russo". Questo livello si occupa solo della traduzione linguistica.
3. **Livello 2 (Dattilografia)**: Il testo in russo viene passato al livello dattilografia, che lo scrive usando l'alfabeto cirillico sulla macchina da scrivere.
4. **Livello 1 (FAX)**: Il foglio in cirillico viene passato al livello FAX, che si occupa della trasmissione fisica.

Dal lato della ricevente, avviene il processo inverso: il livello FAX riceve i dati, il livello dattilografia li interpreta, il livello traduzione li converte in giapponese e infine il livello dialogo li presenta all'innamorata. Ad ogni livello, non ci si cura dei problemi risolti dagli altri: l'innamorato al livello 4 non sa nulla di russo, cirillico o FAX; sa solo che la sua dichiarazione arriverà a destinazione nella lingua corretta.

3.2 Lo Standard ISO/OSI RM

Per standardizzare le architetture di rete, l'International Organization for Standardization (ISO) ha definito l' **Open Systems Interconnection Reference Model (ISO/OSI RM)**. È un modello concettuale rigoroso che

organizza i protocolli in sette livelli.

7. **Livello Applicazione**: Fornisce i servizi di rete direttamente alle applicazioni dell'utente (es. browser web, client di posta). Protocolli come HTTP, SMTP operano qui.
6. **Livello Presentazione**: Si occupa della sintassi e della semantica dei dati, gestendo la traduzione tra formati dati diversi per superare eterogeneità (es. conversione tra codifiche di caratteri o gestione dell'ordine dei byte Big/Little Endian).
5. **Livello Sessione**: Stabilisce, gestisce e termina le sessioni di comunicazione tra applicazioni, mantenendo lo stato del dialogo.
4. **Livello Trasporto**: Fornisce servizi di trasferimento dati end-to-end (da processo a processo). È responsabile dell'affidabilità (con protocolli come TCP) o della velocità (con UDP), del controllo di flusso e del controllo della congestione.
3. **Livello Rete**: Gestisce l'instradamento (*routing*) dei pacchetti attraverso la rete di reti (internetwork). Si occupa dell'indirizzamento logico (indirizzi IP) e della frammentazione dei pacchetti, se necessario.
2. **Livello Data Link (Collegamento Dati)**: Suddiviso in MAC e LLC. Si occupa di trasferire dati in modo affidabile su un singolo collegamento fisico (un "salto" o "hop"). Gestisce l'indirizzamento fisico (indirizzi MAC), l'accesso al mezzo (per evitare collisioni) e il controllo degli errori a livello di frame.
1. **Livello Fisico**: Definisce le caratteristiche fisiche della trasmissione: voltaggi, temporizzazioni, tipi di cavi e connettori. Si occupa di codificare i bit in segnali (elettrici, luminosi, radio) e trasmetterli sul mezzo fisico.

3.3 L'Architettura di Internet (TCP/IP Stack)

Sebbene il modello ISO/OSI sia il riferimento teorico, l'architettura di Internet, nota come **TCP/IP Stack**, utilizza un modello semplificato a 5 livelli, che fonde i tre livelli superiori dell'OSI in un unico livello Applicazione.

3.3.1 Incapsulamento (Encapsulation)

Il principio fondamentale della comunicazione a livelli è l'incapsulamento.

Definition 3.3.1: Incapsulamento

In fase di trasmissione, ogni livello prende i dati ricevuti dal livello superiore e vi aggiunge le proprie informazioni di controllo, creando una nuova unità di dati più grande. Queste informazioni aggiuntive sono dette **header** (intestazione) e, a volte, **trailer** (coda). Questo processo è simile a inserire una lettera in una busta, su cui si scrivono gli indirizzi.

Il processo inverso, che avviene sul dispositivo ricevente, è il **decapsulamento**: ogni livello legge ed elabora l'header del proprio livello, lo rimuove e passa i dati rimanenti al livello superiore. I dati assumono nomi diversi a seconda del livello:

- **Segmento**: L'unità di dati al Livello Trasporto (TCP/UDP).
- **Pacchetto (o Datagramma)**: L'unità di dati al Livello Rete (IP).
- **Frame**: L'unità di dati al Livello Data Link (MAC).

Note:

Per identificare l'applicazione specifica a cui sono destinati i dati su un host (es. il browser e non il client di posta), il Livello Trasporto utilizza un identificatore chiamato **Numero di Porta**. La combinazione di un **indirizzo IP** e un **Numero di Porta** costituisce un **Socket di comunicazione**, che identifica univocamente un'applicazione in esecuzione su un host specifico in tutta Internet.

3.4 Dispositivi di Interconnessione di Rete

Le reti sono costruite collegando segmenti e reti diverse tramite dispositivi specializzati che operano a livelli differenti dello stack.

- **Repeater e Hub (Livello 1 - Fisico)**:
  - **Repeater**: Un dispositivo a due porte che riceve un segnale, lo "ripulisce", lo amplifica e lo ritrasmette, estendendo la lunghezza massima di un segmento di rete.
  - **Hub**: Un repeater multiporta, funge da punto di connessione centrale per una topologia a stella. Trasmette ogni pacchetto ricevuto a tutte le altre porte.
- **Bridge e Switch (Livello 2 - Data Link)**:
  - **Bridge**: Connette due segmenti di rete (che possono usare tecnologie MAC diverse, es. Ethernet e Token Ring) per formare una singola LAN. Legge gli indirizzi MAC e può tradurre i frame da un formato all'altro.
  - **Switch**: Un bridge multiporta e più evoluto. Impara quali indirizzi MAC sono raggiungibili su ciascuna delle sue porte e inoltra i frame in modo selettivo solo sulla porta corretta, riducendo il traffico e le collisioni.
- **Router (Livello 3 - Rete)**:
  - È il dispositivo che **connette reti diverse** e indipendenti (es. la rete di casa con Internet).
  - A differenza di switch e bridge che operano con indirizzi fisici (MAC), i router operano con indirizzi logici e gerarchici (IP).
  - La sua funzione principale è l'**instradamento** (*routing*): esamina l'indirizzo IP di destinazione di un pacchetto e, consultando la propria tabella di routing, decide quale sia il percorso migliore per inoltrare il pacchetto verso la sua destinazione finale.

Example 3.4.1 (Esempio di Rete Locale Complessa)

La figura mostra una rete locale aziendale. Un utente sulla rete ad anello **token ring** (MAC A) deve comunicare con un utente su un lontano segmento **ethernet** (MAC B).

1. Il frame token ring di A arriva a un **Bridge (B)**.
2. Il Bridge opera a livello 2: legge l'indirizzo MAC di destinazione, converte il frame dal formato token ring al formato ethernet e lo inoltra sul segmento successivo.
3. Il frame ethernet attraversa un **Hub (H)**, che lo propaga a tutti i suoi segmenti.
4. Arriva a un **Repeater (R)**, che rigenera il segnale per coprire una distanza maggiore.
5. Attraversa un secondo repeater e finalmente raggiunge il segmento bus a cui è collegato il destinatario (MAC B).

In questo percorso, solo il Bridge ha "aperto" il frame per leggerne il contenuto a livello 2; Hub e Repeater hanno lavorato solo a livello 1, manipolando i segnali fisici senza interpretarne i dati.

Chapter 4

Analisi Dettagliata dei Livelli

4.1 Il Livello 2: Accesso al Mezzo (MAC/LLC)

4.1.1 Rendere il Canale Affidabile

Il livello fisico è intrinsecamente inaffidabile: i segnali possono essere corrotti dal rumore. Il compito del livello Data Link è di mascherare questa inaffidabilità e fornire al livello superiore l'illusione di un canale privo di errori, almeno su un singolo collegamento. Un protocollo semplice per ottenere ciò è basato su conferme e ritrasmissioni:

1. Il mittente invia un frame e avvia un timer.
2. Il destinatario riceve il frame. Se il frame è corretto (lo verifica tramite bit di controllo), invia un frame di conferma (ACK) al mittente.
3. Se il frame ricevuto è danneggiato, il destinatario lo scarta e non fa nulla.
4. Se il mittente riceve l'ACK prima dello scadere del timer, la trasmissione è conclusa con successo.
5. Se il timer scade senza che sia arrivato un ACK (perché il frame originale o l'ACK stesso sono andati persi), il mittente assume un fallimento e ritrasmette il frame da capo.

4.1.2 Affidabilità Locale (Livello 2) vs. Globale (Livello 4)

È fondamentale comprendere la portata dell'affidabilità fornita dai diversi livelli.

- **Affidabilità a Livello 2 (MAC/LLC)**: È un'affidabilità *hop-by-hop* (salto per salto). Il meccanismo di ACK e ritrasmissione garantisce che un frame arrivi correttamente **solo al nodo successivo** nel percorso. Se dopo un numero massimo di tentativi (*Max Retry Limit*, es. 8 in Ethernet) la trasmissione fallisce, il livello 2 si "arrende", scarta il frame e notifica il fallimento al livello 3.
- **Affidabilità a Livello 3 (Rete)**: In caso di fallimento notificato dal livello 2, il livello Rete può decidere di tentare un **percorso alternativo**, se ne esiste uno, per aggirare il collegamento problematico. Questa è una decisione di *routing*.
- **Affidabilità a Livello 4 (Trasporto)**: È l'unica vera affidabilità *end-to-end* (da capo a capo). Il protocollo TCP sul mittente originale e sul destinatario finale si assicura che **tutti i segmenti** del messaggio originale arrivino a destinazione, corretti e in ordine, orchestrando le ritrasmissioni attraverso l'intera rete, indipendentemente da quanti salti intermedi falliscano o da quanti percorsi vengano cambiati.

4.1.3 Esempi di Protocolli MAC

I protocolli di accesso al mezzo definiscono le regole per l'arbitraggio.

- **Ethernet (Random Access)**: Molto usato nelle LAN cablate. Si basa sul principio *CSMA/CD (Carrier Sense Multiple Access with Collision Detection)*.

1. **Ascolta prima di parlare**: La scheda di rete trasmette solo se il canale è libero.
2. **Rileva le collisioni**: Se, nonostante l'ascolto, si verifica una collisione, la scheda la rileva, interrompe la trasmissione e attende un tempo casuale prima di riprovare.

- **Wi-Fi (IEEE 802.11)**: Usato nelle LAN wireless. Nelle reti radio è difficile rilevare le collisioni, quindi si usa un approccio di *prevenzione* (*CSMA/CA - Collision Avoidance*), dilazionando le trasmissioni per ridurre la probabilità che avvengano.
- **Token Ring (Accesso Controllato)**: Usato in topologie ad anello. Evita completamente le collisioni garantendo che solo chi possiede il "testimone" (token) possa trasmettere. Questo garantisce prestazioni prevedibili e una qualità del servizio (QoS) controllabile.

Note:

Il teorema dei "Due Generali" nella teoria dei sistemi distribuiti dimostra formalmente che in una rete inaffidabile è impossibile per due entità raggiungere un accordo con certezza al 100% in un tempo finito. Questo evidenzia perché i protocolli di rete si basano su meccanismi di ritrasmissione e timeout, che forniscono un'affidabilità pratica ma non una garanzia matematica assoluta.

Chapter 5

Il Livello Rete: Internetworking

5.1 Reti di Reti e il Problema della Scalabilità

Il livello Data Link (MAC/LLC) permette di creare reti locali (LAN) efficienti, ma la vera potenza di Internet risiede nella sua capacità di interconnettere miliardi di dispositivi appartenenti a reti eterogenee sparse in tutto il mondo. Questa interconnessione, o **internetworking**, è gestita dal **Livello Rete** (Livello 3). L'architettura di Internet è gerarchica: le singole reti locali sono collegate tra loro attraverso dispositivi specializzati chiamati **router**, che a loro volta comunicano tramite linee di comunicazione ad alta velocità, dette **dorsali** (*backbone*). Se Internet fosse una singola, enorme rete locale "piatta", la comunicazione sarebbe impossibile. Ogni dispositivo dovrebbe mantenere una tabella con gli indirizzi MAC di tutti gli altri miliardi di dispositivi nel mondo per sapere dove inoltrare i dati. Un sistema del genere non sarebbe assolutamente **scalabile**. La soluzione è un'astrazione gerarchica gestita al Livello 3:

- Ogni rete locale elegge un **router** come suo rappresentante verso l'esterno.
- I router nascondono la complessità interna delle proprie reti locali, che dall'esterno appaiono come un'unica entità.
- I router si scambiano informazioni solo tra di loro per decidere il percorso migliore per i pacchetti. Per farlo, ogni router mantiene una **tabella di instradamento** (*forwarding table*).

Definition 5.1.1: Forwarding e Routing

Sebbene spesso usati in modo intercambiabile, questi due termini descrivono processi distinti:

- **Forwarding (Inoltro)**: È l'azione meccanica di prendere un pacchetto in arrivo, consultare la tabella di instradamento e inviarlo all'interfaccia di uscita corretta verso il prossimo router (*next hop*).
- **Routing (Instradamento)**: È il processo più complesso e intelligente con cui i router costruiscono e mantengono aggiornate le loro tabelle di instradamento. Questo viene fatto tramite **protocolli di routing** che permettono ai router di scambiarsi informazioni sulla topologia della rete.

5.2 Il Protocollo Internet (IP)

Il protocollo cardine del livello rete su Internet è l'**Internet Protocol (IP)**. I suoi compiti principali sono:

- Fornire uno schema di **indirizzamento globale e gerarchico** (gli indirizzi IP) che identifica univocamente ogni host e la rete a cui appartiene.
- Gestire l'**inoltro (forwarding)** dei pacchetti attraverso la rete di reti, dal mittente al destinatario finale.
- Suddividere i dati in pacchetti e, se necessario, gestirne la **frammentazione** (dividere un pacchetto in pezzi più piccoli se una rete intermedia non può gestirne le dimensioni originali).
- Fornire un servizio di comunicazione **connectionless** e *best-effort*, ovvero senza garanzie di consegna.

Note:

Il protocollo IP esiste in due versioni principali: **IPv4**, che utilizza indirizzi a 32 bit ed è ancora largamente diffuso, e **IPv6**, la nuova versione con indirizzi a 128 bit, introdotta per superare i limiti di IPv4.

5.2.1 Indirizzamento IPv4

Un indirizzo IPv4 è una sequenza di 32 bit, convenzionalmente scritta come quattro numeri decimali (ognuno da 0 a 255) separati da un punto. Esempio: '130.136.25.1'. Ogni indirizzo IP identifica univocamente un'interfaccia di rete (es. una scheda di rete) e si compone di due parti:

1. **Network Number**: Identifica la rete a cui l'host è connesso.
2. **Host Number**: Identifica l'host specifico all'interno di quella rete.

Gli indirizzi IP possono essere assegnati in due modi:

- **IP Statico**: L'indirizzo è assegnato permanentemente a un dispositivo. È una configurazione tipica per i server, che devono essere sempre raggiungibili allo stesso indirizzo.
- **IP Dinamico**: L'indirizzo viene assegnato temporaneamente a un dispositivo ogni volta che si connette alla rete. È la modalità usata per i dispositivi client come laptop e smartphone, che si muovono e si connettono a reti diverse.

5.2.2 Classi di Reti IPv4

L'indirizzamento originale di IPv4 suddivideva lo spazio degli indirizzi in classi, per accomodare reti di dimensioni diverse. La classe è determinata dai primi bit dell'indirizzo.

- **Classe A**: Indirizzi che iniziano con il bit '0'. Usano il primo byte per il *network number* e i restanti tre per l'*host number*. Questo permette di avere poche reti (126 in totale), ma ognuna può contenere un numero enorme di host (oltre 16 milioni). Sono assegnate a grandi provider e organizzazioni.
- **Classe B**: Indirizzi che iniziano con i bit '10'. Usano i primi due byte per il *network number* e i restanti due per l'*host number*. Permettono di avere più reti (circa 16.000) con un numero considerevole di host ciascuna (circa 65.000).
- **Classe C**: Indirizzi che iniziano con i bit '110'. Usano i primi tre byte per il *network number* e l'ultimo per l'*host number*. Permettono di avere moltissime reti (oltre 2 milioni), ma ognuna con un numero limitato di host (254). Sono tipiche per piccole e medie imprese.

5.2.3 Sottoreti (Subnetting)

Le classi di indirizzi si sono rivelate troppo rigide. Una grande organizzazione a cui veniva assegnata una rete di classe B poteva trovarsi a dover gestire 65.000 host in un'unica rete locale, con enormi problemi di traffico e amministrazione. La soluzione è il **subnetting**.

Definition 5.2.1: Subnetting

Una tecnica che permette di suddividere una singola rete IP di grandi dimensioni in più sottoreti (subnetwork) logiche più piccole. Questo si ottiene "prendendo in prestito" alcuni bit dalla parte *host number* dell'indirizzo per creare un nuovo campo, il *subnet number*.

Definition 5.2.2: Netmask

Una sequenza di 32 bit, scritta come un indirizzo IP, che specifica quali bit dell'indirizzo IP appartengono alla rete/sottorete (indicati da '1') e quali all'host (indicati da '0').

Example 5.2.1 (Esempio di Subnetting)

Si consideri una rete di classe B '130.136.0.0'. L'organizzazione decide di suddividerla in 256 sottoreti.

- **Netmask**: Si utilizza la maschera '255.255.255.0'. In binario: '11111111.11111111.11111111.00000000'.
- **Effetto**: I primi due byte ('130.136') identificano la rete principale. Il terzo byte, che originariamente faceva parte dell'host number, ora identifica la sottorete (grazie ai bit '1' della maschera). L'ultimo byte rimane per identificare l'host all'interno della sottorete.
- **Risultato**: Si creano le sottoreti '130.136.1.0', '130.136.2.0', ..., '130.136.255.0'. L'host '130.136.3.48' è l'host numero 48 della sottorete numero 3.

Note:

Ogni host, per poter comunicare, deve essere configurato con tre parametri fondamentali: il proprio indirizzo IP, la netmask della sua sottorete e l'indirizzo IP del suo **default router** (o gateway), ovvero il router a cui inviare tutti i pacchetti destinati a reti esterne.

5.3 Instradamento e Protocolli di Supporto

5.3.1 Forwarding: il Viaggio di un Pacchetto

Il processo di forwarding si basa su decisioni prese hop-by-hop da ogni router.

Example 5.3.1 (Tragitto di un pacchetto)

L'host '140.217.2.10' deve inviare un pacchetto all'host '130.136.2.33'.

1. L'host mittente si accorge che la destinazione non è nella sua sottorete ('140.217.2.0'). Invia quindi il pacchetto al suo default router ('140.217.2.254').
2. Il router '140.217.2.254' lo inoltra al router di livello superiore della sua rete ('140.217.0.254').
3. Il router '140.217.0.254' consulta la sua tabella di forwarding. Per raggiungere la rete '130.136.0.0', la tabella indica di inoltrare il pacchetto al router '190.89.0.254'.
4. Il router '190.89.0.254' fa lo stesso: consulta la sua tabella e inoltra il pacchetto al router successivo, '130.136.0.254'.
5. Il router '130.136.0.254' riconosce che il pacchetto è destinato a una delle sue sottoreti e lo inoltra internamente al router della sottorete corretta ('130.136.2.254').
6. Infine, il router '130.136.2.254' consegna il pacchetto direttamente all'host di destinazione '130.136.2.33' sulla sua rete locale.

5.3.2 Protocollo ICMP

Definition 5.3.1: ICMP (Internet Control Message Protocol)

Un protocollo di supporto del livello rete utilizzato da host e router per scambiare messaggi di controllo e notificare errori. I messaggi ICMP viaggiano all'interno di normali pacchetti IP.

ICMP è la base per notificare problemi come:

- **Rete di destinazione non raggiungibile**: Un router lungo il percorso non sa come raggiungere la rete di destinazione.
- **Host di destinazione non raggiungibile**: L'host è spento o non connesso.

Due utility di diagnostica fondamentali si basano su ICMP:

- **ping**: Invia un messaggio ICMP 'echo request' a un host. Se l'host è raggiungibile, risponde con un 'echo reply'. Permette di verificare la connettività e misurare il tempo di andata e ritorno (**RTT - Round Trip Time**).
- **traceroute** (o 'tracert'): Mostra il percorso (la sequenza di router) che i pacchetti seguono per raggiungere una destinazione. Funziona inviando pacchetti con un **TTL (Time To Live)** progressivamente crescente. Ogni router che scarta un pacchetto perché il suo TTL è scaduto, invia indietro un messaggio ICMP, rivelando così la sua identità.

5.3.3 Protocollo ARP

C'è un problema fondamentale: il livello rete (IP) ragiona in termini di indirizzi IP, ma per trasmettere dati su una rete locale come Ethernet, il livello data link ha bisogno dell'indirizzo fisico **MAC**.

Definition 5.3.2: ARP (Address Resolution Protocol)

È il protocollo che si occupa di "tradurre" un indirizzo IP in un indirizzo MAC all'interno di una rete locale.

Il processo funziona così:

1. L'host A vuole inviare dati all'host B sulla stessa LAN, ma conosce solo l'IP di B.
2. L'host A invia un **ARP request** in *broadcast* sulla LAN, che essenzialmente chiede: "Chi possiede l'indirizzo IP '192.168.1.5'?".
3. Tutti gli host sulla LAN ricevono la richiesta, ma solo l'host B, che possiede quell'IP, risponde.
4. L'host B invia un **ARP reply** direttamente all'host A, dicendo: "Sono io, e il mio indirizzo MAC è '00:1A:2B:3C:4D:5E'".
5. A questo punto, A conosce il MAC di B e può inviargli il frame. A memorizza questa associazione IP-MAC nella sua **tabella ARP** per non dover ripetere la richiesta in futuro.

5.4 Assegnazione degli Indirizzi IP e DHCP

Come ottiene un host il suo indirizzo IP?

- **Manualmente**: Un amministratore di rete configura l'IP staticamente su ogni dispositivo.
- **Automaticamente (DHCP)**: È il metodo più comune.

Definition 5.4.1: DHCP (Dynamic Host Configuration Protocol)

Un protocollo di livello applicazione che permette a un server (il server DHCP) di assegnare automaticamente e dinamicamente un indirizzo IP e altri parametri di configurazione (netmask, default gateway, DNS server) ai client che si connettono alla rete.

Il server DHCP gestisce un **pool** (un insieme) di indirizzi IP disponibili. Quando un client si connette, richiede un indirizzo al server. Il server ne assegna uno dal pool per un periodo di tempo limitato, detto **lease time**. Se il client si disconnette, l'indirizzo torna disponibile nel pool dopo la scadenza del lease e può essere riassegnato a un altro dispositivo. Questo meccanismo permette un uso efficiente degli indirizzi IP disponibili.

5.5 Il Futuro: IPv6

La crescita esponenziale di Internet ha portato all'esaurimento degli indirizzi IPv4 disponibili. Per questo, è stato sviluppato IPv6. Le sue caratteristiche principali sono:

- **Spazio di Indirizzamento Enorme**: Gli indirizzi IPv6 sono lunghi **128 bit**, offrendo un numero quasi inimmaginabile di indirizzi unici (circa 3.4×10^{38}).
- **Header Semplificato**: L'header dei pacchetti IPv6 è stato riprogettato e semplificato. Ad esempio, la **frammentazione** dei pacchetti non è più gestita dai router intermedi ma solo dall'host sorgente, velocizzando l'inoltro.
- **Supporto Nativo per Qualità del Servizio (QoS)**: Permette di identificare e gestire flussi di dati con priorità diverse.

Poiché IPv4 e IPv6 non sono direttamente compatibili, la transizione è lenta e complessa. Una delle tecniche utilizzate è il **tunnelling**.

Definition 5.5.1: Tunnelling

Una tecnica che permette di trasportare pacchetti di un protocollo (es. IPv6) "incapsulandoli" all'interno di pacchetti di un altro protocollo (es. IPv4). In questo modo, un pacchetto IPv6 può attraversare una porzione di Internet che supporta solo IPv4, come se viaggiasse in un tunnel.

5.6 Il Livello Trasporto

Il livello Rete (IP) fornisce un servizio di consegna *best-effort*, il che significa che i pacchetti possono essere persi, duplicati o arrivare fuori ordine. Il compito del **Livello Trasporto** (Livello 4) è quello di costruire, al di sopra di questa base inaffidabile, servizi di comunicazione end-to-end direttamente per le applicazioni. Su Internet, questo livello è dominato da due protocolli: TCP e UDP.

5.6.1 TCP: Servizio di Trasporto Affidabile (Connection-Oriented)

Il **Transmission Control Protocol (TCP)** è il protocollo che fornisce un servizio di trasporto affidabile e orientato alla connessione. Le sue responsabilità principali sono:

- **Servizio Affidabile**: Garantisce che tutti i dati inviati dal mittente arrivino al destinatario, senza errori, senza perdite e nello stesso ordine in cui sono stati spediti. Per fare ciò, utilizza i meccanismi di numerazione dei segmenti, acknowledgment (ACK) e ritrasmissione basata su timeout.
- **Controllo di Flusso**: Si assicura che il mittente non trasmetta dati a una velocità superiore a quella che il destinatario può gestire, evitando di sovraccaricare il buffer di ricezione di quest'ultimo.
- **Controllo della Congestione**: Modula la velocità di trasmissione per evitare di sovraccaricare la rete stessa (i router intermedi), contribuendo alla stabilità generale di Internet.

5.6.2 UDP: Servizio di Trasporto non Affidabile (Connectionless)

Lo **User Datagram Protocol (UDP)** è l'alternativa a TCP. È un protocollo estremamente semplice e leggero che fornisce un servizio non affidabile e non orientato alla connessione. Essenzialmente, prende i dati dall'applicazione, aggiunge un header con i numeri di porta di origine e destinazione, e li passa al livello IP. Non offre alcuna garanzia: i segmenti possono perdersi, arrivare duplicati o fuori ordine. Il suo unico vantaggio è la **velocità** e il **basso overhead**, che lo rendono ideale per applicazioni sensibili al ritardo che possono tollerare qualche perdita, come le videochiamate, i giochi online o il servizio DNS.

5.6.3 Dettagli del Protocollo TCP

TCP è il cavallo di battaglia di Internet e realizza la sua affidabilità attraverso il concetto di **connessione**.

Definition 5.6.1: Socket

Un socket è l'endpoint di una comunicazione. È una costruzione software, gestita dal sistema operativo, che rappresenta un'estremità di una connessione di rete. Viene identificato univocamente dalla coppia (**Indirizzo IP**, **Numero di Porta**). Il numero di porta (un intero a 16 bit) serve a distinguere tra le diverse applicazioni in esecuzione sullo stesso host.

Il ciclo di vita di una comunicazione TCP si articola in tre fasi:

1. **Apertura della Connessione (Three-Way Handshake)**: Prima di scambiare dati, client e server devono stabilire una connessione. Questo avviene tramite uno scambio di tre pacchetti (SYN, SYN-ACK, ACK) con cui sincronizzano i loro parametri iniziali.
2. **Trasferimento Dati**: I dati vengono scambiati in modo affidabile e bidirezionale.
3. **Chiusura della Connessione**: Al termine dello scambio, la connessione viene chiusa in modo ordinato tramite uno scambio di pacchetti FIN.

Note:

Esistono dei **Numeri di Porta Ben Noti (Well-Known Ports)**, compresi tra 0 e 1023, che sono standardizzati per servizi specifici. Ad esempio, i server web attendono richieste sulla porta **80** (per HTTP), mentre i server di posta per l'invio attendono sulla porta **25** (per SMTP).

5.6.4 Multiplexing, Demultiplexing e Architetture Server

Il livello trasporto gestisce la comunicazione tra processi, non solo tra host.

- **Multiplexing**: Sul lato del mittente, il livello trasporto raccoglie i dati provenienti da più socket (cioè da diverse applicazioni), aggiunge a ciascun segmento l'header TCP/UDP corretto e li passa al livello rete come un unico flusso di pacchetti.
- **Demultiplexing**: Sul lato del ricevente, il livello trasporto riceve i segmenti dal livello rete e, leggendo il numero di porta di destinazione nell'header, smista i dati al socket dell'applicazione corretta.

Per gestire le richieste provenienti da molti client contemporaneamente, i server TCP utilizzano un'architettura specifica:

1. Il server ha un processo principale in ascolto su un **welcoming socket** (es. sulla porta 80).
2. Quando arriva una richiesta di connessione da un client, il sistema operativo del server crea un **nuovo socket dedicato** per quella specifica connessione.
3. Il server tipicamente genera un nuovo processo o, più efficientemente, un nuovo **thread** per gestire la comunicazione con quel client sul nuovo socket.
4. In questo modo, il welcoming socket rimane libero per accettare nuove connessioni, e il server può gestire centinaia di client in parallelo.

5.7 Controllo di Flusso e Congestione in TCP

Un mittente TCP non può inviare dati alla massima velocità possibile senza criterio. Deve adattare la sua velocità di trasmissione per due motivi: non sovraccaricare il ricevente (**controllo di flusso**) e non sovraccaricare la rete (**controllo di congestione**).

Definition 5.7.1: Finestra Scorrevole (Sliding Window)

È il meccanismo fondamentale usato da TCP. La "finestra" è un valore numerico che rappresenta la quantità massima di dati che un mittente può inviare senza aver ancora ricevuto la conferma (ACK) dal destinatario. Questo permette di "riempire il canale" inviando più pacchetti in sequenza invece di attendere l'ACK per ogni singolo pacchetto, migliorando drasticamente l'efficienza.

Il controllo di congestione in TCP è un algoritmo sofisticato che regola dinamicamente la dimensione della finestra, ora chiamata **Congestion Window (CWND)**, per adattarsi alle condizioni della rete.

- **Fase di crescita (Slow Start / Congestion Avoidance)**: TCP inizia inviando pochi dati. Finché riceve gli ACK regolarmente, interpreta che la rete è libera e aumenta esponenzialmente (e poi linearmente) la dimensione della sua finestra, diventando sempre più "aggressivo" nell'invio.
- **Rilevamento della Congestione**: Se un ACK non arriva entro il timeout, TCP assume che un pacchetto sia andato perso a causa della congestione in un router.
- **Fase di riduzione**: In risposta alla congestione, TCP riduce drasticamente la dimensione della CWND (ad esempio, la dimezza o la riporta al valore minimo) e ricomincia a crescere lentamente.

Questo comportamento adattivo, che accelera quando la via è libera e frena bruscamente al primo segno di difficoltà, è la chiave della stabilità di Internet. Il risultato è un andamento a "dente di sega" della velocità di trasmissione.

5.8 Livello Applicazione: Servizi per l'Utente

Il **Livello Applicazione** (Livello 7) è il livello più alto dello stack. Non fornisce servizi ad altri livelli, ma direttamente alle applicazioni con cui l'utente interagisce. Contiene i protocolli che definiscono le regole per specifici servizi di rete.

5.8.1 Il Servizio DNS (Domain Name System)

Gli esseri umani trovano più facile ricordare nomi come 'www.unibo.it', ma i router e i protocolli di rete necessitano di indirizzi IP numerici. Il DNS fa da ponte tra questi due mondi.

Definition 5.8.1: DNS (Domain Name System)

Un sistema di database distribuito e gerarchico che ha il compito di tradurre i nomi di dominio, leggibili dall'uomo, nei corrispondenti indirizzi IP, e viceversa.

Il DNS è strutturato come un albero gerarchico. Quando un'applicazione deve risolvere un nome:

1. Interroga il suo server DNS locale.
2. Se il server locale non conosce la risposta, inizia una catena di interrogazioni che risale la gerarchia: interroga i server **radice** ('.'), poi i server del **Top-Level Domain** (es. '.it', '.com'), e così via, fino a trovare il server autoritativo che conosce l'associazione nome-IP.

Note:

Le query DNS, essendo richieste brevi che necessitano di una risposta rapida, utilizzano tipicamente il protocollo UDP.

5.8.2 Altri Protocolli Applicativi

- **World Wide Web (WWW)**: Utilizza il protocollo **HTTP (HyperText Transfer Protocol)** per il trasferimento di pagine web tra server e client (browser).
- **Posta Elettronica (E-mail)**: Si basa su più protocolli:

  - **SMTP (Simple Mail Transfer Protocol)**: Utilizzato per inviare e trasferire i messaggi tra server di posta.
  - **POP3 (Post Office Protocol 3)** e **IMAP (Internet Mail Access Protocol)**: Utilizzati dai client di posta per scaricare e gestire i messaggi dal server.

5.8.3 Architetture Applicative

Le applicazioni di rete possono essere progettate secondo due paradigmi principali:

- **Client/Server**: È l'architettura tradizionale. Un **server**, un host potente e sempre attivo, offre un servizio. Molti **client** si connettono al server per richiederlo. La comunicazione è centralizzata. Esempi: web, posta elettronica.
- **Peer-to-Peer (P2P)**: Non esiste una distinzione netta tra client e server. Tutti i partecipanti, detti **peer**, sono equivalenti e possono agire sia da client (richiedendo risorse) sia da server (offrendo risorse). La comunicazione è decentralizzata e avviene direttamente tra i peer. Esempio: sistemi di file-sharing come Gnutella.
- **Ibrida**: Un modello che combina elementi di entrambi. Ad esempio, il Napster originale usava un server centrale per l'indicizzazione (per scoprire quali peer avessero un certo file), ma il trasferimento del file avveniva poi in modalità P2P.

5.9 Cenni sulla Sicurezza e Qualità del Servizio

5.9.1 Sicurezza in Rete

La connettività globale di Internet introduce significative sfide di sicurezza. Le contromisure principali includono:

- **Firewall**: Dispositivi, tipicamente router specializzati, posti al confine di una rete privata, che filtrano il traffico in entrata e in uscita sulla base di un insieme di regole di sicurezza, bloccando accessi non autorizzati.
- **Crittografia**: Protegge la **confidenzialità** dei dati, rendendoli illeggibili a chiunque li intercetti senza possedere la chiave di decifratura corretta.
- **Autenticazione**: Processi (come login e password) che verificano l'identità di un utente per garantire che solo le persone autorizzate possano accedere a dati e servizi.
- **Prevenzione da Malware**: Difesa contro software dannosi (virus, worm) che possono diffondersi attraverso la rete e causare danni.

5.9.2 Qualità del Servizio (QoS)

Internet, basata su un modello *best-effort*, non offre garanzie sui tempi di consegna dei pacchetti. Sebbene TCP garantisca che i dati arrivino, non può garantire *quando* arriveranno. Questo è un limite per applicazioni in tempo reale.
Per superare questo limite, sono state sviluppate architetture come **Internet2** e i **Servizi Differenziati**, che introducono il concetto di **Qualità del Servizio (QoS)**.

Definition 5.9.1: Qualità del Servizio (QoS)

Un insieme di tecnologie che permettono a una rete di gestire il traffico in modo differenziato, dando priorità ai pacchetti di applicazioni sensibili al ritardo (es. VoIP, videoconferenze) rispetto a quelli di applicazioni meno critiche (es. e-mail, file transfer). I router abilitati alla QoS possono gestire code separate per i diversi tipi di traffico, assicurando che i pacchetti urgenti vengano inoltrati per primi.

Chapter 6

Il Livello Applicazione

Il livello più alto dello stack, il **Livello Applicazione**, è dove le applicazioni di rete vivono. A differenza degli altri livelli, non fornisce servizi a un livello superiore, ma direttamente alle applicazioni con cui l'utente interagisce (browser, client di posta, ecc.). Questo capitolo esplora i principi di funzionamento di queste applicazioni e approfondisce i protocolli che le rendono possibili, come HTTP e SMTP.

6.1 Principi delle Applicazioni di Rete

La creazione di un'applicazione di rete consiste nello scrivere programmi che vengono eseguiti su host diversi (*end systems*) e comunicano tra loro attraverso la rete. Una caratteristica fondamentale è che gli sviluppatori di applicazioni non hanno bisogno di programmare i dispositivi del nucleo della rete (come router e switch), i quali non eseguono applicazioni utente. Tutta la complessità applicativa risiede ai bordi della rete (*network edge*), un principio di progettazione che ha permesso uno sviluppo e una diffusione delle applicazioni estremamente rapidi.

6.1.1 Architetture Applicative

Le applicazioni di rete sono tipicamente strutturate secondo due paradigmi principali: client-server o peer-to-peer (P2P).

Architettura Client-Server È il modello tradizionale. Le responsabilità sono nettamente separate:

- **Server**: È un host sempre attivo (*always-on*) con un indirizzo IP permanente e conosciuto. Attende passivamente le richieste dai client e risponde fornendo il servizio richiesto. Per gestire un gran numero di richieste, i server sono spesso ospitati in grandi data center.
- **Client**: È un host che avvia la comunicazione inviando una richiesta al server. Può essere connesso in modo intermittente e avere un indirizzo IP dinamico. I client non comunicano mai direttamente tra loro.

Architettura Peer-to-Peer (P2P) In questo modello, non esiste un server centrale sempre attivo. La comunicazione avviene direttamente tra host arbitrari, chiamati **peer**.

- Ogni peer agisce contemporaneamente sia da client, richiedendo servizi ad altri, sia da server, fornendo servizi ad altri.
- Questa architettura gode di **auto-scalabilità**: ogni nuovo peer che si unisce alla rete porta non solo nuova domanda di servizi, ma anche nuova capacità per offrirli.
- I peer sono connessi in modo intermittente e possono cambiare indirizzo IP, rendendo la gestione del sistema più complessa rispetto al modello client-server.

6.1.2 Comunicazione tra Processi

A un livello più basso, la comunicazione di rete avviene tra **processi** (programmi in esecuzione) su host diversi.

- Un processo **client** è quello che inizia la comunicazione.
- Un processo **server** è quello che attende di essere contattato.

I processi si scambiano messaggi attraverso un'interfaccia software chiamata **socket**. Un socket è l'equivalente di una "porta" dell'abitazione di un processo: il processo mittente spinge il messaggio fuori dalla sua porta, affidandosi all'infrastruttura di trasporto (gestita dal sistema operativo) per recapitarlo alla porta del processo destinatario. Per identificare univocamente un processo su Internet, non basta l'indirizzo IP dell'host, poiché su un singolo host possono essere in esecuzione molti processi. L'identificativo completo è quindi la coppia: (**indirizzo IP**, **numero di porta**).

6.1.3 Protocolli di Livello Applicazione

Un protocollo di livello applicazione definisce le regole della comunicazione tra processi. Specifica:

- **Tipi di messaggi**: Ad esempio, messaggi di richiesta e messaggi di risposta.
- **Sintassi dei messaggi**: La struttura dei messaggi, i campi che contengono e come sono formattati.
- **Semantica dei messaggi**: Il significato delle informazioni contenute nei campi.
- **Regole**: Quando e come i processi inviano e rispondono ai messaggi.

Note:

I protocolli possono essere **aperti** (definiti in documenti pubblici chiamati RFC, come HTTP e SMTP) o **proprietari** (controllati da un'entità specifica, come Skype).

6.1.4 Servizi del Livello Trasporto

Le applicazioni hanno esigenze diverse dal livello di trasporto sottostante, che su Internet è fornito da TCP o UDP.

- **TCP (Transmission Control Protocol)**: Offre un servizio **affidabile** e orientato alla connessione. Garantisce che tutti i dati arrivino a destinazione senza errori e in ordine. Include anche meccanismi di controllo di flusso e di congestione. È la scelta per applicazioni che non tollerano la perdita di dati, come il trasferimento di file, il web e l'e-mail.
- **UDP (User Datagram Protocol)**: Offre un servizio "leggero", **non affidabile** e senza connessione. Non offre alcuna garanzia. Il suo vantaggio è la velocità e l'assenza di overhead, che lo rendono adatto per applicazioni sensibili al ritardo che possono tollerare qualche perdita di dati, come lo streaming audio/video o i giochi online.

—

6.2 Il Web e il Protocollo HTTP

Il World Wide Web è un'applicazione basata sul protocollo **HTTP (HyperText Transfer Protocol)**. Una pagina web è composta da **oggetti** (un file HTML di base, immagini JPEG, file audio, ecc.). Ogni oggetto è indirizzabile tramite un **URL (Uniform Resource Locator)**.

Example 6.2.1 (Struttura di un URL)

- 'http': Protocollo di livello applicazione.
- 'www.someschool.edu': Nome dell'host (server).
- '/someDept/pic.gif': Percorso (*path*) dell'oggetto sul server.

HTTP è un protocollo client-server. Il client (un browser) richiede, riceve e visualizza gli oggetti web; il server invia gli oggetti in risposta alle richieste. Utilizza TCP come protocollo di trasporto, tipicamente sulla porta 80.

Definition 6.2.1: Protocollo Stateless

HTTP è un protocollo **stateless** (senza stato), il che significa che il server non conserva alcuna informazione sulle richieste passate di un client. Ogni richiesta è trattata in modo indipendente. Mantenere uno stato renderebbe i server molto più complessi e difficili da gestire in caso di crash.

6.2.1 Connessioni HTTP: Non-persistenti vs. Persistenti

Esistono due modalità con cui HTTP gestisce le connessioni TCP sottostanti.

- **HTTP Non-persistente**: Utilizza una connessione TCP separata per ogni oggetto. Se una pagina contiene 1 file HTML e 10 immagini, vengono aperte e chiuse 11 connessioni TCP. Questo approccio è molto inefficiente, poiché ogni connessione richiede tempo per essere stabilita (almeno un RTT). Il tempo totale di risposta per un oggetto è di circa **2 RTT + tempo di trasmissione del file**.
- **HTTP Persistente (default in HTTP/1.1)**: Permette di scaricare più oggetti attraverso la stessa connessione TCP, che viene mantenuta aperta dal server dopo aver inviato una risposta. Questo riduce drasticamente il ritardo e l'overhead, richiedendo potenzialmente un solo RTT per tutti gli oggetti referenziati dopo il setup iniziale della connessione.

6.2.2 Messaggi HTTP

HTTP definisce due tipi di messaggi, entrambi in formato ASCII leggibile dall'uomo: richiesta e risposta.

Messaggio di Richiesta HTTP È composto da tre parti:

1. **Request Line**: Specifica il metodo, l'URL dell'oggetto e la versione del protocollo. Es: 'GET /index.html HTTP/1.1'.
2. **Header Lines**: Coppie 'nome: valore' che forniscono informazioni aggiuntive. Esempi comuni sono 'Host:' (obbligatorio in HTTP/1.1), 'User-Agent:' (il tipo di browser) e 'Accept-Language:' (la lingua preferita per la risposta).
3. **Entity Body**: Contiene i dati inviati al server, ad esempio i dati di un form compilato dall'utente. È presente solo con alcuni metodi, come POST.

I metodi HTTP principali sono:

- **GET**: Richiede una risorsa. I parametri possono essere passati direttamente nell'URL (es. '.../animalsearch?monkeys&banana').
- **POST**: Invia dati al server affinché li processi (es. i dati di un form), includendoli nell'entity body del messaggio.
- **HEAD**: Come GET, ma chiede al server di non inviare l'oggetto, ma solo gli header. Utile per verificare la data di ultima modifica senza scaricare il file.
- **PUT**: Carica un file sul server, specificato nell'URL.
- **DELETE**: Cancella il file specificato nell'URL.

Messaggio di Risposta HTTP Anche questo è composto da tre parti:

1. **Status Line**: Indica la versione del protocollo, un codice di stato numerico e una frase esplicativa. Es: 'HTTP/1.1 200 OK'.
2. **Header Lines**: Forniscono meta-informazioni sull'oggetto inviato. Esempi: 'Content-Type:' (il tipo di file, es. 'text/html'), 'Content-Length:' (la dimensione in byte) e 'Last-Modified:'.

3. **Entity Body**: L'oggetto richiesto (es. il contenuto del file HTML).

Alcuni codici di stato comuni:

- '200 OK': La richiesta ha avuto successo.
- '301 Moved Permanently': L'oggetto è stato spostato permanentemente a un nuovo URL.
- '400 Bad Request': Il server non ha compreso la richiesta.
- '404 Not Found': L'oggetto richiesto non è stato trovato sul server.

—

6.3 Posta Elettronica (E-mail)

Il sistema di posta elettronica è una delle applicazioni più antiche e fondamentali di Internet. La sua architettura è composta da tre elementi principali: **User Agent**, **Mail Server** e il protocollo **SMTP**.

- **User Agent (UA)**: È il programma con cui l'utente interagisce per comporre, leggere e gestire le email (es. Outlook, Thunderbird, l'app Mail dell'iPhone).
- **Mail Server**: È il cuore del sistema. Ogni utente ha una **mailbox** (casella di posta) su un mail server, dove vengono conservati i messaggi in arrivo. I server dispongono anche di una **coda di messaggi** per le email in uscita, in attesa di essere inviate. I mail server sono sempre attivi.
- **SMTP (Simple Mail Transfer Protocol)**: È il protocollo di livello applicazione che si occupa di trasferire i messaggi **tra i mail server** e dal User Agent del mittente al suo mail server. SMTP è un protocollo di tipo "push", in quanto spinge i messaggi dal client al server.

6.3.1 SMTP (Simple Mail Transfer Protocol)

SMTP è il protocollo standard per l'invio di email, definito nell'RFC 2821.

- Utilizza TCP sulla porta 25 per garantire un trasferimento affidabile dei messaggi.
- La comunicazione avviene tramite comandi e risposte in formato testo ASCII, in modo simile a HTTP.
- Il trasferimento di un messaggio avviene in tre fasi: **handshaking** (presentazione), **trasferimento** dei messaggi e **chiusura** della connessione.
- I messaggi devono essere codificati in ASCII a 7 bit.

Example 6.3.1 (Invio di una mail da Alice a Bob)

1. Alice compone il messaggio con il suo User Agent e lo invia al suo mail server.
2. Il mail server di Alice inserisce il messaggio nella sua coda di uscita.
3. Il lato client di SMTP sul server di Alice apre una connessione TCP con il mail server di Bob.
4. Attraverso questa connessione, il server di Alice invia (push) il messaggio al server di Bob.
5. Il mail server di Bob riceve il messaggio e lo deposita nella mailbox di Bob.
6. Bob, quando vuole, usa il suo User Agent per leggere il messaggio dalla sua mailbox sul suo server.

6.3.2 Protocolli di Accesso alla Posta: POP3 e IMAP

Come visto nell'esempio, SMTP si occupa di far arrivare il messaggio al server del destinatario, ma non di come il destinatario lo preleva. Per quest'ultima fase, sono necessari protocolli di accesso alla posta.

- **POP3 (Post Office Protocol 3)**: È un protocollo molto semplice che permette a un utente di autenticarsi e scaricare i messaggi dalla sua mailbox. La modalità di default è "download and delete": i messaggi vengono spostati dal server al client locale e cancellati dal server. È stateless tra le sessioni.
- **IMAP (Internet Mail Access Protocol)**: È un protocollo più complesso e potente. I messaggi rimangono sempre sul server, e il client li manipola a distanza. Permette di organizzare la posta in cartelle e mantiene lo stato tra una sessione e l'altra (es. quali messaggi sono stati letti). È la scelta ideale per chi accede alla posta da più dispositivi diversi.
- **HTTP**: I servizi di webmail (Gmail, Outlook.com) usano HTTP come protocollo di accesso. L'utente interagisce con un browser, e il web server del servizio di posta agisce da client IMAP/POP per accedere alla mailbox per conto dell'utente.

6.4 Approfondimento sul DNS (Domain Name System)

Come abbiamo visto, gli esseri umani preferiscono usare nomi mnemonici per identificare le risorse di rete (es. 'www.google.com'), mentre i protocolli di rete operano con indirizzi IP numerici. Il DNS è il servizio di livello applicazione che funge da "rubrica telefonica" di Internet, traducendo i nomi in indirizzi e viceversa.

6.4.1 Servizi e Caratteristiche del DNS

Il DNS non si limita a una semplice traduzione nome-IP, ma offre diversi servizi cruciali:

- **Traduzione Hostname-IP**: Il suo compito primario.
- **Host Aliasing**: Un singolo host può avere più nomi (alias). Il DNS può mappare un nome alias a un nome "canonico" più complesso. Ad esempio, 'www.ibm.com' potrebbe essere un alias per 'servereast.backup2.ibm.com'.
- **Mail Server Aliasing**: Permette a un client di posta di trovare il server di posta corretto per un dominio (es. 'unibo.it') interrogando il DNS.
- **Distribuzione del Carico (Load Distribution)**: Per siti web ad alto traffico, un singolo nome di dominio può essere associato a un insieme di indirizzi IP, ognuno corrispondente a un server replicato. Il server DNS, quando interrogato, può restituire a client diversi indirizzi IP differenti, distribuendo così il carico di lavoro su più macchine.

Note:

Il DNS è un esempio perfetto del principio "complessità ai bordi della rete". Sebbene sia una funzione critica per il funzionamento di Internet, è implementato interamente a livello applicazione, senza richiedere modifiche ai router o al nucleo della rete.

6.4.2 Gerarchia del DNS

Una singola base di dati centralizzata per l'intero DNS non sarebbe scalabile. Per questo, il DNS è implementato come un database distribuito e gerarchico, strutturato ad albero.

- **Root DNS Servers**: Al vertice della gerarchia. Esistono 13 server radice logici (identificati dalle lettere da A a M), ma ognuno è replicato in centinaia di server fisici in tutto il mondo per garantire ridondanza e basse latenze.
- **Top-Level Domain (TLD) Servers**: Gestiscono i domini di primo livello come '.com', '.org', '.net', e tutti i domini nazionali come '.it', '.uk', '.fr'.
- **Authoritative DNS Servers**: Sono i server DNS di un'organizzazione, che contengono le associazioni (record) ufficiali tra i nomi degli host di quell'organizzazione (es. 'unibo.it') e i loro indirizzi IP.

6.4.3 Risoluzione dei Nomi e Caching

Quando un host ha bisogno di tradurre un nome in un IP, non interroga direttamente la gerarchia, ma si rivolge al suo **Local DNS Server** (solitamente fornito dall'ISP). Il processo di risoluzione può avvenire in due modi:

- **Query Iterativa**: Il server locale interroga il server radice. Il radice risponde: "Non conosco la risposta, ma chiedi al TLD server per '.com' a questo indirizzo". Il server locale contatta allora il server '.com', che a sua volta lo reindirizza al server autoritativo per 'google.com', e così via. Il "lavoro" di contattare i vari server è a carico del server locale.
- **Query Ricorsiva**: Il server locale chiede al server radice di trovare la risposta per lui. Il server radice contatta il TLD, che contatta l'autoritativo, e la risposta finale viene passata a ritroso fino al server locale. Questo approccio mette un carico maggiore sui server in alto nella gerarchia.

Per rendere il sistema efficiente, il **caching** è fondamentale. Ogni volta che un server DNS riceve una mappatura, la memorizza in una cache locale per un certo periodo di tempo (definito dal valore **TTL - Time To Live**). In questo modo, le richieste successive per lo stesso nome possono essere risolte istantaneamente, senza dover interrogare di nuovo l'intera gerarchia.

—

6.5 Applicazioni Peer-to-Peer (P2P)

Come anticipato, l'architettura P2P si distingue per l'assenza di un server centrale, promuovendo la comunicazione diretta tra i peer.

6.5.1 Analisi delle Prestazioni: Client-Server vs. P2P

L'efficienza del P2P, specialmente nella distribuzione di file di grandi dimensioni a molti utenti, può essere dimostrata matematicamente. Consideriamo la distribuzione di un file di dimensione F a N peer.

- **Tempo di distribuzione in Client-Server**: Il tempo è limitato da due colli di bottiglia: il server deve caricare N copie del file, e il client con la connessione più lenta deve scaricarne una copia. Il tempo totale è quindi $D_{C-S} \geq \max(\frac{N \cdot F}{u_s}, \frac{F}{d_{min}})$, dove u_s è la capacità di upload del server e d_{min} la capacità di download del client più lento. **Il tempo cresce linearmente con il numero di client N**.
- **Tempo di distribuzione in P2P**: Il server deve caricare solo una copia. Il client più lento deve sempre scaricare una copia. Tuttavia, la risorsa critica, ovvero la capacità totale di upload del sistema, ora è data dalla somma della capacità del server e di *tutti i peer* che, una volta ricevuto un pezzo del file, possono a loro volta ridistribuirlo. Il tempo totale è $D_{P2P} \geq \max(\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{N \cdot F}{u_s + \sum_{i=1}^{N} u_i})$. Poiché la capacità totale di upload cresce con N, **il tempo di distribuzione in P2P scala in modo molto più efficiente**.

6.5.2 Caso di Studio: BitTorrent

BitTorrent è un popolare protocollo di file-sharing P2P.

- Il file è diviso in piccoli pezzi (**chunks**). L'insieme dei peer che si scambiano i chunk di un file è chiamato **torrent**.
- Un server centrale, detto **tracker**, non distribuisce il file, ma si limita a tenere traccia dei peer che partecipano al torrent.
- Quando un nuovo peer si unisce, contatta il tracker per ottenere una lista di altri peer. Inizia quindi a scaricare i chunk che gli mancano e, contemporaneamente, a caricare i chunk che già possiede verso altri.
- **Strategia di richiesta**: Per massimizzare la disponibilità, un peer richiede prioritariamente i chunk più **rari** tra i suoi vicini.
- **Strategia di invio (Tit-for-Tat)**: Per incentivare la condivisione, un peer dà priorità di upload a quei peer che gli stanno fornendo dati alla velocità più alta. Questo crea un meccanismo meritocratico: più contribuisci, più velocemente scarichi.

—

6.6 Video Streaming e Content Distribution Networks (CDN)

Lo streaming video costituisce la maggior parte del traffico Internet. Distribuire video a milioni di utenti presenta enormi sfide di scalabilità e di gestione dell'eterogeneità degli utenti (connessioni veloci, lente, mobili, ecc.).

6.6.1 DASH (Dynamic, Adaptive Streaming over HTTP)

DASH è la tecnologia standard per lo streaming video.

Definition 6.6.1: DASH

Una tecnica di streaming in cui il video è suddiviso in piccoli "chunk" di pochi secondi. Ogni chunk è pre-codificato a diverse qualità (bitrate). Un file manifesto, scaricato all'inizio, contiene gli URL di tutte le versioni di tutti i chunk.

L'intelligenza è demandata al client (il player video):

1. Il player misura costantemente la larghezza di banda disponibile.
2. In base alla banda, sceglie e richiede il chunk successivo alla massima qualità possibile che può essere scaricata in tempo, senza causare interruzioni (buffering).
3. Può cambiare la qualità richiesta da un chunk all'altro, adattandosi dinamicamente e in modo trasparente alle fluttuazioni della rete.

6.6.2 Content Distribution Networks (CDN)

Per risolvere il problema della scalabilità, i fornitori di contenuti come Netflix e YouTube non usano un singolo "mega-server". Si affidano invece alle CDN.

Definition 6.6.2: CDN

Una rete di server (chiamati cache o nodi edge), distribuiti geograficamente in tutto il mondo, che memorizzano copie dei contenuti.

Quando un utente richiede un video:

1. La richiesta viene intercettata.
2. Utilizzando meccanismi di reindirizzamento (spesso basati su DNS), l'utente viene indirizzato non al server di origine, ma al server CDN geograficamente più vicino a lui.
3. L'utente riceve il video dal server CDN locale, con una latenza molto più bassa e prestazioni migliori.

Le CDN riducono drasticamente il ritardo per gli utenti, il traffico sulle dorsali Internet e il carico sui server di origine.

—

6.7 Programmazione di Rete: l'API dei Socket

Per concludere, vediamo come, in pratica, un programmatore costruisce applicazioni client/server utilizzando l'API dei socket.

6.7.1 Programmazione con UDP

La programmazione con UDP è connectionless.

- **Server UDP**: Crea un socket legato a una porta specifica e si mette in un ciclo infinito, in attesa di datagrammi. Quando ne riceve uno, estrae i dati, l'indirizzo IP e la porta del client, elabora una risposta e la invia all'indirizzo del client.
- **Client UDP**: Crea un socket, costruisce un messaggio includendo l'IP e la porta del server, e lo invia. Poi si mette in attesa della risposta sullo stesso socket. Non c'è una "connessione" formale.

6.7.2 Programmazione con TCP

La programmazione con TCP è orientata alla connessione.

- **Server TCP**:
  1. Crea un "welcoming socket" legato a una porta specifica.
  2. Chiama la funzione 'listen()' su questo socket, mettendolo in attesa di richieste di connessione.
  3. Chiama la funzione 'accept()', che blocca l'esecuzione finché un client non si connette. Quando ciò accade, 'accept()' crea un **nuovo socket dedicato** a quel client e restituisce il controllo.
  4. Lo scambio di dati avviene sul nuovo socket, mentre il welcoming socket torna in ascolto per altri client.
- **Client TCP**:
  1. Crea un socket.
  2. Chiama la funzione 'connect()', specificando l'IP e la porta del server. Questa chiamata avvia il three-way handshake e stabilisce la connessione.
  3. Una volta che la connessione è stabilita, invia e riceve dati attraverso il socket come se fosse un "tubo" affidabile.

Chapter 7

Introduzione alla sicurezza di rete

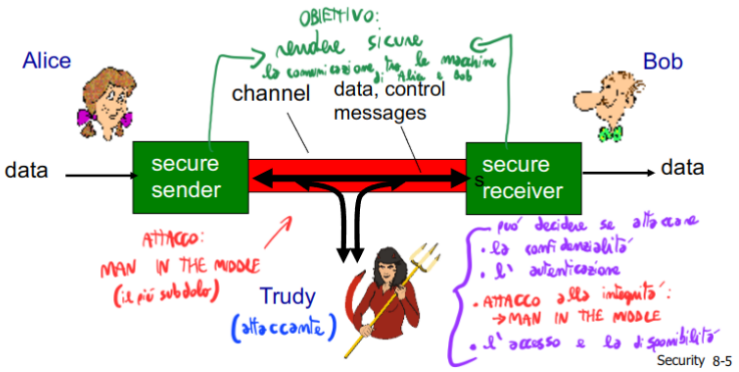
Definition 7.0.1: Sicurezza di rete

La sicurezza di rete è l'insieme delle strategie, tecnologie e protocolli utilizzati per proteggere le reti di calcolatori da accessi non autorizzati, attacchi, perdite di dati e altre minacce informatiche

In pratica consiste nel creare un'architettura di difesa e prevenzione. Questi sono in modo riassuntivo i principali obbiettivi della sicurezza di rete:

- **Confidenzialità**: Solo il mittente e il destinatario devono poter leggere il messaggio. Questo si ottiene con la crittografia
  - Il mittente crittografa (encrypts) il messaggio.
  - Il destinatario decrittografa (decrypts) il messaggio
- **Autenticazione**: Processo per verificare l'identità di un utente o un dispositivo, fondamentale per garantire accessi sicuri (es. Password, certificati digitali, autenticazione a due fattori (2FA))
- **Integrità dei messaggi**: Garantisce che i dati non siano stati alterati durante la trasmissione (es. Password, certificati digitali, autenticazione a due fattori (2FA))
- **Accesso e Disponibilità**: I servizi devono essere accessibili agli utenti autorizzati. Minacce:
  - Attacchi DoS/DDoS che bloccano il servizio.
  - Gestione dei permessi per prevenire accessi non autorizzati

Si introduce il concetto di attacco con il tipico esempio di Alice e Bob che vogliono comunicare e Trudy, l'attaccante, che può intercettare, elliminare o aggiungere messaggi durante la comunicazione dei due amamnti carucci (Morbidelli - Morigi)



Vabbuò passiamo alla sezione sucessiva:

7.1 Principi di crittografa

Partiamo dalla defnizione

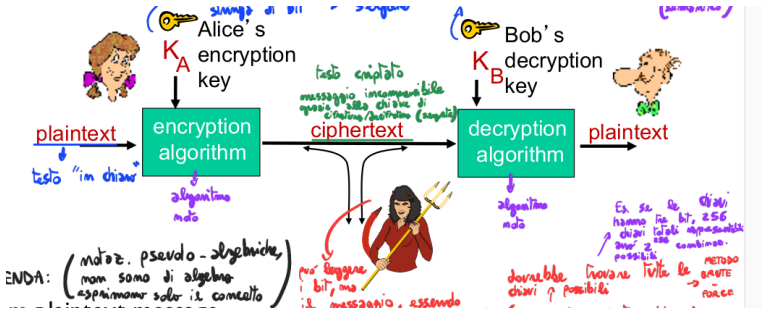
Definition 7.1.1: crittografa

si **crittografa** la disciplina che studia le tecniche per proteggere le informazioni trasformandole in un formato illeggibile per chi non è autorizzato, consentendo solo ai destinatari legittimi di decifrarle.

Nelle reti di comunicazione, i dati trasmessi possono essere intercettati da chiunque abbia accesso al canale di comunicazione, rendendo le informazioni vulnerabili a lettura, modifica o attacchi malevoli. Per proteggere la riservatezza e l'integrità dei dati, si utilizza, quindi, la crittografia, una tecnica che consente di trasformare il testo in chiaro, detto *plaintext*, in un formato incomprensibile chiamato testo cifrato o *ciphertext*, attraverso l'applicazione di un algoritmo di cifratura.
L'obiettivo della crittografia è garantire che, anche se un malintenzionato intercettasse il messaggio durante la trasmissione, non sarebbe in grado di comprenderne il contenuto senza la conoscenza della chiave segreta. Solo il destinatario legittimo, in possesso della chiave corretta, può applicare un algoritmo di decifratura per convertire il testo cifrato nuovamente in testo leggibile.
Va sottolineato che gli algoritmi di cifratura e decifratura sono generalmente pubblici e noti a tutti. Tuttavia, la sicurezza della crittografia si basa sulla segretezza della cosiddetta *chiave crittografica*, ovvero una sequenza di bit utilizzata all'interno di un algoritmo di cifratura per trasformare un messaggio in un formato sicuro e, successivamente, per riconvertirlo nel suo stato originale. Questa chiave viene generata all'inizio della comunicazione tra mittente e destinatario e loro e solo loro ne sono a conoscenza.
Adesso un'introduzione all'algebra della crittografia
Si ha:

- m: plaintext
- $K_A(m)$: ciphertext, encrypted with key K_A
- $m = K_B(K_A(m))$: plaintext ripristinato grazie alla chiave K_B

Si noti la seguente immagine (con appunti bonziani):



7.1.1 Attaccare uno schema di crittografia

Gli attacchi crittografici mirano a violare la sicurezza di un sistema di cifratura. Ne esistono diversi tipi:

1. **Cipher-text only attack**, l'attaccante possiede solo il testo cifrato a disposizione e per decifrare il messaggio ha due approcci possibile:
   - **Forza bruta**: prova tutte le chiavi possibili
   - **Analisi statistica**: cerca pattern ripetuti nei dati cifrati
2. **known-plaintext attack**: Trudy possiede sia il testo cifrato che il corrispondente testo in chiaro. Con queste informazioni, cerca di scoprire la chiave di cifratura o il meccanismo utilizzato, così da poter decifrare altri messaggi cifrati dallo stesso sistema.

Example 7.1.1

Se Trudy sa che nel testo in chiaro compare la parola "hello" e trova nel messaggio cifrato la sequenza "JGRRG", può iniziare a costruire un dizionario di corrispondenze tra lettere:

- $h \rightarrow J$
- $e \rightarrow G$
- $l \rightarrow R$
- $o \rightarrow G$

Note:

un buon metodo per evitare questi attacchi è rendere lo spazio delle chiavi il più vasto possibile e utilizzare algoritmi sicuri che resistano alle analisi statistiche

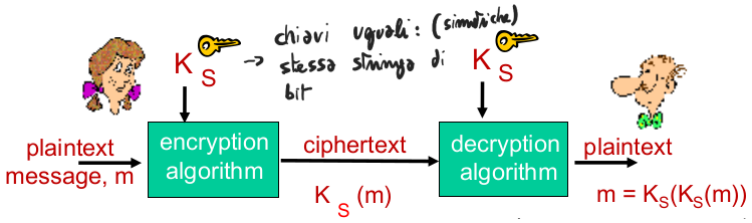
7.1.2 Crittografia a chiave simmetrica

Adesso entriamo nel vivo della difesa cazzo

Definition 7.1.2: Crittografia a chiave simmetrica

mittente e destinatario condividono la stessa chiave segreta K_S per cifrare e decifrare i messaggi

I pratica il mittente (Alice) cirfra il plaintext con l'algoritmo di cifratura usando la chiave $K_S$ottenendo il ciphertext, Bob riceve il ciphertext e lo decifra usando lo stesso algoritmo e la chiave K_S, recuperando il messaggio originale



Per implementare la chiave simmetrica esistono diverse tecniche

Cifrario a sostituzione

Definition 7.1.3: Cirfrario a sostituizione

Un **cifrario a sostituzione** è una tecnica di cifratura in cui ogni lettera del testo in chiaro viene sostituita con un'altra lettera secondo un mapping predefinito

Cifrario monolitico Tra i vari tipi di cifrario a sostruzioni vi è il cifrario monolitico dove ogni lettera dell'alfabeto viene sostituita da un'altra lettera fissa

Example 7.1.2 (Cifrario monolitico)

- Alfabeto in chiaro: abcdefghijklmnopqrstuvwxyz
- Alfabeto cifrato: mnbvcxzasdfghjklpoiuytrewq
- Messaggio originale: bob. i love you. alice
- Messaggio cifrato: nkn. s gktc wky. mgsbc

La chiave di cifratura è una funzione di permutazione ad un insieme di 26 lettere ad un altro insieme di 26 lettere
Tuttavia è poco sicuro perché mantiene la frequenza delle lettere originali e lo rende particolarmente vulnerabile alle analisi delle frequenze

Schema ciclico Per rendere la cifratura più sicura, si può usare più cifrari di sostituzione e applicarli in un ciclo predefinito
La chiave di cifratura quindi è un insieme di n cifrari sostituzione $(M_1, M_2, \ldots, M_n)$ ed un modello cicilico su come usarli. Ogni lettera del plaintext quindi viene cifrata con un cifrario diverso seguendo il ciclo

DES (Data Encryption Standard)

Il DES è stato uno standard di crittografia a chiave simmetrica ampiamente adottato. Funziona come un **cifrario a blocchi**: il messaggio in chiaro viene suddiviso in blocchi di 64 bit, e ogni blocco viene cifrato indipendentemente utilizzando una chiave a 56 bit.
La struttura del DES prevede 16 "round" identici di elaborazione, dove ogni round utilizza una porzione diversa (48 bit) della chiave principale. Questo processo complesso di permutazioni e sostituzioni mira a offuscare la relazione tra il testo in chiaro e quello cifrato.

Note:

Oggi il DES è considerato **insicuro**. Una chiave a 56 bit è troppo corta per resistere a un attacco a forza bruta con la potenza di calcolo moderna. Nel 1998, una chiave DES fu violata in meno di un giorno.

Per aumentarne la sicurezza, è stato introdotto il **3DES (Triple DES)**, che applica l'algoritmo DES tre volte consecutive con tre chiavi diverse, portando la dimensione effettiva della chiave a 168 bit e rendendo gli attacchi a forza bruta computazionalmente impraticabili.

AES (Advanced Encryption Standard)

L'AES è lo standard attuale che ha sostituito il DES. Come il DES, è un cifrario a blocchi, ma è più robusto e flessibile:

- Elabora dati in blocchi da **128 bit**.
- Supporta chiavi di tre dimensioni diverse: **128, 192 o 256 bit**.

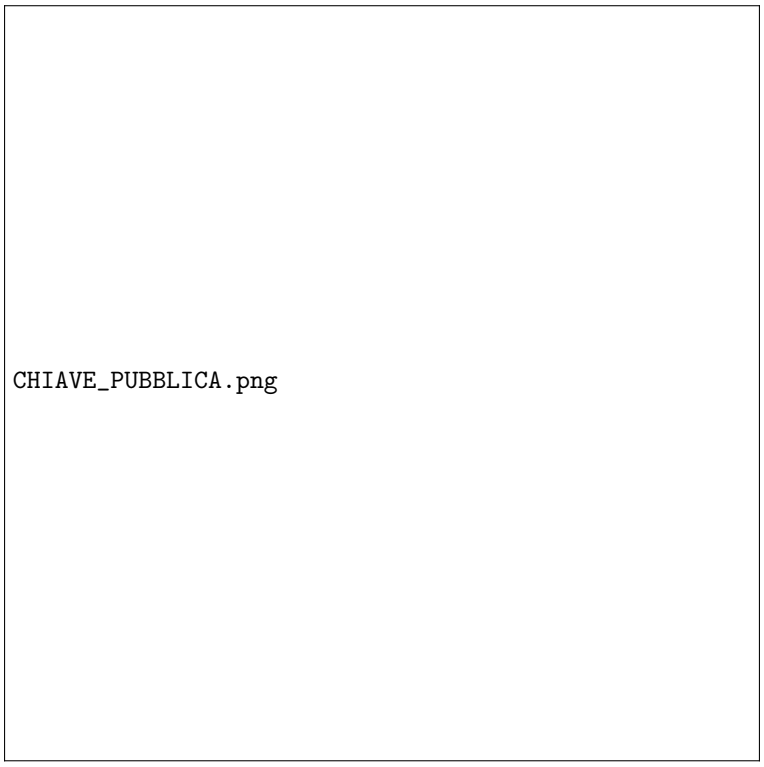
Grazie alla maggiore lunghezza della chiave, l'AES è esponenzialmente più sicuro del DES. Si stima che un attacco a forza bruta che violasse una chiave DES in 1 secondo impiegherebbe circa 149 trilioni di anni per violare una chiave AES a 128 bit.

7.1.3 Crittografia a chiave pubblica (o asimmetrica)

La crittografia a chiave pubblica rappresenta un cambio di paradigma rispetto a quella simmetrica. Il suo vantaggio principale è che **mittente e destinatario non devono condividere una chiave segreta**. Ogni utente possiede una coppia di chiavi:

- Una **chiave pubblica (K^+)**, nota a tutti, che serve per cifrare i messaggi.
- Una **chiave privata (K^-)**, conosciuta solo dal proprietario, che serve per decifrare i messaggi.

Se Alice vuole inviare un messaggio a Bob, cifra il messaggio con la chiave pubblica di Bob (K_B^+). Solo Bob, in possesso della sua chiave privata (K_B^-), potrà decifrarlo.



Questo sistema si basa su **funzioni matematiche unidirezionali (one-way)**, facili da calcolare in una direzione ma computazionalmente impossibili da invertire senza informazioni aggiuntive. Un esempio è la fattorizzazione di numeri molto grandi.

Algoritmo RSA (Rivest, Shamir, Adelson)

L'algoritmo RSA è l'implementazione più nota della crittografia a chiave pubblica. La sua sicurezza si basa sulla difficoltà di fattorizzare un numero intero che è il prodotto di due numeri primi molto grandi.

Generazione delle chiavi

1. Scegliere due numeri primi molto grandi, p e q.
2. Calcolare $n = p \times q$ e $z = (p-1)(q-1)$.
3. Scegliere un numero intero e (con $e < n$) tale che non abbia fattori comuni con z.
4. Calcolare un numero d tale che $ed \pmod{z} = 1$.
5. La **chiave pubblica** è la coppia (n, e).
6. La **chiave privata** è la coppia (n, d).

Cifratura e Decifratura Per poter applicare le operazioni matematiche, un messaggio m viene prima convertito in un numero intero.

- **Cifratura**: $c = m^e \pmod{n}$
- **Decifratura**: $m = c^d \pmod{n}$

La magia matematica assicura che $m = (m^e \pmod{n})^d \pmod{n}$. La sicurezza risiede nel fatto che, pur conoscendo la chiave pubblica (n, e), è computazionalmente infattibile determinare la chiave privata d senza conoscere i fattori primi p e q di n.

Note:

Una proprietà fondamentale di RSA è che le operazioni possono essere invertite: $K_B^+(K_B^-(m)) = m$. Cifrare con la chiave privata e decifrare con quella pubblica è alla base delle **firme digitali**, usate per garantire l'autenticità.

7.1.4 L'approccio ibrido: le chiavi di sessione

La crittografia a chiave pubblica (come RSA) è molto potente ma anche molto **lenta** e computazionalmente costosa. Al contrario, la crittografia a chiave simmetrica (come AES) è estremamente veloce.
Nella pratica, si usa quasi sempre un approccio ibrido che combina il meglio di entrambi i mondi:

1. **Fase 1: Scambio della chiave.** Alice e Bob usano la crittografia a chiave pubblica (lenta ma sicura) per scambiarsi una chiave simmetrica temporanea, detta **chiave di sessione (K_S)**.
2. **Fase 2: Comunicazione.** Una volta che entrambi possiedono K_S, usano la crittografia simmetrica (veloce ed efficiente) con quella chiave per tutta la durata della loro comunicazione.

Questo approccio risolve il problema della condivisione della chiave della crittografia simmetrica, mantenendo al contempo alte prestazioni per lo scambio di grandi quantità di dati.

7.2 Integrità del Messaggio e Autenticazione

Oltre alla confidenzialità, è cruciale garantire che il messaggio non sia stato alterato e che l'identità degli interlocutori sia certa.

7.2.1 Autenticazione

L'obiettivo è permettere a Bob di essere sicuro che sta comunicando con la vera Alice. Protocolli semplici sono insicuri:

- "Sono Alice": Trudy può semplicemente mentire.
- "Sono Alice" con IP di Alice: Trudy può falsificare (spoof) l'indirizzo IP del mittente.
- Invio della password in chiaro: Trudy può intercettare la password e riutilizzarla in un **playback attack**.

Per evitare i playback attack, si usa una tecnica di **challenge-response** con un *nonce*.

Definition 7.2.1: Nonce

Un numero casuale (R) che viene utilizzato una sola volta (*Number used once*) in una sessione di comunicazione.

Il protocollo diventa:

1. Alice si annuncia a Bob.
2. Bob le invia un nonce casuale R.
3. Alice deve cifrare R con una chiave segreta condivisa e inviarlo a Bob. Poiché R è nuovo per ogni sessione, Trudy non può riutilizzare una risposta registrata in passato. Questo prova che Alice è "live" e conosce la chiave segreta.

Questo approccio, seppur efficace per l'autenticazione, è vulnerabile a un attacco **Man-in-the-Middle (MitM)** se la chiave pubblica usata non è certificata.

7.2.2 Firme Digitali e Funzioni Hash

La crittografia a chiave pubblica ha un'altra proprietà fondamentale: $K_B^+(K_B^-(m)) = m$. Cifrare con la chiave privata e decifrare con quella pubblica funziona, ed è la base della **firma digitale**.

Definition 7.2.2: Firma Digitale

Una tecnica crittografica che permette a un mittente di "firmare" un documento digitale per garantirne l'autenticità, l'integrità e il non ripudio (il mittente non può negare di averlo firmato).

Poiché cifrare un intero messaggio con RSA è lento, non si firma il messaggio, ma un suo "riassunto" di dimensione fissa, chiamato *message digest*.

Definition 7.2.3: Funzione Hash Crittografica

Un algoritmo matematico (es. MD5, SHA-1) che prende in input un messaggio di lunghezza arbitraria e produce un output di lunghezza fissa (es. 128 o 160 bit), detto **hash** o **message digest**. Deve essere computazionalmente impossibile trovare due messaggi diversi che producano lo stesso hash.

Il processo di firma digitale diventa:

1. Bob calcola l'hash del suo messaggio, $H(m)$.
2. Bob cifra l'hash con la sua **chiave privata**, $K_B^-(H(m))$. Questo è la firma.
3. Bob invia ad Alice il messaggio in chiaro m e la firma $K_B^-(H(m))$.
4. Alice riceve il tutto, ricalcola autonomamente l'hash del messaggio m, e decifra la firma usando la **chiave pubblica di Bob**, $K_B^+(K_B^-(H(m)))$.
5. Se i due hash coincidono, Alice ha la certezza che il messaggio proviene da Bob e non è stato alterato.

7.2.3 Autorità di Certificazione (Certification Authorities - CA)

Resta un problema: come fa Alice ad essere sicura che la chiave pubblica che sta usando per verificare la firma di Bob appartenga davvero a Bob e non a Trudy? La soluzione sono le Autorità di Certificazione.

Definition 7.2.4: Certification Authority (CA)

Un'entità terza fidata (es. Verisign, Let's Encrypt) che ha il compito di associare un'identità a una chiave pubblica.

Una CA emette un **certificato digitale**, che è un documento elettronico contenente l'identità di un'entità (es. Bob), la sua chiave pubblica, e altri dati. L'intero certificato è **firmato digitalmente dalla CA** con la propria chiave privata.
Quando Alice vuole comunicare con Bob, lui le invia il suo certificato. Alice, che si fida della CA e possiede la chiave pubblica della CA, può verificare la firma sul certificato. Se la verifica ha successo, Alice è certa che la chiave pubblica contenuta nel certificato appartiene legittimamente a Bob, sventando così l'attacco Man-in-the-Middle.

7.3 Mettere in Sicurezza l'E-mail

Un'applicazione come la posta elettronica richiede di garantire tutti e tre i pilastri della sicurezza: confidenzialità, autenticazione e integrità. Questo si ottiene combinando in modo intelligente la crittografia simmetrica e quella asimmetrica.

7.3.1 Confidenzialità per l'E-mail

Per inviare un messaggio confidenziale m a Bob, Alice non usa direttamente la lenta crittografia a chiave pubblica per cifrare l'intero messaggio. Adotta invece un approccio ibrido:

1. Alice genera una **chiave di sessione simmetrica** casuale, K_S.
2. Usa la veloce crittografia simmetrica per cifrare il messaggio: $K_S(m)$.
3. Usa la lenta crittografia a chiave pubblica per cifrare solo la chiave di sessione, utilizzando la **chiave pubblica di Bob**: $K_B^+(K_S)$.
4. Alice invia a Bob entrambi gli elementi: il messaggio cifrato e la chiave di sessione cifrata.

Quando Bob riceve il tutto:

1. Usa la sua **chiave privata** per decifrare e recuperare la chiave di sessione: $K_B^-(K_B^+(K_S)) = K_S$.
2. Usa la chiave di sessione K_S appena recuperata per decifrare il messaggio: $K_S(K_S(m)) = m$.

7.3.2 Autenticazione e Integrità per l'E-mail

Per garantire l'autenticità e l'integrità, Alice utilizza una firma digitale:

1. Calcola un *message digest* del messaggio tramite una funzione hash: $H(m)$.
2. Cifra l'hash con la sua **chiave privata** per creare la firma: $K_A^-(H(m))$.
3. Invia a Bob il messaggio in chiaro m insieme alla firma digitale.

Bob può verificare la firma utilizzando la chiave pubblica di Alice, garantendo così che il messaggio provenga da lei e non sia stato alterato.

7.3.3 Combinare tutte le Proprietà di Sicurezza

Per ottenere contemporaneamente confidenzialità, autenticazione e integrità, Alice combina i due processi: firma prima il messaggio e poi cifra il tutto.

7.4 Mettere in Sicurezza le Connessioni TCP: SSL/TLS

Mentre la sicurezza dell'e-mail protegge il contenuto "a riposo", è spesso necessario proteggere i dati "in transito" per qualsiasi applicazione che utilizzi TCP. La soluzione standard per questo è SSL.

Definition 7.4.1: SSL (Secure Sockets Layer)

Un protocollo che fornisce un canale di comunicazione sicuro, inserendosi tra il livello Applicazione e il livello Trasporto (TCP). Fornisce confidenzialità, integrità e autenticazione a qualsiasi applicazione basata su TCP. La sua versione moderna e standardizzata è chiamata **TLS (Transport Layer Security)**.

SSL è la tecnologia alla base di **HTTPS**. È supportato da quasi tutti i browser e server web e viene utilizzato per proteggere miliardi di transazioni ogni giorno.

7.4.1 Funzionamento di SSL

SSL opera in due fasi principali:

1. **Handshake Phase**: È una fase di negoziazione iniziale in cui client e server si autenticano a vicenda (tipicamente, il server si autentica al client tramite un certificato digitale), concordano gli algoritmi crittografici da utilizzare e generano le chiavi di sessione simmetriche che verranno usate per la comunicazione.
2. **Data Transfer Phase**: I dati dell'applicazione vengono frammentati in record. Ogni record viene protetto con un MAC (Message Authentication Code) per garantirne l'integrità e poi cifrato con la chiave di sessione simmetrica per garantirne la confidenzialità, prima di essere passato a TCP per la trasmissione.

Il protocollo di Handshake L'handshake è il cuore di SSL e stabilisce la connessione sicura. In sintesi:

1. Il client invia un messaggio 'ClientHello' con una lista di suite crittografiche che supporta e un numero casuale (nonce).
2. Il server sceglie una suite dalla lista, invia un 'ServerHello', il suo certificato digitale (contenente la sua chiave pubblica) e un suo nonce.
3. Il client verifica il certificato del server (usando la chiave pubblica della CA). Se valido, genera un segreto condiviso (*pre-master secret*), lo cifra con la chiave pubblica del server e glielo invia.
4. A questo punto, sia client che server, utilizzando i due nonce e il pre-master secret, derivano in modo indipendente lo stesso insieme di chiavi di sessione (per cifratura e MAC in entrambe le direzioni).
5. La fase si conclude con lo scambio di messaggi 'Finished', cifrati con le nuove chiavi, che confermano che l'handshake è avvenuto correttamente e senza manomissioni.

7.5 Sicurezza a Livello Rete: IPsec

SSL/TLS è ottimo per proteggere singole applicazioni, ma cosa succede se si vuole fornire una "copertura di sicurezza" generale a tutto il traffico che esce da un host o da un'intera rete? A questo scopo, si opera a un livello più basso, il Livello 3, con il protocollo **IPsec**.

Definition 7.5.1: IPsec (IP Security)

Una suite di protocolli che fornisce sicurezza a livello di pacchetti IP. Può cifrare e/o autenticare tutto il traffico IP, indipendentemente dall'applicazione (TCP, UDP, ICMP, ecc.). È la tecnologia fondamentale per la creazione di **Virtual Private Networks (VPN)**.

Definition 7.5.2: VPN (Virtual Private Network)

Una rete che utilizza l'infrastruttura di una rete pubblica (come Internet) per trasportare traffico di una rete privata, mantenendolo logicamente separato e sicuro tramite cifratura e "tunnelling". Ad esempio, permette a un'azienda di collegare in modo sicuro la sua sede centrale e una filiale, facendo viaggiare i dati su Internet come se fossero su una linea privata.

7.5.1 Protocolli e Modalità di IPsec

IPsec offre flessibilità attraverso due protocolli e due modalità operative:

- **Protocolli IPsec**:
  - **AH (Authentication Header)**: Fornisce autenticazione dell'origine e integrità dei dati, ma *non* la confidenzialità (i dati non sono cifrati).
  - **ESP (Encapsulating Security Payload)**: Fornisce autenticazione, integrità *e* confidenzialità. È il protocollo più utilizzato.
- **Modalità Operative**:
  - **Transport Mode**: Cifra solo il payload del pacchetto IP (es. il segmento TCP), lasciando l'header IP originale in chiaro. Viene usato per proteggere la comunicazione tra due host finali.
  - **Tunnel Mode**: Cifra l'intero pacchetto IP originale (header incluso), e lo "incapsula" in un nuovo pacchetto IP con un nuovo header. È la modalità utilizzata per le VPN tra router.

7.5.2 Security Associations (SA)

A differenza di IP, che è connectionless, IPsec è **connection-oriented**. Prima che due entità possano scambiarsi dati protetti, devono stabilire una **Security Association (SA)**.

Definition 7.5.3: Security Association (SA)

Un accordo unidirezionale tra due entità che definisce tutti i parametri di sicurezza per il loro traffico, come il protocollo da usare (ESP o AH), gli algoritmi di cifratura e autenticazione, le chiavi e la loro durata. Per una comunicazione bidirezionale sono necessarie due SA (una per ogni direzione).

La configurazione manuale delle SA è impraticabile in reti complesse. Per questo, si utilizza il protocollo **IKE (Internet Key Exchange)** per negoziare e stabilire le SA in modo automatico e sicuro.

7.6 Mettere in Sicurezza le LAN Wireless (IEEE 802.11)

Le reti wireless presentano sfide di sicurezza uniche, poiché il loro mezzo trasmissivo (le onde radio) è intrinsecamente aperto e accessibile a chiunque si trovi nel raggio di copertura, rendendo l'intercettazione molto più semplice rispetto a una rete cablata.

7.6.1 WEP (Wired Equivalent Privacy) e le sue vulnerabilità

Il primo standard di sicurezza per le reti Wi-Fi fu il WEP. Il suo obiettivo era fornire un livello di privacy paragonabile a quello di una rete cablata, ma il suo design si rivelò catastroficamente fallimentare.
WEP utilizza la crittografia a flusso (RC4) con una chiave simmetrica condivisa. Per evitare di usare sempre lo stesso flusso di cifratura, combina la chiave segreta con un **Initialization Vector (IV)** di 24 bit, che dovrebbe essere unico per ogni pacchetto. L'IV viene trasmesso in chiaro insieme al pacchetto cifrato.
Le sue debolezze fatali sono:

- **Spazio dell'IV troppo piccolo**: Un IV a 24 bit è estremamente corto. Su una rete trafficata, la riutilizzazione dello stesso IV è garantita in poche ore o addirittura minuti.
- **IV trasmesso in chiaro**: Un attaccante può facilmente rilevare quando un IV viene riutilizzato.
- **Vulnerabilità della crittografia a flusso**: Se un attaccante conosce il testo in chiaro di un pacchetto cifrato con un certo IV, può calcolare il keystream corrispondente a quell'IV. Quando lo stesso IV viene riutilizzato, l'attaccante può usare il keystream che ha scoperto per decifrare il nuovo pacchetto.

Queste falle hanno reso il WEP completamente insicuro e obsoleto.

7.6.2 802.11i (WPA2/WPA3): la Sicurezza Moderna

Per rimediare alle falle del WEP, è stato introdotto lo standard **IEEE 802.11i**, commercialmente noto come **WPA2** (e il suo successore **WPA3**). Questo standard fornisce un meccanismo di sicurezza robusto, che include:

- Algoritmi di crittografia forti (come AES).
- Meccanismi robusti per l'autenticazione degli utenti e la distribuzione delle chiavi.

In un ambiente aziendale, l'autenticazione non è gestita direttamente dall'Access Point (AP), ma viene delegata a un **Authentication Server (AS)** separato. Il processo avviene in più fasi, durante le quali il client (STA) e il server di autenticazione si autenticano a vicenda e derivano in modo sicuro una gerarchia di chiavi temporanee utilizzate per cifrare i dati della sessione.

7.7 Sicurezza Operativa: Firewall e IDS

Oltre alla crittografia, la sicurezza di una rete si basa su strumenti operativi che ne controllano i confini e ne monitorano il traffico.

7.7.1 Firewall

Definition 7.7.1: Firewall

Un sistema di sicurezza hardware o software che agisce come un filtro al perimetro di una rete, isolando la rete interna "fidata" dalla rete esterna "non fidata" (Internet). Ispeziona il traffico in entrata e in uscita e decide se bloccarlo o lasciarlo passare in base a un insieme di regole di sicurezza predefinite.

I firewall sono essenziali per:

- Prevenire attacchi come il **Denial of Service (DoS)**.
- Impedire accessi non autorizzati ai dati interni.
- Applicare policy di accesso, consentendo la comunicazione solo agli utenti e ai servizi autorizzati.

Esistono tre tipi principali di firewall:

1. **Stateless Packet Filter**: È il tipo più semplice. Esamina ogni pacchetto individualmente, senza tenere traccia delle connessioni. La decisione di inoltrare o bloccare si basa su regole definite in una **Access Control List (ACL)**, che filtrano in base a indirizzi IP, numeri di porta e flag TCP.
2. **Stateful Packet Filter**: È un firewall più intelligente. Tiene traccia dello stato di ogni connessione TCP attiva. Ad esempio, permette l'ingresso di un pacchetto con il flag ACK impostato solo se questo appartiene a una connessione legittimamente avviata dall'interno. Questo lo rende molto più efficace contro tecniche di scansione e attacchi di spoofing.
3. **Application Gateway (Proxy Firewall)**: È il tipo più sofisticato, in quanto opera a livello applicazione. Invece di far comunicare direttamente client e server, agisce da intermediario. Poiché comprende il protocollo applicativo (es. HTTP), può ispezionare il contenuto dei dati e applicare regole molto più granulari, ad esempio bloccando l'upload di certi tipi di file o filtrando comandi specifici.

7.7.2 Intrusion Detection Systems (IDS)

Mentre un firewall applica regole per prevenire attacchi, un IDS si concentra sul rilevarli mentre accadono.

Definition 7.7.2: Intrusion Detection System (IDS)

Un dispositivo o software che monitora il traffico di rete o i sistemi alla ricerca di attività malevole o violazioni delle policy. A differenza di un firewall, il suo scopo principale non è bloccare, ma **rilevare e allertare**.

Le tecniche utilizzate dagli IDS includono:

- **Deep Packet Inspection**: Analisi del contenuto (payload) dei pacchetti alla ricerca di "firme" corrispondenti a virus o attacchi noti.
- **Analisi Correlata**: Esame del traffico su più pacchetti per identificare pattern sospetti che un firewall tradizionale non vedrebbe, come una scansione delle porte, un tentativo di mappatura della rete o un attacco DoS distribuito.

Gli IDS sono spesso dispiegati in congiunzione con i firewall per fornire una difesa a più livelli, offrendo visibilità sulle minacce che potrebbero superare i filtri perimetrali.

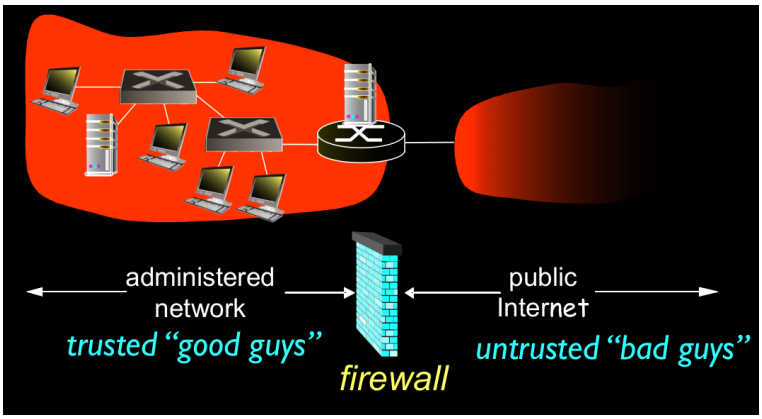
7.8 Firewall e IDS

Innanzi definiamo che cos'è un *Firewall*

Definition 7.8.1: Firewall

Si definisce **Firewall** un sistema di sicurezza che isola una rete interna di un'organizzazione da internet, controllando e filtrando il traffico di rete in entrata e uscita in base a regole di sicurezza definite dette di `ACCESS` o `DENIED`

L'idea del Firewall è che l'esterno di una rete è composta dai cosiddetti "cattivi ragazzi" che la mamma non vi raccomanderebbe come compagni d'uscita, mentre l'interno della rete è composta da "bravi ragazzi" di cui fidarsi, lo scopo del sistema è, quindi, separare i buoni dai cattivi



Il Firewall si rivela utile per principalmente tre motivi:

- **Prevenzione degli attacchi Denial of Service (Dos)**: tipologia di attacco informatico che mira a rendere inaccessibili o indisponibili i servizi di una rete ad utenti legittimi

  Un esempio è il **SYN flooding** in cui un attaccante invia molte richieste di connessione false, esaurendo le risorse del server sovraccaricandolo e impedendo connessioni legittime

- **Protezione dei dati interni da accessi non autorizzati**: ovvero impedisce che gli attaccanti possona *modificare o rubare dati sensibili*

  Un esempio classico è un attaccante che sostituisce il sito web di un'organizzazione con un contenuto malevolo.

- **Accesso selettivo e autorizzato alla rete interna**: Permette l'accesso solo a utenti o dispositivi autenticati, migliorando la sicurezza

Esistono tre tipi di tipoligie di Firewall che approfonoidremo nel dettaglio:

- **Stateless Packet Filter**: Controlla ogni pacchetto singolarmente, senza tenere traccia delle connessioni
- **Statefull Packet Filter**: Tiene traccia dello stato delle connessioni (es. richieste e risposte)
- **Application Gateway** (proxy Firewall): controlla il traffico a livello applicativo (es. HTTP, FTP, e.mail), filtrando così le informazioni che all'interno dei protocolli del livello (ad esempio il contenuto di una mail)

Si notino nel dettaglio

7.8.1 Stateless Packet Filtering

Definition 7.8.2: Stateless Packet Filtering

Lo **Stateless Packet Filtering** è una tecnica di sicurezza di rete in cui un firewall esamina ogni pacchetto individualmente, senza tener conto delle connessioni stabilite in precedenza. Questo significa che ogni pacchetto è valutato isolatamente sulla base di un insieme di regole predefinite

Di solito un firewall con filtraggio stateless è implementato su un router che collega una rete interna a internet. Il router analizza ogni pacchetto in ingresso e in uscita e decide se bloccarlo o lasciarlo in base a regole definite nelle cosiddette "**white list**" (pacchetti che possono passare) e o "**black list**" (lista di pacchetti da bloccare)
Il firewall prende decisioni basandosi su parametri del pacchetto, tra cui:

- **IP** del mittente e destinatario
- **Numero di porta TCP/UDP** del mittente e destinatario

- **Tipo di messaggio ICMP** bloccando, ad esempio, attacchi di scansione provenienti dall'esterno
- **bit SYN e ACK nei pacchetti TCP**: Il firewall può, ad esempio, bloccare pacchetti con SYN in entrata per impedire connessioni indesiderate dall'esterno

Qui degli esempietti:

Example 7.8.1 (Bloccare tutti i pacchetti con protocollo UDP o Telnet)

- **Regola**: blocca i pacchetti in entrata e in uscita se il protocollo IP è 17 (UDP) e se con la porta di origine o desitinazione è 23
- **Risultato**: Tutto il traffico UDP e telnet vengono bloccati

Example 7.8.2 (Bloccare pacchetti TCP in ingresso con ACK=0)

- **Regola**: blocca tutti i pacchetti TCP in ingresso se il bit ACK = 0
- **Risultato**:Le connessioni in entrata non possono essere iniziate da un host esterno verso la rete interna e le connessioni in uscita funzionano normalmente, perché il traffico di ritorno (che ha ACK=1) è consentito.

Riporto qui una tabella con altri esempi:

Politica	**Impostazione Firewall**
Nessun accesso Web esterno.	Elimina tutti i pacchetti in uscita verso qualsiasi indirizzo IP, porta 80
Nessuna connessione TCP in entrata, tranne quelle per il server Web pubblico dell'istituzione.	Elimina tutti i pacchetti TCP SYN in entrata verso qualsiasi IP tranne 130.207.244.203, porta 80
Impedisci alle Web-radio di consumare la larghezza di banda disponibile.	Elimina tutti i pacchetti UDP in entrata - tranne DNS e broadcast del router.
Impedisci alla tua rete di essere utilizzata per un attacco DoS smurf.	Elimina tutti i pacchetti ICMP diretti a un indirizzo di "broadcast" (ad esempio, 130.207.255.255).
Impedisci alla tua rete di essere tracciata	Elimina tutto il traffico ICMP TTL scaduto in uscita

Access control list

Definition 7.8.3: Liste di controllo d'accesso

Le **ACL** sono tabelle di regole applicate, *con priorità dall'alto verso il basso*, ai pacchetti in arrivo per decidere se consentire (allow) o bloccare (deny) il traffico di rete

Queste tabelle sono il vero e proprio cuore pulsante del Firewall e indicano quale pacchetto può passare e chi no

Example 7.8.3 (ACL)

azione	**indirizzo sorgente**	**indirizzo destinazione**	**protocollo**	**porta sorgente**	**p**
allow	222.22/16	outside of 222.22/16	TCP	> 1023	8
allow	outside of 222.22/16	222.22/16	TCP	80	>
allow	222.22/16	outside of 222.22/16	UDP	> 1023	5
allow	outside of 222.22/16	222.22/16	UDP	53	>
deny	all	all	all	all	a

Criticità

Un serio problema dei Firewall Stateless che potrebbero lasciare passare pacchetti che non hanno senso nel contesto di una connessione. Esempio:

- Un pacchetto con destinazione porta 80 (HTTP) e ACK=1 arriva dall'esterno
- Il firewall stateless lo accetta, anche se nessuna connessione HTTP è stata avviata da un client interno.
- Un attaccante potrebbe sfruttare questa debolezza per inviare pacchetti falsi alla rete interna

Per porre rimedio a questi tipi di problemi si veda la tipoligia di firewall sucessiva

7.8.2 Stateful packet filtering

Definition 7.8.4: Stateful packet filtering

Lo **Stateful packet filtering** è una tecnica di filtraggio di pacchetti tenendo traccia delle connessioni attive e delle loro fasi

Un firewall stateful tiene traccia di ogni connessione TCP attiva e controlla le tre fasi fondamentali della connessione TCP (assicurandosi che ogni pacchetto in entrata o uscita abbia senso):

- **Setup**: Il firewall rileva il pacchetto SYN iniziale, segnalando l'inizio di una connessione
- **Trasferimento dati**: I pacchetti con ACK vengono accettati solo se appartengono a una connessione già avviata
- **Chiusura**: Quando un pacchetto FIN o un timeout segna la fine di una connessione, il firewall non accetta più pacchetti fino a che non se apre un altro

Chapter 8

Il Livello Rete: Il Piano di Controllo (Control Plane)

Dopo aver analizzato il *data plane*, ovvero come un router inoltra i pacchetti basandosi su una tabella di forwarding, esploriamo ora il *control plane*. Il suo compito è determinare come vengono costruite e aggiornate queste tabelle. È qui che risiede l'intelligenza della rete, incarnata dagli algoritmi di routing.

8.1 Introduzione al Piano di Controllo

Il piano di controllo ha il compito di calcolare i percorsi che i pacchetti devono seguire. Questo processo, chiamato **routing**, si contrappone al **forwarding** (l'azione meccanica di inoltro dei pacchetti). L'obiettivo di un algoritmo di routing è popolare le tabelle di forwarding dei router con percorsi "buoni", ovvero a costo minimo.
Come visto in precedenza, il control plane può essere implementato in due modi:

- **Per-Router Control Plane (Tradizionale)**: Ogni router esegue un algoritmo di routing e comunica con gli altri router per calcolare in modo distribuito la propria tabella di forwarding.
- **Logically Centralized Control Plane (SDN)**: Un controller centrale calcola tutte le tabelle e le distribuisce ai router, che agiscono come semplici esecutori.

In questa sezione ci concentreremo sul primo approccio, quello tradizionale, che è alla base dei protocolli storici di Internet.

8.2 Algoritmi di Routing

Per un algoritmo di routing, la rete è un **grafo**, dove i nodi sono i router e gli archi sono i link fisici tra di essi. A ogni arco è associato un **costo**, che può rappresentare la distanza, il ritardo, o essere inversamente proporzionale alla banda. L'obiettivo dell'algoritmo è trovare il percorso a costo minimo tra una sorgente e una destinazione.

8.2.1 Classificazione degli Algoritmi di Routing

Gli algoritmi di routing si classificano principalmente secondo due criteri:

1. **Globali vs. Decentralizzati**:
    - **Algoritmi Link-State (Globali)**: Ogni router possiede una conoscenza completa della topologia della rete e dei costi di tutti i link. Con questa mappa globale, ogni router può calcolare il percorso migliore in autonomia.
    - **Algoritmi Distance-Vector (Decentralizzati)**: Ogni router conosce solo i propri vicini diretti e i costi per raggiungerli. I percorsi vengono calcolati in modo iterativo, attraverso lo scambio di informazioni solo con i router adiacenti. Nessun router ha una visione completa della rete.
2. **Statici vs. Dinamici**:

- **Routing Statico**: I percorsi vengono configurati manualmente e cambiano raramente. Adatto a reti piccole e stabili.
- **Routing Dinamico**: I percorsi vengono aggiornati automaticamente in risposta a cambiamenti nella topologia o nei costi dei link. È l'approccio utilizzato nella maggior parte delle reti moderne.

8.2.2 Algoritmi Link-State (LS)

L'approccio Link-State si basa su un principio semplice: "informa tutti di tutto".

1. Ogni router scopre i propri vicini e il costo dei link per raggiungerli.
2. Queste informazioni (il suo "link state") vengono impacchettate e diffuse a **tutti gli altri router** della rete tramite un processo chiamato *flooding*.
3. Al termine di questo processo, ogni router possiede una mappa completa e identica dell'intera rete.
4. Su questa mappa, ogni router esegue in locale un algoritmo per trovare il percorso a costo minimo verso tutte le destinazioni. L'algoritmo standard per questo compito è l'**algoritmo di Dijkstra**.

L'Algoritmo di Dijkstra L'algoritmo di Dijkstra calcola, da un nodo sorgente 'u', l'albero dei cammini minimi verso tutti gli altri nodi del grafo. Procede in modo iterativo: a ogni passo, aggiunge all'insieme dei nodi 'N'' di cui si conosce il percorso definitivo, il nodo 'w' non ancora in 'N'' che ha il costo minimo attuale. Successivamente, aggiorna i costi di tutti i vicini di 'w'.

Example 8.2.1 (Esempio di Dijkstra)

Dato il grafo seguente, calcoliamo i percorsi a costo minimo a partire dal nodo 'u'.
L'algoritmo procede per passi, aggiornando la tabella dei costi. Alla fine, il risultato per il nodo 'u' sarà la seguente tabella di forwarding:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Questo significa che per raggiungere 'y', 'w' o 'z', 'u' deve prima inviare il pacchetto al suo vicino 'x'.

8.2.3 Algoritmi Distance-Vector (DV)

L'approccio Distance-Vector è decentralizzato e si basa sul principio "informa solo i tuoi vicini". Ogni router conosce molto poco della rete, ma collabora con i vicini per costruire una visione globale.
L'algoritmo si basa sull'equazione di **Bellman-Ford**:

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

In parole: il costo minimo per andare da 'x' a 'y' è il minimo, calcolato su tutti i vicini 'v' di 'x', della somma tra il costo per raggiungere il vicino 'v' ($c(x,v)$) e il costo che 'v' dichiara di avere per raggiungere 'y' ($d_v(y)$).
Il processo è iterativo:

1. Ogni router mantiene un proprio **distance vector**, ovvero una stima del costo per raggiungere ogni destinazione.
2. Periodicamente, ogni router invia il proprio distance vector ai suoi vicini.
3. Quando un router riceve un nuovo distance vector da un vicino, ricalcola le proprie stime usando l'equazione di Bellman-Ford.
4. Se i propri costi cambiano, informa a sua volta i suoi vicini.

Questo processo continua finché le stime non convergono ai valori reali.

Problemi del DV: Count-to-Infinity Un grosso svantaggio degli algoritmi DV è il problema del "count-to-infinity". Mentre le "buone notizie" (la scoperta di un percorso migliore con costo inferiore) si propagano velocemente, le "cattive notizie" (un aumento del costo di un link o un guasto) si propagano molto lentamente. In caso di guasto, i router possono entrare in un loop in cui si scambiano informazioni di routing errate, facendo crescere il costo di un percorso verso l'infinito prima di rendersi conto che la destinazione non è più raggiungibile.

8.2.4 Confronto tra LS e DV

- **Complessità dei messaggi**: LS richiede la diffusione di informazioni a tutti i nodi ($O(nE)$ messaggi), mentre DV scambia informazioni solo tra vicini.
- **Velocità di convergenza**: LS è più veloce e robusto. DV può essere lento a convergere e soffre del problema del count-to-infinity.
- **Robustezza**: In LS, un router malfunzionante può inviare informazioni errate sul costo dei propri link, ma ogni altro router calcola la propria tabella in autonomia. In DV, l'errore di un router può propagarsi e influenzare le tabelle di molti altri router nella rete.

8.3 Routing su Internet: Intra-AS e Inter-AS

Internet è troppo grande per essere gestita da un unico algoritmo di routing. Per garantire la scalabilità, è organizzata gerarchicamente in **Sistemi Autonomi (AS)**.

Definition 8.3.1: Sistema Autonomo (AS)

Un insieme di router e reti sotto un'unica amministrazione tecnica, che presenta una politica di routing comune verso il resto di Internet. Esempi sono la rete di un provider (come TIM), di una grande università o di un'azienda come Google.

Questa struttura divide il problema del routing in due:

1. **Routing Intra-AS (Interior Gateway Protocol - IGP)**: L'algoritmo di routing utilizzato *all'interno* di un singolo AS. Tutti i router dell'AS eseguono lo stesso protocollo. L'obiettivo principale qui è l'ottimizzazione delle prestazioni (trovare il percorso migliore).
2. **Routing Inter-AS (Exterior Gateway Protocol - EGP)**: L'algoritmo utilizzato per scambiare informazioni di raggiungibilità *tra* AS diversi. L'obiettivo qui non è tanto trovare il percorso più performante, quanto implementare le **politiche** commerciali e di interconnessione tra i provider.

I router al confine di un AS, che si connettono ad altri AS, sono chiamati **gateway router**.

8.3.1 OSPF: Un Protocollo di Routing Intra-AS

OSPF (Open Shortest Path First) è uno dei protocolli IGP più diffusi su Internet.

- È un protocollo di tipo **Link-State**.
- Ogni router in un AS che usa OSPF ottiene una mappa completa dell'AS.
- Su questa mappa, ogni router esegue l'**algoritmo di Dijkstra** per calcolare i percorsi a costo minimo.
- "Open" significa che le sue specifiche sono pubbliche (non è proprietario).
- Offre funzionalità avanzate come l'autenticazione dei messaggi di routing per la sicurezza e la possibilità di definire percorsi multipli a parità di costo.

Per migliorare ulteriormente la scalabilità all'interno di AS molto grandi, OSPF può essere strutturato gerarchicamente, suddividendo l'AS in **aree**. I router all'interno di un'area conoscono solo la topologia della propria area, riducendo così il sovraccarico di comunicazione e di calcolo.

8.4 Routing Inter-AS: il Protocollo BGP

Mentre protocolli come OSPF gestiscono in modo efficiente il routing *all'interno* di un singolo Sistema Autonomo (AS), è necessario un protocollo diverso per gestire il routing *tra* AS differenti. Questo compito è affidato al BGP.

Definition 8.4.1: BGP (Border Gateway Protocol)

È il protocollo di routing Inter-AS (o EGP) standard de facto su Internet. È la "colla" che tiene insieme le migliaia di reti dei provider (ISP), permettendo loro di scambiarsi informazioni su quali destinazioni sono raggiungibili attraverso le rispettive reti.

BGP permette a ogni AS di:

- **Ottenere informazioni di raggiungibilità** dai propri AS vicini. Questo avviene tramite una sessione **eBGP** (external BGP) tra i gateway router di confine.
- **Propagare queste informazioni** a tutti i router interni al proprio AS. Questo avviene tramite sessioni **iBGP** (internal BGP).
- **Determinare le rotte "migliori"** verso le reti esterne, basandosi non solo sulla performance, ma soprattutto su **politiche** economiche e di interconnessione.

8.4.1 Path Vector e Politiche di Routing

A differenza di OSPF, BGP non si basa sul costo dei link, ma è un protocollo **Path Vector**.

- Quando un AS pubblicizza una rotta verso una destinazione, non comunica un semplice costo, ma l'intero percorso di Sistemi Autonomi da attraversare. Questo percorso è contenuto in un attributo chiamato **AS-PATH**.
- L'attributo AS-PATH è fondamentale per prevenire i loop: un router scarta ogni rotta pubblicizzata in cui compare già il numero del proprio AS.

La caratteristica più importante di BGP è che la selezione delle rotte è guidata dalla **policy**. Un provider può decidere, ad esempio, di non far transitare traffico tra due altri provider attraverso la propria rete (perché non ne trarrebbe alcun guadagno). Questa policy viene implementata semplicemente scegliendo di non ri-pubblicizzare certe rotte ai propri vicini.

Example 8.4.1 (Hot Potato Routing)

Una comune politica di routing in BGP è l'hot potato routing. Se un router all'interno di un AS ha più possibili uscite per raggiungere una destinazione esterna, sceglierà l'uscita che ha il costo *interno* (calcolato con OSPF) più basso. In pratica, cerca di "liberarsi" del pacchetto il più velocemente possibile, passandolo a un altro AS, anche se questo potrebbe non corrispondere al percorso globale più breve per il pacchetto stesso.

8.5 Approfondimento sul Control Plane SDN

Come anticipato, l'approccio **Software-Defined Networking (SDN)** rappresenta un cambio di paradigma rispetto al routing tradizionale, separando il "cervello" della rete (il control plane) dal "corpo" (il data plane).

8.5.1 L'Architettura di un Controller SDN

Un controller SDN è un sistema software complesso, un vero e proprio "sistema operativo di rete", che può essere scomposto in tre livelli funzionali:

1. **Livello di Comunicazione (Southbound API)**: È l'interfaccia verso il basso, con cui il controller comunica con i dispositivi di rete (gli switch). Il protocollo più comune per questa interfaccia è **OpenFlow**. Tramite questa API, il controller può installare/modificare le tabelle di flusso e ricevere notifiche dagli switch (es. un link è caduto).

2. **Livello di Gestione dello Stato della Rete**: È il cuore del controller. Mantiene una visione centralizzata e costantemente aggiornata dell'intera rete: la topologia (switch e link), le statistiche di traffico, le informazioni sugli host, ecc.
3. **Interfaccia per le Applicazioni (Northbound API)**: È l'interfaccia verso l'alto, con cui le **applicazioni di controllo di rete** interagiscono con il controller per implementare la logica di rete. Queste applicazioni (per il routing, il bilanciamento del carico, il controllo degli accessi) possono leggere lo stato della rete dal controller e usare l'API per programmare il comportamento desiderato, che il controller tradurrà in regole di flusso per gli switch.

Example 8.5.1 (Gestione di un guasto in una rete SDN)

1. Lo switch S1 rileva che un suo link è caduto.
2. S1 invia un messaggio OpenFlow 'Port Status' al controller.
3. Il controller aggiorna il suo stato interno della rete per riflettere il guasto.
4. Un'applicazione di routing, che si era registrata per ricevere notifiche sui cambiamenti di topologia, viene attivata.
5. L'applicazione di routing ricalcola i nuovi percorsi a costo minimo (es. con Dijkstra) basandosi sulla nuova topologia.
6. L'applicazione istruisce il controller di installare le nuove tabelle di flusso negli switch interessati dal cambiamento.
7. Il controller traduce queste istruzioni in messaggi OpenFlow 'Modify-State' e li invia agli switch, riprogrammando così il forwarding della rete in tempo reale.

8.6 ICMP: il Protocollo di Controllo di Internet

Definition 8.6.1: ICMP (Internet Control Message Protocol)

Un protocollo di supporto del livello rete, utilizzato da host e router per comunicare informazioni di controllo e, soprattutto, per segnalare errori. I messaggi ICMP non sono pacchetti ordinari, ma sono incapsulati direttamente all'interno di datagrammi IP.

Un messaggio ICMP è definito da un campo 'Type' e un campo 'Code' che ne specificano il significato. I messaggi più comuni includono:

- **Echo Request/Reply (Type 8/0)**: Usati dal comando **ping** per verificare la raggiungibilità di un host.
- **Destination Unreachable (Type 3)**: Inviato da un router quando non può inoltrare un pacchetto (es. host irraggiungibile, porta non disponibile).
- **Time Exceeded (Type 11)**: Inviato da un router quando scarta un pacchetto perché il suo campo TTL ha raggiunto zero. È il meccanismo alla base di **traceroute**.

Example 8.6.1 (Funzionamento di Traceroute)

Traceroute scopre i router su un percorso inviando una serie di pacchetti (tipicamente UDP) verso la destinazione:

1. Il primo pacchetto viene inviato con un TTL=1. Il primo router lungo il percorso lo scarta e invia indietro un ICMP 'Time Exceeded'.
2. Il secondo pacchetto viene inviato con TTL=2. Il secondo router lo scarta e invia indietro un ICMP.
3. Il processo continua, aumentando il TTL di uno a ogni passo, fino a quando i pacchetti non raggiun-

gono la destinazione finale.

4. L'host di destinazione, ricevendo un pacchetto UDP destinato a una porta non in uso, risponde con un ICMP 'Destination Port Unreachable', segnalando la fine del tracciamento.

8.7 Reti Wireless e il Livello Fisico

Le reti wireless hanno rivoluzionato il modo in cui ci connettiamo, ma la loro apparente semplicità nasconde una notevole complessità a livello fisico. La trasmissione di dati attraverso l'etere, senza un mezzo guidato come un cavo, è soggetta a una moltitudine di fenomeni fisici che ne influenzano le prestazioni e l'affidabilità. Questo capitolo esplora i fondamenti delle onde radio, la loro propagazione e i parametri chiave per la progettazione di un sistema wireless.

8.8 Fondamenti delle Onde Radio (RF)

Definition 8.8.1: Onda a Radiofrequenza (RF)

Un'onda elettromagnetica generata da una corrente alternata ad alta frequenza che scorre in un'antenna. L'antenna agisce come un trasduttore, convertendo la corrente elettrica in onde elettromagnetiche durante la trasmissione e viceversa durante la ricezione.

Un'onda elettromagnetica è composta da un campo elettrico e un campo magnetico che oscillano in modo perpendicolare tra loro e alla direzione di propagazione. Questa onda può viaggiare anche nel vuoto, a differenza della corrente elettrica che necessita di un conduttore.

8.8.1 Parametri di un'Onda RF

Un'onda RF sinusoidale è caratterizzata da tre parametri fondamentali, che possono essere modulati per codificare le informazioni (i bit):

- **Ampiezza (Amplitude)**: Rappresenta l'intensità o la potenza del segnale, misurata in Watt (W) o milliwatt (mW). A parità di altre condizioni, un segnale con ampiezza maggiore può percorrere una distanza maggiore prima di diventare troppo debole per essere ricevuto. La potenza necessaria per coprire una certa distanza cresce in modo quadratico (o peggio) con la distanza stessa: per raddoppiare la portata, è necessario quadruplicare la potenza.
- **Frequenza (Frequency)**: Misurata in Hertz (Hz), indica il numero di oscillazioni complete che l'onda compie in un secondo. Le diverse tecnologie wireless operano su bande di frequenza specifiche, assegnate da enti regolatori per evitare interferenze.
- **Lunghezza d'onda (Wavelength, λ)**: È la distanza fisica tra due picchi consecutivi dell'onda. È inversamente proporzionale alla frequenza, secondo la relazione $\lambda = c/f$, dove c è la velocità della luce. La lunghezza d'onda è un parametro critico per la progettazione delle antenne, la cui dimensione fisica ottimale è tipicamente una frazione ($\frac{1}{2}$ o $\frac{1}{4}$) della lunghezza d'onda del segnale.

Example 8.8.1 (Dimensione di un'antenna Wi-Fi)

Una rete Wi-Fi opera nella banda a 2.4 GHz (2.4×10^9 Hz). La sua lunghezza d'onda è:

$$\lambda = \frac{3 \times 10^8 \text{ m/s}}{2.4 \times 10^9 \text{ Hz}} = 0.125 \text{ m} = 12.5 \text{ cm}$$

Per questo motivo, le antenne dei router Wi-Fi hanno dimensioni dell'ordine di pochi centimetri. Al contrario, una radio FM che trasmette a 100 MHz ha una lunghezza d'onda di 3 metri, richiedendo antenne molto più grandi.

8.9 Propagazione dei Segnali Wireless

Nel mondo reale, la propagazione di un'onda RF è tutt'altro che lineare. È influenzata dall'ambiente in modi complessi.

8.9.1 Attenuazione e Aree di Copertura

L'energia di un segnale si disperde man mano che si allontana dalla fonte, un fenomeno noto come **attenuazione**. Questo definisce tre aree di copertura concentriche attorno a un trasmettitore:

1. **Transmission Range**: L'area in cui il segnale è sufficientemente forte per permettere una comunicazione affidabile e a basso tasso di errore.
2. **Detection Range**: L'area in cui il segnale può ancora essere rilevato, ma è troppo debole per estrarre le informazioni in modo corretto.
3. **Interference Range**: L'area più esterna, in cui il segnale non è più rilevabile singolarmente ma contribuisce comunque al rumore di fondo, potenzialmente interferendo con altre comunicazioni.

8.9.2 Effetti Ambientali sulla Propagazione

Un segnale RF che viaggia attraverso l'ambiente interagisce con gli oggetti che incontra, dando luogo a diversi fenomeni:

- **Shadowing (Assorbimento)**: Ostacoli come muri, edifici o vegetazione assorbono parte dell'energia del segnale, creando "zone d'ombra" con una copertura molto debole. Le frequenze più basse tendono a penetrare gli ostacoli meglio di quelle più alte.
- **Riflessione, Diffrazione e Scattering**: Il segnale viene riflesso da superfici grandi (come un palazzo), diffratto attorno agli spigoli e disperso (scattering) da oggetti piccoli.

8.9.3 Multipath Propagation e Fase

A causa di questi fenomeni, il segnale raggiunge il ricevitore attraverso percorsi multipli. Il ricevitore riceve quindi non una singola onda, ma la sovrapposizione del segnale diretto (se presente) e di molteplici "echi" riflessi che arrivano in momenti diversi e con fasi diverse.

Definition 8.9.1: Multipath Propagation (Propagazione Multipath)

La ricezione di un segnale radio come somma di più repliche dello stesso, che hanno percorso cammini diversi e sono quindi sfasate nel tempo e nella fase.

La **fase** di un'onda descrive la sua posizione nel ciclo di oscillazione. L'effetto del multipath dipende criticamente dalla fase con cui gli echi si sovrappongono:

- **Interferenza Costruttiva**: Se due onde arrivano "in fase", le loro ampiezze si sommano, rafforzando il segnale ricevuto.
- **Interferenza Distruttiva**: Se due onde arrivano "in opposizione di fase" (sfasate di 180°), le loro ampiezze si sottraggono, indebolendo o addirittura annullando completamente il segnale.

Questo è il motivo per cui, spostandosi anche solo di pochi centimetri in un ambiente interno, la qualità del segnale Wi-Fi può cambiare drasticamente.

8.9.4 Mobilità e Fading

Quando un dispositivo wireless è in movimento, il suo ambiente multipath cambia continuamente, causando rapide e profonde fluttuazioni della potenza del segnale ricevuto.

Definition 8.9.2: Fading

La variazione della potenza del segnale ricevuto dovuta agli effetti del multipath. Può essere *short-term* (variazioni veloci dovute all'interferenza) o *long-term* (variazioni lente dovute allo shadowing).

8.9.5 Polarizzazione

La polarizzazione descrive l'orientamento del campo elettrico dell'onda radio ed è determinata dall'orientamento fisico dell'antenna. Per un trasferimento di potenza ottimale, l'antenna trasmittente e quella ricevente devono avere la stessa polarizzazione (es. entrambe verticali). Una polarizzazione disallineata (es. una verticale e una orizzontale) introduce una significativa perdita di segnale.

8.10 Progettazione e Misura della Potenza nei Sistemi Wireless

La progettazione di un sistema wireless affidabile richiede un'attenta analisi di tutti i guadagni e le perdite di potenza lungo il percorso, un processo noto come *link budget*.

8.10.1 Unità di Misura della Potenza: dB, dBm, dBi

Poiché la potenza dei segnali RF varia su molti ordini di grandezza, si utilizzano unità logaritmiche per semplificare i calcoli.

- **Decibel (dB)**: È un'unità **relativa** che esprime il rapporto tra due valori di potenza. Trasforma moltiplicazioni e divisioni in addizioni e sottrazioni.
  - $+3$ dB $\approx \times 2$ la potenza
  - -3 dB $\approx /2$ la potenza
  - $+10$ dB $= \times 10$ la potenza
  - -10 dB $= /10$ la potenza
- **dBm**: È un'unità **assoluta** di potenza. Il valore è espresso in decibel rispetto a un riferimento fisso di 1 milliwatt (mW). Quindi, 0 dBm = 1 mW, 20 dBm = 100 mW. È l'unità standard per misurare la potenza di trasmettitori e la sensibilità dei ricevitori.
- **dBi**: È un'unità che misura il **guadagno passivo** di un'antenna. Le antenne reali non irradiano energia uniformemente in tutte le direzioni come una teorica **antenna isotropica**, ma la concentrano in una direzione preferenziale. Il guadagno in dBi indica di quanti decibel il segnale in quella direzione è più forte rispetto a quello che avrebbe prodotto un'antenna isotropica.

8.10.2 Potenza Irradiata e Normative

Le agenzie governative impongono limiti sulla massima potenza che un dispositivo può irradiare per evitare interferenze.

Definition 8.10.1: EIRP (Equivalent Isotropically Radiated Power)

La potenza totale effettivamente irradiata da un'antenna nella sua direzione di massimo guadagno. Si calcola sommando la potenza in uscita dal trasmettitore (in dBm) e il guadagno dell'antenna (in dBi).

Un progettista deve quindi regolare la potenza del trasmettitore in modo che, una volta aggiunto il guadagno dell'antenna, l'EIRP totale non superi i limiti di legge.

8.10.3 Il Calcolo del Link Budget

Il link budget è un bilancio energetico che verifica se una connessione wireless è fattibile.

Definition 8.10.2: Link Budget

Un calcolo che somma tutti i guadagni (potenza del trasmettitore, guadagno delle antenne) e sottrae tutte le perdite (perdite dei cavi, attenuazione nello spazio libero, ostacoli) per determinare la potenza del segnale che arriva al ricevitore.

Questo valore deve essere confrontato con la **Sensibilità del Ricevitore (Receiver Sensitivity, RS)**, che è la minima potenza (in dBm) necessaria al ricevitore per decodificare correttamente il segnale a una data velocità. Un buon progetto richiede che la potenza ricevuta sia significativamente superiore alla sensibilità, lasciando un **Fade Margin** (es. 10-20 dB) per far fronte a imprevisti come il fading o le cattive condizioni atmosferiche.

8.10.4 Dettagli sulle Unità di Misura della Potenza

Per progettare e analizzare un sistema wireless, è fondamentale padroneggiare le unità di misura logaritmiche, che semplificano notevolmente i calcoli di guadagni e perdite.

Il Decibel (dB) Il Decibel è un'unità **relativa** utilizzata per esprimere il rapporto tra due livelli di potenza, P_1 e P_2. La sua definizione è:

$$\text{dB} = 10 \cdot \log_{10}\left(\frac{P_1}{P_2}\right)$$

Il suo grande vantaggio è che trasforma le moltiplicazioni e divisioni in addizioni e sottrazioni, rendendo i calcoli più intuitivi. Si usano spesso le seguenti regole pratiche:

- **+3 dB**: Raddoppia la potenza (x2).
- **-3 dB**: Dimezza la potenza (/2).
- **+10 dB**: Moltiplica la potenza per 10 (x10).
- **-10 dB**: Divide la potenza per 10 (/10).

Example 8.10.1 (Calcolo con i dB)
Se un segnale di 100 mW subisce una perdita di 7 dB, la potenza finale è:

$$-7 \text{ dB} = -10 \text{ dB} + 3 \text{ dB}$$

Quindi la potenza finale sarà: $(100 \text{ mW}/10) \times 2 = 20 \text{ mW}$. L'uso dei dB ha trasformato un calcolo complesso in semplici somme.

Il dBm e il dBi Poiché il dB è relativo, sono state introdotte unità assolute con un punto di riferimento fisso.

- **dBm**: È un'unità di potenza **assoluta**, dove il riferimento è 1 milliwatt (mW). Pertanto, per definizione:

  $$0 \text{ dBm} = 1 \text{ mW}$$

  È l'unità standard per indicare la potenza di un trasmettitore o la sensibilità di un ricevitore. Ad esempio, 20 dBm = 100 mW.

- **dBi**: È un'unità che misura il **guadagno** di un'antenna. Indica di quanti decibel la potenza irradiata dall'antenna in una specifica direzione è maggiore rispetto a quella che sarebbe irradiata da un'antenna isotropica teorica.

8.11 Le Antenne Wireless

Definition 8.11.1: Antenna

Un dispositivo passivo che funge da trasduttore: converte l'energia elettrica guidata da un cavo in onde elettromagnetiche che si propagano nello spazio (in trasmissione) e viceversa (in ricezione).

La forma e la dimensione di un'antenna determinano il suo **diagramma di radiazione** (*radiation pattern*), ovvero come l'energia viene distribuita nello spazio.

8.11.1 Tipi di Antenne

- **Antenna Isotropica**: Un'antenna teorica e ideale che irradia potenza in modo perfettamente uniforme in tutte le direzioni, formando una sfera. È il punto di riferimento (0 dBi) per misurare il guadagno di tutte le antenne reali.
- **Antenne Omnidirezionali**: Irradiano potenza uniformemente sul piano orizzontale, creando un diagramma di radiazione a forma di "ciambella" (toroide). Sono ideali per coprire un'area a 360° attorno a un punto centrale. L'esempio più comune è l'**antenna a dipolo** dei router Wi-Fi domestici. Un dipolo ad alto guadagno produce una ciambella più piatta e larga, coprendo una maggiore distanza orizzontale a scapito della copertura verticale.
- **Antenne Direzionali**: Concentrano la maggior parte dell'energia in una direzione specifica, formando un "lobo" di radiazione principale. Questo permette di coprire distanze maggiori con meno potenza o di creare collegamenti punto-punto sicuri. Il grado di direzionalità è misurato dal **beamwidth**, l'ampiezza angolare del lobo principale. Esempi includono le antenne **Yagi**, **Patch/Panel** e le **paraboliche**.

8.11.2 Utilizzo delle Antenne Direzionali

Le antenne direzionali sono cruciali per i collegamenti wireless a lunga distanza. Il loro uso richiede però la comprensione di due concetti chiave:

- **Line of Sight (LOS)**: Deve esistere una linea di vista ottica libera da ostacoli tra l'antenna trasmittente e quella ricevente.
- **Zona di Fresnel (Fresnel Zone)**: L'energia RF non viaggia come un raggio laser, ma occupa un volume a forma di ellissoide attorno alla linea di vista. Questa zona deve essere per la maggior parte libera da ostacoli (alberi, edifici). Un'ostruzione significativa della zona di Fresnel può interrompere il collegamento anche se la linea di vista ottica è libera. La dimensione della zona di Fresnel dipende dalla frequenza e dalla distanza del link.

8.11.3 Concetti Avanzati sulle Antenne

- **Antenne Settorizzate**: Tipiche delle torri per la telefonia mobile, sono composte da un array di più antenne direzionali, ognuna delle quali copre un "settore" (es. 120°). Questo permette di riutilizzare le stesse frequenze in settori non adiacenti, aumentando la capacità totale della cella.
- **Antenna Diversity**: Una tecnica fondamentale per combattere il fading causato dal multipath. Si utilizzano due o più antenne distanziate di una piccola frazione di lunghezza d'onda. La probabilità che tutte le antenne si trovino contemporaneamente in un punto di interferenza distruttiva è molto bassa. Il ricevitore può quindi scegliere dinamicamente il segnale proveniente dall'antenna con la qualità migliore in un dato istante. È il motivo per cui i moderni router Wi-Fi hanno più antenne.

8.12 Calcolo del Link Budget: Esempio Pratico

Il **link budget** è il calcolo fondamentale nella progettazione di un collegamento wireless, specialmente outdoor. Esso fa un bilancio di tutti i guadagni e le perdite per assicurarsi che il segnale arrivi a destinazione con una potenza sufficiente.

Definition 8.12.1: Link Budget

La differenza, misurata in dB, tra la potenza del segnale ricevuto e la sensibilità minima del ricevitore. Un link budget positivo indica che il collegamento è fattibile.

La formula generale è:

$$\text{Potenza Ricevuta (dBm)} = P_{TX} + G_{TX} - L_{TX} - L_{FS} - L_M + G_{RX} - L_{RX}$$

dove:

- P_{TX}: Potenza del trasmettitore (dBm)
- G_{TX}, G_{RX}: Guadagno delle antenne trasmittente e ricevente (dBi)
- L_{TX}, L_{RX}: Perdite nei cavi e connettori (dB)
- L_{FS}: Perdita di propagazione nello spazio libero (*Free Space Path Loss*), che dipende da distanza e frequenza (dB).
- L_M: Margine per perdite varie (es. pioggia, fogliame) (dB)

Il risultato, 'Potenza Ricevuta', deve essere maggiore della 'Sensibilità del Ricevitore' di un valore detto **Fade Margin**, che garantisce la stabilità del link.

Example 8.12.1 (Calcolo di un Link Budget)

Consideriamo un link di 10 km a 2.4 GHz.

- Potenza del Trasmettitore: +15 dBm
- Guadagno Antenna TX: +24 dBi
- Perdite cavi/connettori TX: -5.4 dB
- Path Loss per 10 km a 2.4 GHz: -120 dB
- Guadagno Antenna RX: +24 dBi
- Perdite cavi/connettori RX: -5.4 dB
- **Guadagno Totale**: $15 + 24 + 24 = +63$ dB
- **Perdita Totale**: $-5.4 - 120 - 5.4 = -130.8$ dB
- **Potenza Ricevuta**: $+63 - 130.8 = -67.8$ dBm

Se la sensibilità del ricevitore per la velocità desiderata è di -82 dBm:

$$\text{Link Budget} = -67.8 \text{ dBm} - (-82 \text{ dBm}) = +14.2 \text{ dB}$$

Il budget è positivo di 14.2 dB. Se questo valore è considerato un Fade Margin sufficiente, il collegamento è affidabile e non necessita di amplificatori o antenne a più alto guadagno.

Chapter 9

Il Controllo di Accesso al Mezzo (MAC) nelle Reti Wireless

Nei capitoli precedenti abbiamo esplorato i fondamenti fisici della trasmissione wireless. Ora ci addentriamo nel livello immediatamente superiore, il **MAC (Medium Access Control)**, per capire come i dispositivi wireless coordinano l'accesso al canale di comunicazione condiviso, un problema molto più complesso rispetto al mondo cablato.

9.1 La Sfida del MAC Wireless

In una rete wireless, tutti i dispositivi comunicano sullo stesso canale broadcast. Se più dispositivi trasmettono contemporaneamente, i loro segnali si sovrappongono e si distruggono a vicenda, generando una **collisione**. Il compito del protocollo MAC è orchestrare le trasmissioni per minimizzare le collisioni e utilizzare in modo efficiente le scarse risorse a disposizione: la capacità del canale e l'energia delle batterie.

9.1.1 L'Inadeguatezza di Ethernet nel Wireless

Il protocollo MAC per eccellenza delle reti cablate, Ethernet, si basa su un meccanismo chiamato **CSMA/CD (Carrier Sense Multiple Access with Collision Detection)**.

- **Carrier Sense**: Un dispositivo "ascolta" il canale e trasmette solo se lo percepisce come libero.
- **Collision Detection**: Un dispositivo, mentre trasmette, continua ad ascoltare. Se rileva una collisione, interrompe immediatamente la trasmissione, attende per un tempo casuale e riprova.

Questo approccio **non è praticabile** in un ambiente wireless per due ragioni fondamentali:

1. **Impossibilità della Collision Detection**: Un'antenna radio non può trasmettere e ricevere contemporaneamente sulla stessa frequenza. La potenza del segnale trasmesso è immensamente superiore a quella di qualsiasi segnale in ricezione, rendendo impossibile per il trasmettitore "accorgersi" di una collisione mentre sta avvenendo. La collisione si manifesta solo presso il **ricevitore**.
2. **Problemi del Carrier Sensing (Terminali Nascosti/Esposti)**: Il concetto di canale "libero" o "occupato" è relativo alla posizione. Un dispositivo potrebbe non sentire una trasmissione in corso perché troppo distante dal trasmettitore (problema del **terminale nascosto**), e iniziare a sua volta una trasmissione che causerà una collisione presso un ricevitore comune.

Di conseguenza, il MAC wireless deve abbandonare la *rilevazione* delle collisioni in favore della loro *prevenzione* (**Collision Avoidance**).

9.2 Classificazione dei Protocolli MAC Wireless

I protocolli MAC si possono classificare in base a come gestiscono la contesa (*contention*) per il canale.
La distinzione principale è tra:

- **Contention-Free**: L'accesso al canale è regolato in modo da eliminare a priori le collisioni. Questo si ottiene tipicamente tramite un **coordinatore centralizzato** che assegna a ogni dispositivo il proprio turno per trasmettere (es. tramite *polling* o assegnando slot di tempo fissi - *TDMA*). Garantisce prestazioni prevedibili ma può essere inefficiente.
- **Contention-Based**: Non c'è un coordinatore centrale (o il suo ruolo è limitato). I dispositivi competono per l'accesso al canale, e il protocollo si occupa di risolvere le collisioni quando avvengono. Sono protocolli più flessibili e adatti a traffico "a raffica" (*bursty*). Questi si dividono a loro volta in **deterministici** e **probabilistici** (o ad accesso casuale).

9.2.1 Protocolli ad Accesso Casuale (Random Access)

Questi protocolli sono il fondamento dei sistemi Wi-Fi moderni e si sono evoluti nel tempo per migliorare l'efficienza.

ALOHA È il protocollo più semplice: quando un nodo ha un pacchetto da inviare, lo trasmette immediatamente. Se il pacchetto collide con un altro, il mittente non riceve la conferma (ACK) dal destinatario, attende un tempo casuale e ritrasmette.

- **Vulnerabilità**: Una trasmissione è vulnerabile per un tempo pari al doppio della durata del frame, perché una collisione si verifica se un altro nodo inizia a trasmettere in qualsiasi momento durante questo intervallo.
- **Efficienza**: Molto bassa, con un throughput massimo teorico di circa il 18%.

Slotted ALOHA Un miglioramento di ALOHA in cui il tempo è suddiviso in slot discreti di durata pari a un frame. Le stazioni possono iniziare a trasmettere solo all'inizio di uno slot.

- **Vulnerabilità**: La finestra di vulnerabilità si dimezza, poiché le collisioni possono avvenire solo se due stazioni scelgono lo stesso slot.
- **Efficienza**: Il throughput massimo raddoppia, raggiungendo circa il 37%.

CSMA (Carrier Sense Multiple Access) Aggiunge il principio "ascolta prima di parlare". Una stazione trasmette solo se sente il canale libero. Questo riduce drasticamente il numero di collisioni, ma non le elimina a causa del ritardo di propagazione.

- **Vulnerabilità**: La finestra di vulnerabilità è legata al tempo di propagazione del segnale.
- **Efficienza**: Molto più alta di ALOHA a bassi carichi, ma le prestazioni crollano a carichi elevati, quando il canale è quasi sempre occupato e le collisioni tra stazioni in attesa diventano frequenti.

9.3 La Dimensione Spaziale: Terminali Nascosti ed Esposti

Gli algoritmi visti finora si concentrano sul dominio del tempo, ma nel wireless è fondamentale gestire anche il dominio dello spazio.

Definition 9.3.1: Problema del Terminale Nascosto (Hidden Terminal)

Si verifica quando un nodo A è visibile a un ricevitore C, ma non a un altro nodo B, che è anch'esso visibile a C. Se A sta trasmettendo a C, B non lo sente, percepisce il canale come libero e potrebbe iniziare una sua trasmissione verso C, causando una collisione che corrompe entrambi i messaggi.

Definition 9.3.2: Problema del Terminale Esposto (Exposed Terminal)

È la situazione opposta. Il nodo B sta trasmettendo al nodo A. Il nodo C, vicino a B ma lontano da A, vorrebbe trasmettere a D. C sente la trasmissione di B, percepisce il canale come occupato e si astiene dal trasmettere, anche se la sua trasmissione non avrebbe interferito con la ricezione di A. È uno spreco di capacità del canale.

9.3.1 La Soluzione: l'Handshake RTS/CTS

Per risolvere in particolare il problema del terminale nascosto, sono stati introdotti meccanismi di **Collision Avoidance** basati su un handshake a quattro vie.

Definition 9.3.3: Handshake RTS/CTS (Request to Send / Clear to Send)

Un meccanismo in cui un trasmettitore, prima di inviare un lungo pacchetto di dati, "prenota" il canale inviando un breve pacchetto di controllo **RTS** (Richiesta di Invio). Il ricevitore, se è pronto, risponde con un pacchetto **CTS** (Pronto a Ricevere).

Questo semplice scambio risolve il problema:

- Il pacchetto CTS viene sentito da tutti i nodi vicini al **ricevitore** (incluso il terminale nascosto B).
- Il messaggio CTS contiene la durata della trasmissione che sta per avvenire.
- Qualsiasi nodo che sente il CTS sa che deve rimanere in silenzio per quella durata, anche se non sente l'RTS o il trasmettitore stesso.

In questo modo, lo spazio attorno al ricevitore viene "silenziato" per la durata della trasmissione, evitando collisioni da parte di terminali nascosti. L'uso di RTS/CTS introduce un overhead, e per questo motivo viene tipicamente attivato solo per pacchetti di dati che superano una certa soglia di dimensione.

9.3.2 Limiti di RTS/CTS e Variazioni sul Tema

L'handshake RTS/CTS, sebbene efficace contro il problema del terminale nascosto, non è una panacea e introduce le sue complessità.

Vulnerabilità e Overhead L'uso di RTS/CTS aggiunge un sovraccarico significativo: prima di ogni pacchetto di dati, è necessario scambiare due pacchetti di controllo, consumando tempo e capacità del canale. Per questo motivo, il suo utilizzo è tipicamente limitato ai pacchetti di grandi dimensioni. Inoltre, non è immune da attacchi: un malintenzionato potrebbe inondare la rete di pacchetti CTS falsi (*CTS flooding*), silenziando di fatto tutti i nodi nell'area e causando un attacco Denial of Service.

Evoluzioni del Protocollo: MACA, MACAW, FAMA L'idea di base del Collision Avoidance si è evoluta attraverso una serie di protocolli di ricerca:

- **MACA (Multiple Access with Collision Avoidance)**: Formalizza l'uso di RTS/CTS, eliminando del tutto il carrier sensing e affidandosi unicamente all'handshake per gestire la contesa, che di fatto avviene attorno al ricevitore.
- **MACAW**: Migliora MACA introducendo un pacchetto **ACK** dopo la ricezione dei dati. Questo crea un handshake a 4 vie (**RTS-CTS-DATA-ACK**) che non solo previene le collisioni, ma fornisce anche un meccanismo di riscontro per la trasmissione affidabile a livello MAC.
- **FAMA**: Reintroduce il **carrier sensing** prima dell'invio dell'RTS. Combina così i vantaggi di entrambi gli approcci: si "ascolta" il canale per evitare collisioni ovvie e si usa l'handshake per risolvere i problemi spaziali più complessi come i terminali nascosti.

9.4 Il MAC dello Standard IEEE 802.11 (Wi-Fi)

Lo standard IEEE 802.11, su cui si basano tutte le reti Wi-Fi, definisce un'architettura MAC ibrida e sofisticata, progettata per operare in due modalità diverse.

9.4.1 Le Due Funzioni di Coordinamento: DCF e PCF

Il MAC 802.11 è composto da due meccanismi che possono coesistere, alternandosi nel tempo all'interno di una **super-frame**:

1. **DCF (Distributed Coordination Function)**: È la modalità fondamentale, obbligatoria e più comune. È un meccanismo **distribuito e basato sulla contesa** (*contention-based*). Utilizza l'algoritmo **CSMA/CA** (Carrier Sense Multiple Access with Collision Avoidance) per gestire l'accesso. Offre un servizio *best-effort* senza garanzie sulla qualità del servizio (QoS). È la modalità usata nelle reti domestiche e nelle configurazioni *ad-hoc*.
2. **PCF (Point Coordination Function)**: È una modalità opzionale e **centralizzata, priva di contesa** (*contention-free*). Richiede un coordinatore centrale, l'**Access Point (AP)**, che gestisce l'accesso al canale tramite *polling*: interroga a turno le stazioni, dando loro il permesso di trasmettere. Questo elimina le collisioni e permette di offrire garanzie di base sulla banda e sul ritardo (soft QoS).

9.4.2 Gli Interframe Spaces (IFS): la Base della Priorità

Per orchestrare l'alternanza tra DCF, PCF e le risposte immediate (come gli ACK), lo standard 802.11 definisce degli intervalli di tempo di attesa obbligatori, chiamati **Interframe Spaces**, di diversa durata. La regola è semplice: chi deve attendere per meno tempo ha una priorità più alta.

- **SIFS (Short IFS)**: L'intervallo più breve. Ha la priorità più alta ed è usato per operazioni che devono avvenire immediatamente dopo una trasmissione, come l'invio di un CTS in risposta a un RTS, o di un ACK in risposta a un pacchetto di dati.
- **PIFS (PCF IFS)**: Intervallo di media durata. Viene usato dall'Access Point. Se il canale è libero per un tempo pari a un PIFS, l'AP ha il diritto di prenderne il controllo per avviare una fase PCF (contention-free).
- **DIFS (DCF IFS)**: L'intervallo più lungo. Una stazione che deve inviare un nuovo pacchetto di dati in modalità DCF deve attendere che il canale sia libero per almeno un tempo pari a un DIFS.

Questa gerarchia ($SIFS < PIFS < DIFS$) garantisce che un ACK abbia sempre la precedenza su un poll del PCF, che a sua volta ha la precedenza su una nuova trasmissione dati in DCF.

9.4.3 La Distributed Coordination Function (DCF) in Dettaglio

La DCF è il cuore del Wi-Fi e si basa sul CSMA/CA con un meccanismo di backoff per evitare le collisioni.

La Procedura di Backoff Quando una stazione vuole trasmettere e rileva che il canale è libero da almeno un DIFS, non trasmette immediatamente. Per evitare che più stazioni in attesa trasmettano contemporaneamente, avvia una procedura di attesa casuale:

1. **Scelta del contatore**: La stazione sceglie un numero intero casuale, il *backoff counter*, in un intervallo $[0, CW]$, dove CW è la **Contention Window** (finestra di contesa).
2. **Conto alla rovescia**: La stazione continua a monitorare il canale. Per ogni *slot time* in cui il canale rimane libero, decrementa il suo contatore. Se il canale diventa occupato, il contatore viene "congelato" e riprenderà il conto alla rovescia non appena il canale sarà di nuovo libero per un DIFS.
3. **Trasmissione**: La stazione trasmette il suo frame solo quando il suo contatore di backoff ha raggiunto lo zero.

Binary Exponential Backoff (BEB) Per adattarsi al livello di congestione della rete, la dimensione della Contention Window non è fissa.

- All'inizio (o dopo una trasmissione riuscita), la CW è impostata al suo valore minimo (CW_{min}).
- A ogni collisione (rilevata dalla mancata ricezione di un ACK), la stazione **raddoppia** la dimensione della CW, fino a un valore massimo (CW_{max}).

Questo meccanismo, noto come **Binary Exponential Backoff**, aumenta esponenzialmente l'intervallo da cui viene estratto il contatore casuale, "sparpagliando" maggiormente nel tempo i tentativi di ritrasmissione e riducendo la probabilità di ulteriori collisioni.