Reti di Calcolatori

Appunti delle Lezioni di Reti di Calcolatori

Anno Accademico: 2024/25

Giacomo Sturm

Dipartimento di Ingegneria Civile, Informatica e delle Tecnologie Aeronautiche
Università degli Studi "Roma Tre"

Sorgente del *file* LaTeX disponibile al seguente link:
https://github.com/00Darxk/Reti-di-Calcolatori/

Indice

1 Introduzione

Una qualsiasi interconnessione di calcolatori può rappresentare una rete di calcolatori, ma in base alla distanza reciproca tra questi componenti si tratta di reti differenti. Convenzionalmente si considerano reti di calcolatori, sistemi di calcolatori interconnessi ad una distanza superiore ai 50 cm. Una distanza minore, fino ai 5 cm, generalmente interessa componenti dello stesso computer, sulla stessa scheda madre, connesse tra di loro; mentre una distanza inferiore ai 5 cm rappresenta componenti sullo stesso chip. Inoltre le reti considerate possono essere ulteriormente divise in base alla distanza dei loro elementi:

- Se hanno una distanza minore di 5 km, si tratta di risorse connesse sulla stessa rete o edificio, o su edifici vicini. Questo tipo di rete si chiama *Local Area Network* (LAN);
- Se hanno una distanza superiore ai 5 km, si tratta di risorse connesse su una vasta area geografica. Questo tipo di rete si chiama *Wide Area Network* (WAN).

Tra questi due livelli possono essere presenti anche tecnologie molto diverse tra di loro, queste tecnologie vengono identificate da acronimi da cui è possibile ricavare lo scopo della tecnologia, senza tuttavia conoscere il suo funzionamento.

Una connessione tra componenti di una rete coinvolge sempre uno scambio di informazioni, tramite uno scambio di messaggi in serie. Gli elementi della rete effettuano degli accessi ad essa apparentemente in parallelo e simultanei, per poter comunicare tra di loro. Mentre su componenti sulla stessa macchina o sullo stesso chip avvengono tramite accessi ad una memoria condivisa.

Le connessioni tra componenti di una rete avvengono su uno strato fisico, quindi attraverso diversi mezzi trasmissivi, i quali non verranno analizzati approfonditamente a questo livello di astrazione. Tra i più comuni mezzi trasmissivi abbiamo cavi in fibra ottica, o in rame, ed onde radio.

1.1 Commutazione

Una rete di calcolatori può essere rappresentata come un grafo composto da vari nodi, per realizzare tutte le possibili coppie di calcolatori che potrebbero comunicare tra di loro attraverso la rete. Ma se venissero collegati individualmente tutte le possibili coppie di calcolatori necessiterebbe di infrastrutture massicce, poiché il numero dei possibili percorsi aumenta quadraticamente rispetto all'aumento dei calcolatori della rete. Infatti avendo n, tutte le possibili combinazioni tra questi calcolatori sono $n(n-2)/2$, nel caso ognuna di queste coppie corrisponda ad una connessione differente, il costo di costruzione e gestione della rete sarebbe eccessivo.

Per risolvere questo problema e diminuire il numero totale di connessioni nella rete si utilizza il meccanismo della commutazione. Questo termine risale alla telefonia, dove sorse lo stesso problema, risolto introducendo centralini intermedi dove si potevano collegare diverse area telefoniche contenenti i telefoni che tentavano di comunicare. In questo modo si può drasticamente diminuire il numero di connessioni individuali nella rete, e non bisogna integrare un numero elevato di connessioni all'aggiunta di un singolo elemento. Si indica quindi con commutazione di circuito questo meccanismo di creare una connessione fisica tra due calcolatori, connettendo diverse zone della rete attraverso nodi intermedi.

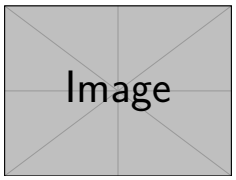

Fig. 1: Commutazione di Circuito

L'interazione tra computer consiste intrinsecamente da grandi quantità di dati trasmessi velocemente a grandi distanze, per cui hanno bisogno di infrastrutture dedicate massicce, si preferisce quindi questo sistema di nodi intermedi, nonostante non consenta di soddisfare contemporaneamente tutte le coppie di calcolatori.

Poiché questa grande quantità di dati deve attraversare la rete velocemente, si utilizza una diversa tecnica di comunicazione a livello dei singoli messaggi, dividendoli in pacchetti da spedire separatamene. Nella commutazione a datagramma questi pacchetti vengono spediti su linee anche diverse e si mescolano a tutti i pacchetti che attraversano quel percorso. Le linee non sono quindi ad uso esclusivo di una singola connessione.

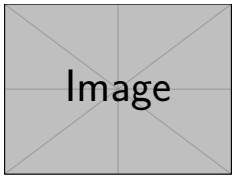

Fig. 2: Commutazione di Pacchetto a Datagramma

Ma per ricomporre il messaggio originale bisogna combinare questi pacchetti nello stesso ordine in cui sono stati separati, sono necessari dati aggiuntivi per poter riconoscere il loro ordine, perso durante la trasmissione. La distanza attraversata da ciascun pacchetto infatti non è garantito sia uguale.

Esiste un altro tipo di commutazione a circuito virtuale, dove i pacchetti vengono inviati sullo stesso percorso sequenzialmente, ed ogni linea può essere condivisa da un altro circuito virtuale, quindi non sono ad uso esclusivo.

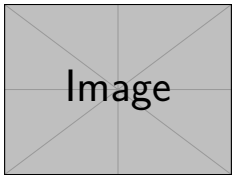

Fig. 3: Commutazione di Pacchetto a Circuito Virtuale

In questo caso invece è necessario un meccanismo per poter distinguere tra di loro questi circuiti virtuali sulla stessa linea fisica.

Si è risolto tramite la commutazione di pacchetto l'esclusività delle linee della rete, introdotta dal modello a commutazione di circuito.

La rete *internet* moderna utilizza la commutazione a datagramma, per motivi economici e gestionali. Altrimenti sarebbe necessario un gestore della rete che deve trovare un percorso ed attribuirlo ad una coppia ad ogni tentativo di connessione. Data la complessità della rete moderna, lasciare che i pacchetti trovino il percorso autonomamente è la scelta più efficiente. Per realizzare una commutazione a datagramma, piccole aree geografiche diverse vengono coperte da *Internet Service Provider* (ISP) differenti che possono comunicare solamente con altri ISP adiacenti. Quindi all'invio di un pacchetto, se il destinatario non è nella stessa zona dell'ISP corrente, questo lo invia ad un ISP adiacente che crede possa contenere il destinatario, così anche per la ricezione da un altro ISP. In caso il destinatario sia nella zona dell'ISP corrente, questo lo trasmette a lui. Accordi possono essere stipulati da chiunque a chiunque, un ISP ha sempre la necessità di trasmettere i pacchetti attraverso la rete.

In certi casi l'ISP può gestire la rete a circuito virtuale, se sia il destinatario che il mittente siano coperti dal singolo ISP.

1.2 Velocità

Nelle reti LAN e WAN si possono trasmettere dati a velocità diverse:

- LAN: velocità tra 10 ai 100 Mb/s;
- WAN: velocità tra 64 Kb/s ai 200-400 Mb/s.

Le reti WAN presentano hanno molti livelli di retrocompatibilità mantenuti, per cui si possono trasmettere dati a velocità minori di una rete LAN. In generale sono sempre richieste reti a velocità di trasmissione elevata, e connessioni ad alta velocità.

Data una rete si può definire la velocità in due modi differenti. Se si considera il tempo in cui il primo bit del messaggio arriva a destinazione. Le connessioni ad alta velocità vengono realizzate in linee a fibra ottica, per cui i bit vengono inviati come impulsi di luce, e viaggiano ad una velocità costante, quindi per ogni rete la velocità di trasmissione di un singolo bit è la stessa. Si definisce quindi il tempo di ritardo o *delay*, il tempo per trasmettere un singolo bit, alla velocità della luce, sulla rete e dipende interamente dalla distanza.

Un pacchetto non viene rappresentato da un singolo bit, per cui non possono essere trasmessi alla stessa velocità, si definisce banda la quantità di bit trasmessi contemporaneamente sulla linea. Questa si chiama banda, e generalmente è sempre possibile comprare più banda in modo relativamente facile, ma è molto difficile comprare meno *delay*.

1.3 Gestione delle Risorse

La rete è essenzialmente un insieme di risorse interconnesse tra di loro e dalla teoria dei sistemi operativi, il loro controllo può essere descritto da varie attività:

- Verifica dei diritti d'accesso;
- Sequenziamento degli accessi alla risorsa;
- Esecuzione delle operazioni disponibili.

Ad ogni risorsa vengono assegnato almeno un gestore, di numero variabile in base al tipo di gestione. Le modalità di gestione delle risorse sono varie, si dividono in gestione autocratica e multilaterale. Nella gestione autocratica ogni risorsa ha un unico gestore associato ed univoco. Nella gestione multilaterale per ogni risorsa può esserci più di un gestore, si possono identificare quindi tre sottotipi di questa gestione:

- Gestione partizionata, dove attività di gestione viene effettuata da un singolo processo;
- Gestione successiva, dove tutte le attività di gestione vengono effettuate a turni da più processi;
- Gestione replicata, dove tutti i gestori partecipano a ciascuna attività, se ogni gestore ha peso decisionale uguale allora si tratta di gestione democratica.

La gestione replicata fornisce una forte resistenza ai guasti per un numero elevato di gestori che partecipano a ciascuna istanza di una attività, con alto grado di uguaglianza nella responsabilità di gestione. Sono abbastanza diffusi meccanismi di elezione per la scelta dei gestori.

2 Modello ISO-OSI

Per il funzionamento della rete gli standard sono strettamente necessari, altrimenti non sarebbe possibile una comunicazione tra un mittente ed un destinatario qualsiasi, alcuni di questi standard vengono imposti dalla case costruttrici, altri vengono definiti da organizzazioni internazionali, nell'ambito informatico o delle telecomunicazioni. Alcune di queste associazioni come IETF sono indipendenti da stati nazionali, dove varie aziende o istituzioni propongono modifiche di vecchi standard o introduzione e definizione di nuovi.

Il modello ISO-OSI rappresenta un importante strumento di classificazione nel modo delle reti. Venne realizzato in parte e sostanzialmente dismesso, ma nonostante questo viene utilizzato a livello globale.

Questo modello si basa sull'architettura stratificata di *hardware* o software, dove partendo da un nucleo centrale il sistema viene diviso in livelli o strati indipendenti dal livello inferiore, ed uno strato fornisce servizi solamente allo strato immediatamente superiore. Avanzando da uno strato al superiore i servizi vengono mostrati in modo sempre più astratto ed il sistema aumenta progressivamente di utilità. Per la sua utilità questo tipo di architettura stratificata permane molti campi dell'informatica.

La rete viene divisa in 7 livelli numerati dal basso verso l'alto, il livello indica la funzione delle tecnologie che vi appartengono e questo fornisce uno strumento di classificazione per analizzarle senza dover conoscere i loro meccanismi interni:

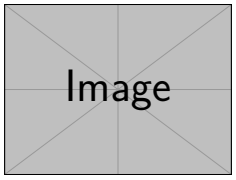

Fig. 4: Standard ISO-OSI ed Internet Protocol Suite

Ogni strato rappresenta un diverso livello di astrazione ed offrono funzioni ben definite. Poiché ognuno di questi strati è indipendente dal livello inferiore, viene minimizzato lo scambio di informazioni tra strati. Il numero dei livelli venne scelto in base alle funzioni distinte di una rete da descrivere e dalla realizzabilità.

All'interno di ogni strato si possono individuare diverse "entità", *hardware* o software dove sono contenuti i protocolli di quel livello. Per offrire servizi allo strato superiore, è presente un punto logico chiamato *Service Access Point* (sap) al quale può accedere il livello superiore. L'unico punto di contatto tra livelli e quello inferiore è la loro interfaccia. Un protocollo è un linguaggio utilizzato da entità dello stesso livello, quindi entità di uno stesso strato possono comunicare con le adiacenti tramite protocolli e con superiori tramite sap, ed inferiori tramite interfacce:

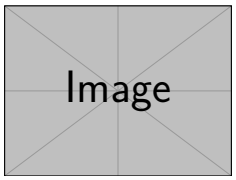

Fig. 5: Comunicazione tra Entità della Rete

I protocolli su uno stesso strato possono comunicare con altre entità dello stesso livello, e possono inviare indicazioni o conferme a richieste di entità utenti del servizio del livello superiore.

Secondo questo modello i pacchetti sono contenuti in altri pacchetti, destinati a livelli inferiori, per cui quando vengono ricevuti da un livello $n-1$, viene letto il pacchetto di livello $n-1$ ed estratto il pacchetto di livello n contenuto ed inviato all'entità di livello superiore n.

I dati generati da un protocollo di livello n sono detti n-pdu, *Protocol Data Unit*, composti da un *header* indirizzato all'entità di livello n ed una payload, contente una $n+1$-pdu, destinata al livello superiore. Per cui all'aumento dei livelli aumenta l'*overhead*:

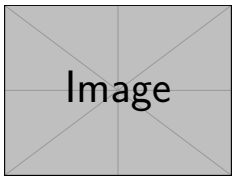

Fig. 6: Pila ISO-OSI e Rispettive pdu

2.1 Livelli

I diversi livelli di questo modello presentano le seguenti funzioni:

1. Il primo strato della pila ISO-OSI rappresenta lo strato fisico, che si interfaccia direttamente con il mezzo trasmissivo della rete e quindi rappresenta il livello di natura fisica della trasmissione. Offre al livello superiore una comunicazione indipendente dal mezzo trasmissivo. Fornisce allo strato di collegamento servizi di trasmissione di bit a tra sistemi adiacenti, consegna in sequenza di bit o notifiche di malfunzionamenti.

2. Il secondo livello rappresenta lo strato data-link per risolvere eventuali malfunzionamenti dello strato fisico, rilevando e correggendo errori, tramite algoritmi di correzione, come i bit di parità. Offre allo strato superiore la possibilità di trasmettere pdu a sistemi adiacenti utilizzando due code nelle due direzioni.

3. Il terzo livello è lo strato di rete e conosce la topologia completa della rete, per effettuare operazioni di instradamento. Contiene i protocolli come IPv4, progressivamente sostituito da IPv6, e permette il trasferimento di pdu da estremo ad estremo. Inoltre permette la commutazione di circuito o di pacchetto a datagramma e a circuito virtuale.

4. Il quarto livello di trasporto, divide il messaggio in pacchetti, prova a colmare fluttuazioni della qualità del servizio dello strato di rete in modo trasparente rispetto agli strati superiori. In caso manchino dei pacchetti prova a recuperarli attraverso algoritmi di correzione, è il primo strato che risiede solamente nei terminali. Offre allo strato superiore la possibilità di instaurare una connessione e gestione della stessa, una trasmissione affidabile, ed il rilascio della connessione.

5. Il quinto livello di sessione sincronizza e struttura il dialogo tra due processi.

6. Il sesto livello di presentazione permette uno scambio di messaggi indipendentemente dalla sintassi della trasmissione.

7. Il settimo livello di applicazione offre un mezzo per accedere alla rete tramite un processo, interfacciando l'utente alla rete.

Per tutti i livelli superiori a quello fisico si possono definire due modalità operative ed associati servizi e protocolli, connessi e non connessi. Nei servizi o protocolli connessi, si instaura una connessione o dialogo tra le entità, e termina solamente dopo convenevoli finali. La modalità non connessa non ha bisogno di una connessione costante tra le due entità, viene instaurata senza un dialogo da una delle entità senza una terminazione.

Nella prima modalità l'entità non ha bisogno di ascoltare tutto il traffico per determinare quali pdu sono indirizzati alla stessa, ma necessita di una connessione continua anche se vengono trasmessi una piccola quantità di dati. Mentre nella seconda modalità possono essere inviate pdu indipendente dalla connessione e dalla distanza temporale tra le due, ma le entità che offrono questo servizio o protocollo devono costantemente analizzare il traffico per individuare le pdu a loro indirizzate. I protocolli non connessi sono quindi più efficienti, ma mancano di affidabilità, poiché manca una conferma di ricezione dei dati come nei protocolli connessi. quest'ultimi sono quindi più affidabili, ma meno efficienti, poiché dopo aver instaurato il dialogo non è possibile terminarlo preventivamente, e ciò può causare uno spreco di risorse.

In un servizio connesso sono presenti primitive per instaurare una connessione, inviare messaggi e una conferma di ricezione o ricevere messaggi, specificare l'indirizzo o nome della connessione ed abbattere la connessione. Mentre per servizi non connessi sono presenti solo primitive per inviare messaggi separatamene. Nelle reti LAN sono disponibili servizi connessi, solamente sul quarto strato, mentre nelle reti WAN è possibile siano offerti anche nel primo strato.

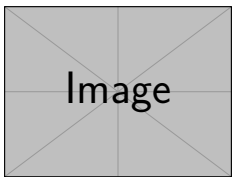

Fig. 7: Gli Strati ISO-OSI agli Estremi e nei Sistemi Intermedi

I primi tre livelli della pila ISO-OSI sono presenti su ogni nodo della rete non solo sui calcolatori, poiché rappresentano i livelli di trasmissione dei pacchetti, necessari anche nei nodi intermedi per poter trasmettere i pacchetti.

Nei protocolli di livello 2, 3 e 4 si utilizzano meccanismi di riscontro o *acknowledgment* e tecniche di controllo a finestra, a riga indice e puntatore in avanti per correggere eventuali errori nei pacchetti.

Gli ultimi tre strati della pila si interfacciano con le applicazioni e lavorano generalmente in parallelo invece che in serie come il resto della pila.

Una singola connessione di livello n può essere sfruttata da più connessioni di livello $n+1$, interne in modo che gli n-pdu contengono entrambe le $n+1$-pdu delle due connessioni. Può essere il caso di connessioni tra più processi diversi sulle stesse due macchine, dove una singola connessione tra queste due macchine contenga numerose connessioni processo-processo tra le due.

Inoltre una singola connessione $n+1$ può utilizzare più di una connessione di livello n, per parallelizzare la trasmissione e velocizzarla partizionando i dati da inviare, oppure per implementare una resistenza ai guasti. La connessione $n+1$ utilizza più canali di comunicazione di livello n non in competizione.

Una singola connessione di livello $n+1$ nel tempo, può utilizzare più di una connessione di livello inferiore; è possibile che il terminale si sposta durante la trasmissione e si aggancia a reti diverse da quella iniziale, senza interrompere la connessione. Nello stesso caso, una stessa connessione di livello n continua nel tempo può essere utilizzata da diverse connessioni di livello $n+1$. La connessione originale dell'esempio precedente vede uscire il primo terminale e quindi la prima connessione $n+1$ per poi vedere accedere un altro terminale ed un'altra connessione $n+1$.

3 Livello 2: Standard IEEE 802

Lo standard IEEE 802 riguarda i primi due livelli del modello ISO-OSI, ovvero il livello fisico, ed il livello data link, per permettere la comunicazione di calcolatori sulla stessa rete locale, personale o metropolitana:

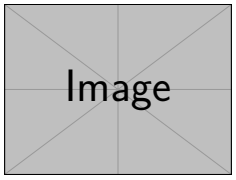

Fig. 8: Standard IEEE 802

Le tecnologie definite in base a questo standard quindi hanno come obiettivo la trasmissione e la rivelazione o correzione di bit attraverso un mezzo trasmissivo. Si occupano della connessione e quindi comunicazione tra macchine adiacenti, per una qualche definizione di adiacenza. Altri protocolli e standard noti sono l'IPv4 ed IPv6, protocolli di routing di livello tre, protocolli TCP ed UDP di livello quattro, ed il protocollo HTTP di livello sette, ma si occupa anche da solo delle funzioni dei livelli 5 e 6.

Questi standard vengono realizzati dall'organizzazione IEEE, *Institute of Electrical and Electronics Engineers*, organizzazione indipendente da stati sovrani. Il progetto IEEE 802 venne definito con l'obiettivo di realizzare una serie di standard di livello fisico e data-link per permettere la comunicazione di calcolatori sulla stessa rete locale, LAN, personale, PAN, o rete metropolitana, MAN, di grandezza intermedia tra le reti LAN e WAN. Ha avuto successo sopratutto per le reti LAN e MAN, ma per le reti personali si utilizza uno standard diverso basato sul bluetooth. Questi standard riguardano tecnologie con pacchetti di lunghezza variabile.

Le specifiche tecnologie vengono individuate tramite una notazione puntata, con 802.x, dove x rappresenta un numero, ed identifica la tecnologia. I numeri precedenti al punto individuano lo standard dove è stata introdotta questa tecnologia. Ma le tecnologie non rimangono invariate nel tempo, per cui si possono assegnare delle lettere dopo il numero per specificare la versione o tipo di quella specifica tecnologia.

3.1 802.2: Sottolivello MAC ed LLC

Lo standard IEEE 802.2 divide il livello due in due sottolivelli: *Logical Link Control* (LLC) e *Media Access Control* (MAC), questi gestiscono due tipologie diverse di pacchetti. Per le diverse tecnologie dello standard, il livello MAC è diverso, mentre il livello LLC è comune a tutti. Il sottolivello MAC è specifico per ogni tipo di LAN, si suppone che tutti i calcolatori che devono comunicare siano nella stessa LAN. Data questa ipotesi il sottolivello MAC risolve il problema di determinare il destinatario in ricezione, e di verificare la disponibilità della LAN in trasmissione, in caso la LAN sia a singolo canale condiviso. Quindi bisogna evitare che il canale sia utilizzato da più utenti.

3.1.1 Sottolivello MAC

Poiché il canale è condiviso, tutti gli utenti possono vedere i pacchetti inviati, sono quindi necessari protocolli di sicurezza e cifratura per impedire che sia possibile a chiunque connesso alla rete di leggere il contenuto dei pacchetti. Tecniche che non verranno trattate in questo corso. Inoltre si utilizza un canale condiviso in modo che in caso di malfunzionamento fisico, una sola connessione verrebbe compromessa e non l'intera rete.

Per determinare il destinatario di un pacchetto nella MAC pdu è presente un campo per definire il tipo di trasmissione:

- Punto a Punto: da un calcolatore ad un altro nella LAN;
- Punto a Gruppo: da un calcolatore a diversi altri nella LAN;
- *Broadcast*: a tutti gli utenti connessi alla LAN.

Per permettere di identificare univocamente un unico elemento nella rete, gli indirizzi MAC devono essere univoci nella rete considerata. Dato che è possibile connettersi ad una LAN dall'esterno senza conoscere gli indirizzi MAC utilizzati, servirebbe un gestore di rete per assegnarli ad ogni nuova connessione, ma questo è un approccio inefficiente. Si utilizzano quindi indirizzi MAC univoci a livello mondiale, in questo modo nell'intera rete esisteranno solo indirizzi MAC differenti. Questa condizione vale anche su VPN o LPN, inoltre se su una stessa macchina vengono simulate diverse macchine virtuali, ognuna di esse dovrà avere un indirizzo MAC differente nella LAN che utilizzano per comunicare tra di loro. Se due macchine avessero lo stesso MAC, riceverebbero gli stessi pacchetti, e verrebbero riconosciuti da entrambe le macchine come propri.

La MAC pdu è composta da diversi campi, che possono variare in base alla tecnologia con l'aggiunta di campi specifici. Ma per ogni tecnologia aderente allo standard IEEE 802 sono presenti sicuramente questi quattro campi per la MAC pdu:

- MAC dsap (*Destination Service Access Point*): indirizzo di destinazione;
- MAC ssap (*Source/Send Service Access Point*): indirizzo di partenza;
- Info: LLC pdu;
- FCS (*Frame Check Sequence*): per identificare e correggere eventuali errori.

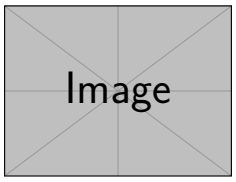

Fig. 9: Struttura MAC pdu

Per ottenere l'indirizzo del destinatario, si utilizzano protocolli di acquisizione descritti in seguito.

Gli indirizzi MAC sono composti da 6 byte, in base allo standard EUI-48, *Extended Unique Identifier*, per indirizzi a 48 bit, ma esiste anche uno standard a 64 bit non utilizzato. Questi byte vengono rappresentati in forma esadecimale, separati da due punti o trattini. I primi tre byte dell'indirizzo MAC vengono assegnati al costruttore e rappresentano gli OUI *Organization Unique Identifier*, gli ultimi 3 byte vengono scelti dal costruttore. Per cui dato un indirizzo MAC, è sempre possibile determinare il costruttore della macchina a cui appartiene.

Esistono diversi tipi di indirizzi MAC, in base al valore di determinati bit:

- *Unicast*: indirizzi che individuano le singole schede di rete dei calcolatori; se l'ultimo bit del primo byte ha valore zero;
- *Multicast*: indirizzi che identificano gruppi di schede di rete; se l'ultimo bit del primo byte ha valore uno;
- *Broadcast*: identificano tutte le schede di rete; se l'indirizzo è `FF:FF:FF:FF:FF:FF`.

Inoltre è possibile assegnare indirizzi MAC non unici a livello mondiale, se il penultimo bit del primo byte vale uno. In questo modo è possibile gestire localmente indirizzi MAC nella stessa LAN.

Per risolvere i conflitti in trasmissione si utilizzano nelle rete WiFi degli algoritmi distribuiti sulle singole macchine in contemporanea, che collaborano per determinare a chi abilitare gli accessi alla rete.

3.1.2 Sottolivello LLC

Il sottolivello LLC consegna al livello MAC un pacchetto da spedire, questo pacchetto è uguale per ogni tecnologia aderente allo standard e presenta campi analoghi al sottolivello MAC. I campi LLC-dsap/ssap individuano gli indirizzi LLC del mittente e destinatario, il campo *control* contiene il tipo di pdu per gestire diverse tipologie di pacchetto, e l'ultimo campo contiene la pdu di terzo livello:

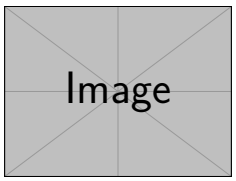

Fig. 10: Struttura LLC pdu

Gli indirizzi LLC non individuano macchine come gli indirizzi MAC, ma vengono utilizzati per identificare i protocolli di livello 3 a cui sono indirizzati i pacchetti. Consentono la convivenza di diversi protocolli di livello 3 sulla stessa macchina e sulla stessa LAN, dove sono presenti diverse pile protocollari con obiettivi e versionatura diversa.

Gli indirizzi LLC vengono attribuiti dall'IEEE solo a protocolli "ufficialmente" standard, ma questa è una classificazione che ignora molti dei protocolli più utilizzati a livello globale come TPC/IP, il protocollo più utilizzato al mondo. Vengono identificati da un byte in esadecimale, se non sono protocolli standard il loro indirizzo è `AA`, ed il pacchetto subisce una variazione con la snap-pdu, *SubNet Access Point*, dopo il campo control di 5 byte per identificare il protocollo. Questo rappresenta un ulteriore livello di *overhead*, già elevato per il modello ISO-OSI, per i pacchetti.

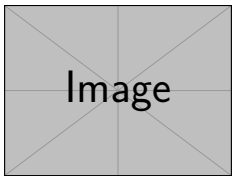

Fig. 11: MAC pdu ed LLC pdu

3.2 802.3: Ethernet

La prima tecnologia ethernet nacque nel 1970 dal consorzio DIX, dalle iniziali delle tre più grandi case produttrici informatiche dell'epoca: Digital, Intel e Xerox. In seguito venne revisionato negli anni '80 due volte, nel 1989 lo standard IEEE 802.3 diventa lo standard ISO 8802.3, e negli anni '90 ebbe talmente successo sulle reti LAN e WAN che divenne essenzialmente lo standard di tutte le trasmissioni su filo, completamente retrocompatibile. Nel tempo ha usato diversi mezzi trasmissivi, da cavi di rame intrecciati a coppie utp e stp *(Un)Shielded Twisted Pairs*, alla fibra ottica moderna. I cavi di rame venivano avvolti da una isolante dielettrico ed una schermatura esterna (stp), dove erano presenti quattro coppie di cavi di rame intrecciati. Questo permetteva connessioni punto-punto bidirezionali e contemporanee.

Dagli anni '90 in poi la banda massima possibile è aumentata fino ad un massimo di 100-400 Gb/s.

Il formato del pacchetto ethernet è rimasto sostanzialmente invariato nel tempo, nonostante le sue revisioni, nello standard IEEE 802.3 presenta i seguenti campi:

- Preambolo (56 bit): veniva utilizzato per sincronizzare in fase i pacchetti, modalità di trasmissione non più in uso;
- SFD *Start Frame Delimiter* (8 bit): indica l'inizio del pacchetto;
- Indirizzi MAC di sorgente e destinazione (96 bit);
- Lunghezza del campo dati (16 bit);
- Dati: di lunghezza variabile con un massimo di 1500 Byte, contiene una LLC pdu;
- Pad: eventuale riempimento, da 0 a 46 Byte, la somma con il campo dati deve essere compresa tra 46 e 1500 Byte;

- FCS *Frame Check Sequence* (32 bit): contiene il valore del codice di ridondanza ciclica (CRC) calcolato.

Non è presente invece un delimitatore finale del pacchetto. I pacchetti hanno una lunghezza minima di 512 bit, mentre una lunghezza massima di 1512 Byte, esclusi il preambolo ed il SFD. Si definisce per la rete ethernet l'*Inter-Frame Space* (ITR) o *Inter-Packet Gap* (IPG) come il tempo minimo tra due pacchetti consecutivi. Questo viene definito indipendentemente dalla velocità della rete, come il tempo necessario per inviare 96 bit, è necessario per permettere di distinguere due pacchetti inviati consecutivi e determinare si tratti di spazio tra i due.

Agli albori di questa tecnologia si utilizzavano reti condivise e non punto a punto, quindi bisogna effettuare una connessioni a turni, e per occupare la connessioni su reti di 5 km, si scelse la lunghezza minima di 512 bit. Analogamente per la lunghezza massima, se il pacchetto è troppo grande occuperebbe il mezzo trasmissivo per troppo tempo, quindi si è scelto il valore di 1512 Byte.

Nella trasmissione ethernet il livello MAC in trasmissione riceve la LLC-pdu, inserendolo nel pacchetto di livello MAC e convertendolo in bit da passare al livello fisico. In ricezione, converte i bit in un pacchetto MAC, se questo è indirizzato ad un altro oppure contiene errori, calcolando il CRC, viene scartato, altrimenti viene rimossa la parte MAC ed inviato al livello LLC. Scartando pacchetti in questo modo si perde di affidabilità del sistema, ma si guadagna efficienza, poiché si suppone queste informazioni perse vengano recuperate ad un livello superiore (livello di trasporto), senza che sia l'ethernet ad inviare una richiesta di ritrasmissione. Ogni terminale che riceve pacchetti deve quindi ricalcolare il CRC, quest'operazione potrebbe rappresentare il collo di bottiglia e deve essere il più veloce possibile. Questo livello garantisce una distanza minima tra pacchetti rispettando l'IPG e verifica la lunghezza minima del pacchetto. Se un pacchetto non rispetta la lunghezza minima, viene anch'esso scartato. Vengono inoltre generati in trasmissione, e rimossi in ricezione, il preambolo e lo SFD dal pacchetto.

In passato le reti ethernet erano formate da connessioni sullo stesso filo condiviso tra varie stazioni ed usato a turno, non era punto-punto e bidirezionale. Questo mezzo trasmissivo veniva e viene ancora oggi chiamato dominio di collisione. I pacchetti inviati da due o più macchine potevano collidere e si poteva richiedere una ritrasmissione da queste. Questo mezzo condiviso era realizzato da un *repeater* o ripetitore o hub, questo ripete il segnale su tutti i canali connessi e li amplifica, a causa della lunghezza delle reti infatti, il segnale poteva perdere di potenza durante la trasmissione. A livello fisico si comporta come un filo, non memorizza infatti i pacchetti che riceve, ma li trasmette, non è una macchina *Store & Forward*. Ripetitori del genere sono ancora diffusi in alcuni contesti. Si trova al livello fisico ed ha diverse porta su cui può ricevere dati e trasmetterli su tutte le altre.

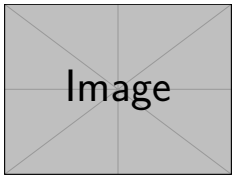

Fig. 12: Rapporto tra Ripetitore e Standard ISO-OSI

Il mezzo trasmissivo per le connessioni ethernet viene realizzato in cavi di rame o fibra ottica, al massimo di 100 m di lunghezza, progettati senza amplificatori. Si usano coppie separate per trasmissione e ricezione per mantenere una connessione bidirezionale. Si usano connettori RJ-45. Reti a fibra ottica invece possono percorrere distanze molto significative rispetto a cavi in rame.

Ethernet II e ethernet IEEE 802 sono diversi tra di loro, lo standard ethernet II non ha uno sottolivello LLC ed è in generale molto più snello. Nelle reti locali convivono connessioni di entrambi gli standard, e sono tendenzialmente molti di più nella vecchia versione di ethernet. Sono necessari quindi meccanismi per mantenere la retrocompatibilità completa dei pacchetti. Non essendoci lo strato LLC, il pacchetto di terzo livello viene contenuto direttamente nel pacchetto MAC, poiché non esiste il livello LLC per questi pacchetti, si utilizza un altro campo type, per specificare a quale protocollo di livello superiore inviare il pacchetto. Questo sostituisce il campo lunghezza di IEEE 802.3. Viene riconosciuto se il campo lunghezza ha un valore maggiore di 1500, lunghezza massima del campo data, e viene interpretato come un codice identificativo di un protocollo di livello superiore.

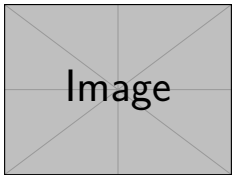

Fig. 13: Rapporto tra Pacchetti Ethernet II e IEEE 802.3

Poiché ethernet ha vissuto un enorme crescita tecnologia, sono presenti diversi sottolivelli dello standard per classificarle, da versioni 802.3u da 100 Mb/s alle ultime versioni 802.3ba/bg/bm dai 40 ai 100 Gb/s. Sono inoltre in corso di standardizzazione tecnologie ethernet nell'ordine dei terabit al secondo.

Generalmente se nel sottolivello dello standard è presente una lettera maiuscola, questo rappresenta un tecnologia estremamente importante.

Sono presenti inoltre molte versioni differenti per scopi diversi, esiste uno standard specifico per il mercato automobilistico, dove è presente poco spazio interno, e quindi si utilizzano cavi da una singola coppia di cavi intrecciati: 802.3bw e 802.3bp (2015/2016) permettono una banda nell'ordine dei gigabit sulla singola coppia. Esiste inoltre uno standard per inviare corrente elettrica su ethernet ed alimentare tramite ethernet telefoni IP a bassa tensione: 802.3af e 802.3at (2003/2009).

3.3 802.1D: *Bridge-Switch*

La connessione instaurata tramite ethernet è bidirezionale simultanea solamente su due calcolatori, ma su una stessa connessione può inviare i dati un solo calcolatore, quindi sono necessarie altre componenti. Un *bridge* è una componente che consente di connettere tra di loro più di un computer tramite ethernet, comportandosi come se fosse un calcolatore intermedio ai calcolatori della rete, connesso a ciascuno di questi tramite una connessione ethernet.

Inoltre connettendo tra di loro diversi *bridge* è possibile creare una struttura più articolata, creando una struttura simile ad un albero. La parte wired o cablata delle connessioni LAN vengono instaurate in questo modo.

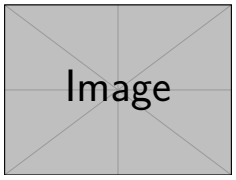

Fig. 14: Rete con *Bridge*

I *bridge* svolgono una prima funzione di rendere possibili topologie articolate, effettuando un'operazione di *filtering*, per separare tra di loro porzioni di rete che non devono dialogare tra di loro in modo diretto

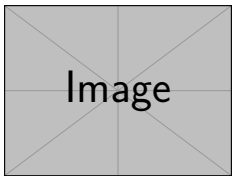

Fig. 15: *Filtering* con *Bridge*

I *bridge* sono delle macchine *Store & Forward*, ovvero quando ricevono un pacchetto, prima di essere inviato su altre porte, viene memorizzato e trasmesso su altre porte, analogamente come se fosse un calcolatore, in caso le altre porte siano impegnate a trasmettere altri pacchetti, quindi in caso di traffico. Si può quindi immaginare una coda di pacchetti sulle porte del *bridge* per essere trasmesse.

Il *bridge* sono delle tecnologie di livello 2, ed utilizzano algoritmi di instradamento per inviarli ad un MAC *address* specifico, ma questo tipo di algoritmo viene effettuato a livello 3. Questo non sorge problemi, poiché quest'operazione di instradamento è interna alla LAN, e non coinvolge alcun'altra componente della rete. I *bridge* devono essere conformi allo standard IEEE 802.1D. Gli standard comprendenti il carattere "D", sono di grande importanza. I sistemi connessi a reti LAN ignorano i *bridge*, si dicono quindi trasparenti, poiché i calcolatori connessi alla rete non conoscono la loro posizione all'interno della rete.

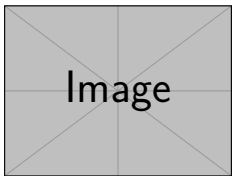

Fig. 16: Rapporto tra *Bridge* e Standard IEEE 802

Un calcolatore per inviare un messaggio ad un altro calcolatore su una rete LAN, invia il suo pacchetto ad un *bridge* attraverso il sul MAC *address*. Il *bridge* quindi utilizza in principio un diverso MAC *address*, per spedire questo pacchetto al computer di destinazione tramite il suo MAC *address*. Tra questi due MAC è presenta un componente di relay, per trasmettere il pacchetto tra porte diverse del *bridge*.

Le porte di un *bridge* possono avere lo stesso MAC o MAC differenti. Poiché il pacchetto è specifico al MAC del *bridge*, deve ricostruire il pacchetto scartando i campi specifici al MAC *address* del *bridge*. Inoltre poiché i pacchetti non sono tutti conformi allo standard IEEE 802.3, deve ricostruire anche il campo LLC. I MAC *address* dei computer nella rete sono realizzati in modo da poter essere connessi a ciascun tipo di MAC.

3.3.1 *Learning*

Si vuole avere un modello di rete *plug & play*, ovvero indipendente da un intervento umano. I *bridge* costruiscono la loro tabella di instradamento per identificare dove sono presenti i diversi indirizzi MAC autonomamente attraverso un meccanismo di *learning*, salvando questa tabella nel *filtering database*. Ogni porta del *bridge* rappresenta una linea ethernet diversa, identificando un loro dominio di collisione, a cui possono essere connessi diversi calcolatori.

Si considera una rete dove ogni componente connesso è spento, ed una tabella vuota. Appena si accende un calcolatore ed invia un pacchetto da un dominio di collisione, allora il *bridge* capisce a quale porta corrisponde il MAC *address* del mittente. Ma ancora non conosce dove si trova il destinatario, quindi lo invia su tutte le sue porte disponibili, su tutta la rete. Invece se conosce la porta dov'è presente il destinatario lo invia solamente su quella porta.

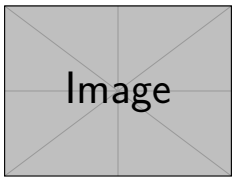

Fig. 17: Processo di *Learning* e *Filtering Database*

Il learning permette di costruire autonomamente il filtering *database* di un *bridge*, questo meccanismo tuttavia non funziona quando la rete presenta una topologia diversa dalla topologia ad albero.

Per esempio se è presente un ciclo all'interno della rete, il *bridge* si vede arrivare un pacchetto dallo stesso MAC *address* su porte diverse. Un albero è una topologia contenente solo *Single Points of Failures* (SPoF) e quindi fortemente sconsigliata, poiché un singolo malfunzionamento causerebbe la perdita di funzionalità dell'intera rete. Per cui data una topologia a grafo, un *bridge* è in grado di calcolare autonomamente un albero ricoprente della rete, ricalcolandolo ad ogni cambiamento della topologia. I *bridge* inoltre vengono collegati tra di loro per più di una connessione per evitare altri SPoF, ed evitare che a un singolo guasto la rete venga tagliata in due.

Tramite un meccanismo progressivo i *bridge* individuano la loro posizione nella struttura dell'albero ricoprente e sono in grado di staccare alcune porte e rimanere collegati sull'intera rete; quando questi *bridge* rilevano un guasto su una di queste connessioni, riattivano una delle porte disattivate per mantenere in funzione la rete, ottenendo una significativa resistenza ai gusti. Questo tipo di algoritmo di *spanning tree* verrà trattato in corsi più avanzati di reti di calcolatori.

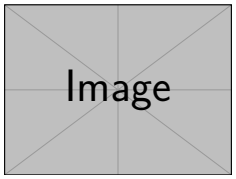

Fig. 18: Rete con Albero Ricoprente

3.3.2 Prestazioni

Le prestazioni di un *bridge* influenzano le prestazioni dell'intera rete locale, vengono identificate da una serie di parametri. Il numero massimo di pacchetti al secondo processabili dal *bridge* rappresentano un collo di bottiglia, limita il numero massimo dii pacchetti che possono essere presenti sulla rete in ogni singolo momento. Infatti se vengono inviati un numero superiore di pacchetti, alcuni verranno scartati. Un altro parametro caratteristico è il tempo medio di latenza, ovvero il tempo in cui il *bridge* prende le sue decisioni ed invia il pacchetto alla porta corretta. Per cui è preferibile avere *bridge* full speed, ovvero con una velocità pari al massimo teorico. Più corti sono i pacchetti maggiore è il numero di decisioni effettuate nell'unità di tempo. Nello standard IEEE 802.3 a 10 Mb/s, un *bridge* si definisce full speed se è in grado di processare 14880 pacchetti al secondo, per ogni porta. Questi esperimenti di verifica a parità di frequenza devono essere effettuati utilizzando pacchetti di lunghezza minima, così ad ogni porta è presente la massima frequenza di funzionamento del *bridge*. Il pacchetto più piccolo che può essere inviato è da 512 bit, per cui il numero massimo di pacchetti al secondo ad una velocità di 10 Mb/s è di circa 19500 pacchetti, ma il pacchetto comprende anche il preambolo ed il SFD, per cui vanno aggiunti altri 64 bit ed il numero di pacchetti al secondo scende a 17300. Tuttavia tra un pacchetto ed il successivo in ethernet è presente l'IPG di 96 bit, in questo modo si ottiene la cifra di 14880 pacchetti al secondo. Per ciascuna tipologia di porta del *bridge* si effettua questa analisi e si verifica nel caso peggiore quanti pacchetti è in grado di gestire.

3.3.3 Architettura di un *Bridge-Switch*

Il *bridge* è un calcolatore, con una CPU, RAM ed interfacce per le diverse LAN, in ROM le funzionalità dello standard IEEE 802.1D, mentre nella memoria principale le tabelle di instradamento i buffer dati, ed eventuali strutture dati ausiliarie. Per *bridge* più potenti, le porte vengono realizzate tramite schede ASIC, per risolvere il problema dell'instradamento localmente. Le porte vengono realizzati tramite diversi slot che possono essere inseriti o rimossi in base al tipo di porta necessaria. Inoltre sono necessari massicci impianti di raffreddamento per riuscire a mantenere basse la temperature dei data center contenenti *bridge-switch*. A differenza di un *server* che può essere rallentano in caso di traffico elevato e quindi diminuire la temperature, le apparecchiature di *bridge* non possono spegnersi, e quindi comportano una temperatura costantemente elevata. *Bridge* di fascia alta sono in grado di effettuare bilanci sulle prese di corrente, per distribuire il carico dell'alimentazione.

Logicamente sono presenti almeno due porte una MAC *relay entity*, per trasmettere i pacchetti tra le varie porte, ed un'entità di livello superiore, per la gestione del *bridge*, degli algoritmi e dei protocolli. Queste entità di alto livello comunicano con altri *bridge* attraverso pacchetti per realizzare lo *spanning tree*.

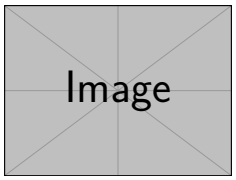

Fig. 19: Architettura Logica *Bridge-Switch*

Le porte del *bridge* possono essere abilitate o disattivate dall'amministratore di rete. Una porta attiva può essere in stato di *forwarding* o di *blocking*, se sono bloccate lo *spanning tree* lo ha bloccate. Ogni porta ha un indirizzo MAC univoco, e sono numerate progressivamente a partire da uno. Convenzionalmente l'indirizzo MAc del *bridge* corrisponde all'indirizzo MAC della porta numero uno.

La tabella di instradamento contiene *entries* (righe) statiche o dinamiche. Le righe statiche vengono inserite dall'amministratore a causa di esigenze di sicurezza importanti, altrimenti la posizione degli indirizzi MAC vengono mantenute per un tempo finito, configurabile di default di 5 minuti. Infatti è possibile che il calcolatore venga spostato spazialmente attraverso la rete, e quindi si colleghi ad una porta differente.

Lo sviluppo di ethernet ha portato alla creazione di meccanismi di controllo di flusso, soprattutto per gli *switch*. Una richiesta attraverso il *bridge* può essere di pochi byte verso un *server*, ma può provocare un trasferimento notevole di dati verso il *client*, quindi attraverso il *bridge* ad una porta ad alta velocità, ma questi pacchetti da ritrasmettere verso il cliente passano attraverso una porta di banda minore, quindi la porta più lente può andare facilmente in saturazione. Viene introdotto quindi tramite lo standard IEEE 802.3x e 802.3bd un controllo di flusso tramite dei *pause frame* un MAC control frame di 512 bit, per fermarsi prima di riprodurre traffico i pause frame non contengono dati ma contengono informazioni di controllo, e rappresentano una novità, viene quindi

implementato attraverso un nuovo sottostato di MAC chiamato MAC control. Il supporto allo standard 802.3x è opzionale e viene negoziato tra le schede alle due estremità del filo.

Prima dello standard 802.3x poiché ethernet era una comunicazione a turni, per impedire la saturazione i *bridge* potevano inviare pacchetti senza dati prendendo il controllo della connessione.

3.4 802.11: WiFi

Nelle reti ethernet, si ha una connessione punto a punto bidirezionale e simultanea, attraverso *bridge* che permettono di comunicare tra diversi computer senza connetterli direttamente. Per cui gli indirizzi MAC ethernet sono molto semplice, ma questo non si può dire per le reti WiFi. Una rete locale è un ambiente dove tutti parlano contemporaneamente con tutti, e ciò non può avvenire su di un filo condiviso tra due calcolatori. Quindi le connessioni WiFi presentano un indirizzo MAC molto più complesso.

Ethernet è molto versatile, ma su alcuni edifici non è possibile effettuare un cablaggio economico, oppure sono presenti uffici nei quali gli impiegati sono presenti occasionalmente. Oppure nel caso di reti offerte pubblicamente, con utenti occasionali. Per questi motivi di praticità nei portatili moderni sono presenti schede di rete WiFi e non ethernet.

Nonostante questi benefici, il mezzo trasmissivo non è affidabile, e comporta un costante consumo elettrico per connettersi alla rete, inoltre comprende una zona di copertura limitata. Esistono diversi studi sull'uso delle frequenze o micro-frequenze sulla salute.

Il sistema WiFi nel venne definito nel comitato IEEE 802, dove forma il working group 802.11 dedicato alle LAN senza fili. Il primo standard ad affermarsi è 802.11b. Nel 1999 si forma il consorzio *Wireless Ethernet Compatibility Alliance*, successivamente denominato WiFi *Wireless Fidelity Alliance* per certificare i prodotti IEEE 802.11.

3.4.1 Architettura

Una rete WiFi può presentare architetture di due tipi, ad hoc o strutturate. In una rete ad hoc le stazioni comunicano direttamente l'una con l'altra:

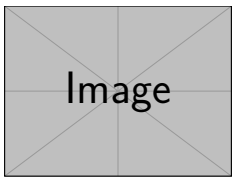

Fig. 20: Rete WiFi ad Hoc

Quest'architettura è prevista dallo standard, ma le reti WiFi non funzionano in questo modo, è stata progettata per permettere di comunicare ai dispositivi personali, ma su questo ambiente applicativo ha prevalso il Bluetooth.

Le reti WiFi sono invece realizzate con un'architettura strutturata, dove tutte le stazioni (astrazione di calcolatore poiché funziona su tante tipologie di dispositivi) possono accedere solamente

tramite punti di accesso *Access Point* (AP). Questi punti di accesso sono interconnessi mediante fili, quindi non rappresenta una rete completamente senza fili. Questi collegamenti vengono realizzati tramite ethernet, rappresentato come un unico domino di collisione, nonostante sia presente una topologia più complessa. In queste reti non è presente un master, ma tutti gli AP sono allo stesso livello, in modo che non siano presenti SPoF.

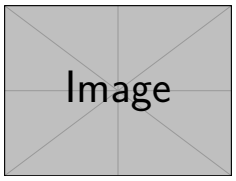

Fig. 21: Rete WiFi Strutturata

Quest'architettura permette una semplice scalabilità, infatti è sufficiente connettere altri AP alla rete ethernet. L'informatica deve essere scalabile, con costi limitati deve poter aumentare le sue infrastrutture.

Ciascuno di questi AP ha un MAC di tipo IEEE 802.11, e si comporta come un *bridge*, presenta almeno due interfacce, una 802.11 per comunicare con i dispositivi wireless, ed un'altra porta 802.3 per comunicare sulla rete ethernet. Ma questi pacchetti possono essere inviati con due MAC differenti, quindi necessitano di un'entità di relay per poter gestire questi i due diversi tipi. Ogni AP controlla un *Basic Service Set* (BSS), che estende la rete cablata. Ogni BSS ha un identificatore, BSSID, che può essere il MAC della scheda AP corrispondente, ma è possibile assegnare valori arbitrari. La parte cablata si chiama *Distribution System* (DS), e rappresenta l'infrastruttura portante della rete.

Sono presenti delle eccezioni, infatti è possibile costruire dei collegamenti tra AP che estendono il DS tramite WiFi. Ma dal punto di vista dell'architettura si comporta come un filo, collegando solamente due AP.

Per gestire il sottolivello MAC, bisogna gestire il problema dell'accesso al mezzo trasmissivo condiviso. Inoltre bisogna avere una spedizione affidabile dei pacchetti su un mezzo trasmissivo poco affidabile. Livelli fisici differenti utilizzano frequenze e bande diverse. Il sottolivello MAC viene a sua volta diviso in due sottolivelli, questo viene effettuato per permettere di realizzare architetture modulari. Un sottolivello *Distributed Coordination Function* (DCF), e *Point Coordination Function* (PCF), questi due livelli sono connessi indipendentemente al sottolivello LLC. L'accesso distribuito al mezzo trasmissivo viene realizzato nel DCF, questo è il modello effettivamente utilizzato. Rappresenta il MAC vero e proprio, e tramite questo le macchine tentano di accedere al mezzo trasmissivo, ma non è garantito questo accesso, e può comportare ritardi. In alcuni ambienti applicativi si vuole accedere al mezzo trasmissivo con garanzia di un tempo di ritardo massimo che non superi una certa soglia. In queste applicazioni si sceglie una stazione di riferimento centrale che in ogni intervallo temporale indica chi può trasmettere con il mezzo trasmissivo. Ambienti *Mission Critical*, dove una macchina deve necessariamente effettuare un azione in un certo intervallo di tempo utilizzano quindi il PCF.

La probabilità che il ritardo sia elevato in una rete WiFi è molto bassa, per cui questa tecnologia non viene quasi mai utilizzata. Poiché bisognerebbe utilizzare un controllore centralizzato che si cerca molto spesso di evitare in una rete.

3.4.2 Indirizzamento

Per qualunque MAC IEEE 802.2 deve essere presente l'indirizzo del destinatario, del mittente, il campo dati e il FCS. Invece nello standard 802.11 sono presenti campi come il *Frame Control* che indica il tipo di pacchetto, di controllo o contenente dati, e fornisce anche informazioni sul mittente ed il destinatario del DS, sulla frammentazione e sulla riservatezza. Contiene un campo per indicare il tempo necessario in cui deve essere effettuata la trasmissione. Sono inoltre presenti fino a quattro indirizzi MAC. Il *Sequence Control* contiene informazioni utili per la frammentazione o il riassemblaggio dei pacchetti. Il pacchetto contiene una LLC pdu oppure informazioni di controllo ed un FCS di 32 bit.

Si parla quindi di indirizzamento per spiegare il motivo per cui sono presenti fino a quattro indirizzi MAC in un unico pacchetto. Ogni scheda di rete wireless contiene un suo indirizzo MAC, e questa raccoglie i pacchetti e verifica che sia diretto alla stessa macchina dal suo indirizzo MAC. Nel pacchetto di livello MAC sono presenti quattro indirizzi numerati da 1 a 4, più due bit. Questi bit sono ToDS e FromDS, se il primo bit vale 1, questo pacchetto è spedito all'AP per essere smistato dal DS, il secondo vale uno quando il pacchetto è destinato al DS. In funzione di questi valori i quattro indirizzi hanno significati diversi.

ToDS	FromDS	Addr. 1	Addr. 2	Addr. 3	Addr. 4
0	0	DA	SA	BSSID	
0	1	DA	BSSID	SA	
1	0	BSSID	SA	DA	
1	1	RA	TA	DA	SA

Dove RA, *Recipient Address* indica la scheda del ricevente, TA, *Transmitter Address*, del trasmittente. DA, *Destination Address*, indica il destinatario finale e SA, *Sender Address*, indica la sorgente.

In generale il primo indirizzo è sempre l'indirizzo MAC della scheda di rete a cui è destinato il pacchetto, ed il secondo indirizzo della scheda che lo trasmette. Il terzo indirizzo contiene l'indirizzo sorgente se viene spedito dal DS, ed il destinatario se viene spedito al DS. In caso è un pacchetto di comunicazione tra il DS entrambi questi bit valgono uno, quindi si utilizza il quarto indirizzo per contenere l'indirizzo sorgente.

Quando due computer comunicano direttamente, su una rete ad hoc, il primo indirizzo del pacchetto corrisponde all'indirizzo del destinatario ed il seguente all'indirizzo del mittente. Il terzo campo di indirizzo corrisponde al BSSID, ma questo si riferisce solamente alle reti strutturate. In questo modello le macchine comunicano tra di loro in gruppi chiusi, quindi condividano un identificatore chiamato BSSID. La connessione è quindi lecita solamente se il BSSID è lo stesso. Poiché si tratta di una comunicazione diretta, i suoi bit hanno valore nullo.

In una trasmissione da una stazione ad un AP, il primo indirizzo è l'indirizzo dell'access point, il secondo è quello del mittente vero. Il terzo indirizzo è quello del pacchetto vero. Questi AP sono

comunque trasparenti rispetto alle macchine, poiché alla prima connessione il calcolatore memorizza l'indirizzo del BSSID a cui può richiedere accesso per inviare e ricevere pacchetti.

Quando un pacchetto viene inviato dal DS, il primo indirizzo è il MAC del destinatario vero, il secondo è l'indirizzo dell'AP che lo invia, ed il terzo è l'indirizzo del vero mittente.

Nell'ultimo caso, il pacchetto è inviato e ricevuto dal DS, è il caso di due AP che comunicano tra di loro tramite WiFi. Sono necessari i due indirizzi degli AP e gli indirizzi del destinatario e del mittente vero.

3.4.3 Sottolivello DCF ed RTS/CTS

Per realizzare dei turni senza una stazione di coordinamento centralizzata si utilizza il sottolivello DCF. Il DCF ha come requisiti principali evitare interferenze di trasmissioni simultanee, consentendo il maggior numero possibile di connessioni e gestendo il canale trasmissivo in modo equo. Non si vuole utilizzare un controllore centralizzato, e quindi non si può utilizzare un clock. Le trasmissioni sono quindi asincrone. Si utilizza un algoritmo *csma/ca*, *Carrier Sense Multiple Access/Collision Avoidance*. Questo meccanismo determina se il mezzo trasmissivo è disponibile, in caso una stazione ha un pacchetto da spedire *Carrier Sense*. Se il mezzo è occupato aspetta che la stazione sia libera prima di trasmettere. È possibile che due stazioni provino a trasmettere contemporaneamente, in questo caso si verifica una collisione ed i pacchetti diventano intellegibili per i destinatari, e dovranno essere ritrasmessi. Per evitare le collisioni si utilizzano degli strumenti come *backoff*, *acknowledgment*, *Request to Send/Clear to Send* (RTS/CTS).

DCF per effettuare il suo lavoro utilizza gli intervalli tra due pacchetti consecutivi per gestire delle priorità. Questo tempo è l'interpacket gap, che in ethernet è lungo 96 bit-time. Ci sono due tipi principali di IFS, *Inter-Frame Space*, di tempo variabile tra i vari pacchetti: DIFS, DCF IFS, in generale quello standard, e SIFS, *Short* IFS di lunghezza minore. Se un pacchetto consecutivo può essere trasmesso immediatamente, allora si utilizza il SIFS, altrimenti si utilizza il tempo di standard DIFS.

Per ora si considera il DCF senza il RTS/CTS, per evitare collisioni. Quando una stazione vuole trasmettere, ed il canale è occupato, capisce che è presente traffico nel canale, e si autolimita scegliendo un numero random di *backoff*, nell'intervallo $[0, cw]$, ed incrementa il timer solo quando è libero. La trasmissione può essere effettuata solamente quando il timer raggiunge il termine.

Quando una stazione rileva una collisione, utilizzando vari meccanismi, duplica il cw in modo che l'intervallo di casualità sia duplicato, per rendere più improbabili collisioni future. Il valore di cw è limitato superiormente al valore $cw_{\max}$, immutabile. Se è un pacchetto viene inviato con successo, il valore di cw viene posto al valore minimo $cw_{\min}$. Valori ragionevoli per questo intervallo $[cw_{\min}, cw_{\max}]$ sono $[7, 31]$ e $[255, 1023]$.

Per determinare se il canale trasmissivo è libero, si utilizza un'altro meccanismo. In ogni pacchetto viene specificata la sua durata, nel campo *duration*. Poiché il mezzo trasmissivo è condiviso, tutte le stazioni connesse ascoltano questo campo e si segnano la durata della trasmissione corrente nel vettore NAV, *Network Allocation Vector*. Rappresenta un contatore per sincronizzare le stazioni. Ogni stazione lo decrementa con il passare del tempo, ed ogni stazioni può trasmettere solamente se questo campo vale zero. Questo campo deve essere presente come primo campo del pacchetto, per essere letto.

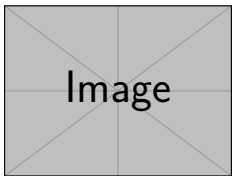

Fig. 22: Trasmissione Senza RTS/CTS

Ogni pacchetto spedito da una stazione deve essere riscontrato dalla stazione destinataria tramite un *acknowledgment*. Quindi una macchina quando calcola la durata della trasmissione, calcola il tempo di trasmissione, nota la banda ed il numero di bit del pacchetto, una durata SIFS più corta di DIFS prima dell'invio dell'*acknowledgment*, e la durata di questo piccolo pacchetto. Il SIFS corrisponde all'*interpacket space* che separa un pacchetto dal suo *acknowledgment*.

La verifica della disponibilità della rete può essere effettuata anche a livello fisico, controllando se nel canale trasmissivo è presente del segnale attivo. Se viene rilevato un segnale attivo, la stazione aspetta di trasmettere, quindi solo quando entrambe le condizioni fisiche e logiche sono verificate. Una trasmissione non si interrompe fino alla fine, e termina quando viene ricevuto il pacchetto *ack*. La stazione mittente si calcola quando dovrebbe ricevere l'*ack*, se non viene ricevuto, oppure è intelligibile, allora si è verificata una collisione, e deve essere ritrasmesso il pacchetto. Quindi tra tutte le stazioni nella rete solo la stazione trasmittente è in grado di accorgersi che si è verificata una collisione. In questo caso duplica il valore di *cw* ed aspetta prima di inviare nuovamente il pacchetto.

Se una stazione trasmettesse senza interruzioni, allora il mezzo trasmissivo non sarebbe mai libero. Questo potrebbe rappresentare un malfunzionamento oppure un attacco di tipo DOS *Denial Of Service*.

Se il pacchetto che viene trasmesso è molto lungo, e si effettua una collisione, la stazione mittente se ne può accorgere solamente dopo il periodo di trasmissione di questo lungo pacchetto. Si perde molto tempo prima dell'identificazione della collisione. Per risolvere questo problema la dimensione massima dei pacchetti WiFi è di 1500 byte come in ethernet. Inoltre è possible che tra le stazioni ci sia visibilità parziale, ovvero può vedere solo una parte della rete, a causa della loro collocazione.

Per risolvere questi problemi si considera l'algoritmo DCF con RTS/CTS. Quando una stazione vuole trasmettere, invia un *frame* al destinatario, di breve lunghezza, chiedendo l'autorizzazione alla trasmissione. Se il destinatario è disponibile emette un breve *frame* di conferma. Alle stazioni vicine è richiesto di non interferire per l'intera durata della trasmissione che sta per avvenire. Questo meccanismo di prenotazione del canale tra due stazioni permette di evitare le collisioni. Se il canale è libero, la stazione mittente invia un pacchetto RTS per richiedere l'autorizzazione, e viene concessa alla stazione con un pacchetto CTS. Tutte le altre stazioni aspettano la durata di tempo indicata nel campo *duration* dei pacchetti RTS e CTS, invece che nel pacchetto da trasmettere.

Le stazioni memorizzano la durata residua nel loro NAV, decrementandolo al passare del tempo. Una collisione si verifica quando all'invio del RTS, non viene inviato il CTS, causato da un invio simultaneo del RTS. Analogamente alle condizioni precedenti, dove al pacchetto non viene seguito l'*ack*.

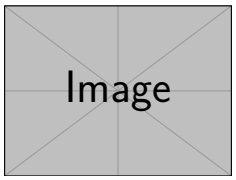

Fig. 23: Trasmissione Con RTS/CTS

Quindi nel MAC 802.11 l'algoritmo di *backoff* impone ad una stazione che rileva una collisione o trova il canale occupato di scegliere casualmente in $[0, cw]$ utilizzato per inizializzare il timer di attesa del *backoff* moltiplicandolo per lo *slot time*. Ad ogni collisione sullo stesso pacchetto viene raddoppiato il cw fino ad un valore massimo di $cw_{\max}$. Quando il canale è libero, ovvero se NAV è nullo, viene decrementato il timer per ogni *slot time* passato. Quando il timer si azzera la stazione può trasmettere e se ha successo il wuo cw viene riassegnato al valore minimo $cw_{\min}$. L'algoritmo di backoff rappresenta un algoritmo distribuito su molte macchine.

Su ogni stazione WiFi è presente un parametro che indica quale pacchetti da utilizzare. Viene utilizzato RTS/CTS per pacchetti di lunghezza maggiore a questo parametro s. Molto spesso questo valore corrisponde alla dimensione massima di un pacchetto 1500 Byte.

3.4.4 *Handshaking*

In queste reti senza fili l'insieme delle stazioni appartenenti ad uno stesso BSS cambia continuamente. Per poter accedere ad un BSS si possono utilizzare protocolli di *handshake* in due modalità.

Si utilizza un protocollo di *handshake* per scambiare informazioni tra l'AP ed il BSS. Ogni stazione AP presente il proprio indirizzo MAC, potrebbe inviare un messaggio al BSS per indicare la sua presenza alla rete, tramite un pacchetto *broadcast* chiamato *beacon frame*, ricevuto da tutte le stazioni connesse Altrimenti le stazioni potrebbero inviare pacchetti sonda o *probe*, di tipo *probe request*, per esplorare la rete, ed attende un pacchetto di *probe response* di risposta dall'AP. Queste rappresentano le due modalità per cui una stazione può accedere ad un BSS.

Un amministratore può definire varie reti logiche sulla stessa rete fisica, ciascuna identificata a un suo SSID, *Service Set ID*. Un AP con un BSSID può rendersi disponibile per trasportare il traffico di uno o più SSID. Solo chi ha gli opportuni permessi può accedere ad una specifica rete logica definita da uno specifico SSID. Si definisce ESS, *Extended Service Set* l'insieme delle stazioni appartenenti ai BSS di una rete e con lo stesso SSID. Queste stazioni nello stesso ESS possono muoversi cambiando BSS, mantenendo invariato il loro ESS.

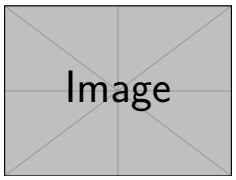

Fig. 24: Rete WiFi con BSS ed ESS

3.4.5 Frammentazione

Il livello MAC può decidere se frammentare un pacchetto ed in caso si occupa anche del riassemblaggio. In queste reti è consigliabile diminuire l'*overhead* di un pacchetto e ridurre la probabilità di collisione, diminuendo la dimensione dei singoli pacchetti. Ogni frammento del MAC *Service Data Unit* viene frammentato e ciascuno di questi frammenti viene trattato come un pacchetto e viene riscontrato singolarmente.

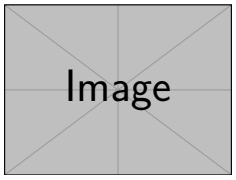

Fig. 25: MAC *Service Data Unit* Frammentata

La dimensione dei frammenti può essere modificata dall'utente per guadagnare un vantaggio in trasmissione sulle altre stazioni. In ricezione il livello MAC ricompone questi frammenti in modo che gli strati superiori non si accorgono della frammentazione, né gli altri MAC che non sono coinvolti nella trasmissione.

In questo momento storico i riscontri di LLC non vengono utilizzati, quindi anche se nello standard è possibile che l'invio e ricezione di pacchetti *ack* sia effettuata a livello LLC, nella realtà non viene realizzato.

Per inviare pacchetti di tipo *broadcast*, dove ToDS sia pari a zero, non si utilizzano *ack* e RTS/CTS, le eventuali collisioni non vengono quindi rilevate. Quando ToDS è pari ad uno, si utilizzano pacchetti di *ack*, ed RTS/CTS per i pacchetti diretti verso gli AP. Si può configurare l'AP in modo che ogni pacchetto *broadcast* ricevuto venga inviato o meno a tutto il BSS.

4 Livello 3: Il Livello di Rete

Il livello di rete sceglie un percorso per i pacchetti, conoscendo la topologia della rete, attraverso passaggi intermedi chiamati salti o *hop* tra varie stazioni.

Se il destinatario è nella stessa LAN del mittente, allora lo raggiunge direttamente, altrimenti deve percorrere diverse reti geografiche effettuando questi salti. Fino ad ora ci si è occupati di ogni salto separatamente nella rete locale o nella rete geografica. Queste due reti presentano due topologie molto diverse, la rete locale presenta una topologia molto complessa con diverse stazioni intermedie, mentre le reti geografiche sono composte da una trasmissione punto a punto su un unico filo. Per connettere tra di loro la LAN e la WAN si utilizzano reti di raccolta o di accesso.

4.1 Reti di Raccolta e Reti di Accesso

Nella LAN la trasmissione tra due stazioni, per quanto sia complessa la topologia della rete, è sempre diretta. Nelle reti geografiche (WAN), la trasmissione tra due stazioni a livello due avviene su un canale punto a punto. Viene realizzato solamente tramite un filo, non sono necessarie altre componenti della rete. Tipicamente questo filo è di tipo ethernet, anche se lo standard è stato sviluppato principalmente per reti locali.

I collegamenti tra questi due tipi di rete vengono realizzati dall'ISP di cui si usufruiscono i servizi. Questa zona intermedia viene realizzata tramite reti di raccolta o di accesso. La rete di raccolta è la rete nella quale si raccoglie il traffico, mettendolo assieme, proveniente da tutte le stazioni coperte dal servizio dell'ISP. Le reti di accesso permettono, una volta raccolto il traffico, di accedere alla rete geografica vera e propria, composta da lunghi collegamenti punto-punto.

Per molti anni in quasi tutti i paesi del mondo, la rete di raccolta è stata la rete telefonica composta da un doppino telefonico, due fili di rame. Questa struttura molto pervasiva è stata il supporto delle comunicazioni di rete per molti anni. In seguito venne effettuata una progressiva sostituzione tra rame e fibra ottica, secondo una terminologia FTTx, *Fiber To The x*, che distingue le reti di raccolta in base alla vicinanza della rete locale alla fibra ottica. La linea di tendenza punta a FTTH, *Fiber To The Home*, dove la fibra ottica arriva direttamente alla stazione dell'utente. Un altro metodo di raccolta simile è FTTB, *Fiber To The Building*, che utilizza le esistenti strutture telefoniche per trasmettere i dati all'interno dell'edificio.

FTTN, *Fiber To The Node*, e FTTC, *Fiber To The Center*, sono difficili da distinguere, questi rappresentano gli accessi xDSL o aDSL, *x/aDigital Subscriber Line*, quelli normalmente venduti, il carattere prefisso di DSL descrive la trasmissione in termini di bilanciamento e traffico disponibile. In generale il cavo in fibra ottica raggiunge una stazione centrale che utilizza le preesistenti reti telefoniche per raggiungere gli utenti ed i loro edifici. FTTN ed FTTC si distinguono in base alla distanza tra queste stazioni centrali e le stazioni degli utenti, in generale se la distanza è maggiore di 300 metri si parla di FTTN, altrimenti FTTC.

Tipicamente gli accessi FTTH e FTTB sono a 1 Gb/s, recentemente si stanno vendendo accessi a velocità superiore. Queste strutture utilizzano la tecnologia GPON, *Gigabit-capable Passive Optical Network*, il dispositivo vero e proprio di cui fanno uso è l'OLT, *Optical Line Termination*, essenzialmente è uno *switch* che gestisce fibre ottiche. Permette di collegare fino a 64 clienti sullo stesso OLT. Al livello del cliente è presente un dispositivo *router* su cui si può connettere per acce-

dere alla rete, questi sono collegati a vari livelli di *splitting* ottico, per connettere tutti i *router* dei clienti, connessi tutti su una porta di uno stesso *switch* OLT. Questi dispositivi di *splitting* ottico non necessitano di alimentazione, ma uniscono tra di loro il traffico proveniente da più clienti.

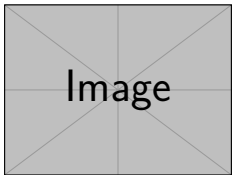

Fig. 26: OLT e *Splitting* Ottico

La banda di questi OLT arriva fino a 1 Gb/s, ma se più clienti vengono connessi alla stessa porta allora la banda effettiva per clienti diminuisce all'aumentare dei clienti, fino a 64 per porta. Questa è la rete con gli ISP attualmente raccolgono il traffico.

Per essere collegati alla WAN, si utilizza la rete di accesso, realizzata tramite *switch* OLT tutti collegati tra di loro, con un estensione geografica significativa. Ma è consigliabile non estendere troppo questa struttura, generalmente indicata come MAN. La maggior parte degli ISP realizza la propria rete di accesso tramite un anello che connette le varie reti di raccolta e ciò che lo collega alla spina dorsale, la WAN, è un nodo di aggregazione. Vengono realizzati in cicli per mantenere una certa resistenza ai guasti, e si utilizzano alberi ricoprenti di anello e *loop-avoidance*, con tecnologie analoghe alle reti locali, per mantenere attivi solo i collegamenti strettamente necessari. Se si rompe un singolo *switch*, solo i clienti attestati a quello *switch* subiranno un disservizio. Generalmente vengono utilizzati più *aggregation node*, per mantenere un'importante resistenza ai guasti. Questo meccanismo di duplicare componenti può essere attuato a vari livelli per mantenere alta la resistenza dell'ISP.

Molto spesso per aumentare la robustezza si utilizzano più anelli con doppi collegamenti, anche in fibra, che effettuano percorsi diversi. Si utilizza un anello poiché è la struttura contenente cicli più semplice possibile.

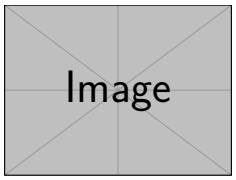

Fig. 27: Rete di Accesso

4.2 Indirizzamento ed Instradamento

Il livello di rete fornisce servivi al livello di trasporto, che non è interessato della topologia delle varie reti attraversate per raggiungere una destinazione. Inoltre non conosce le tecnologie attraversate al

livello due. Può offrire servizi connessi e non. Il protocollo di rete più diffuso IPv4 utilizza servizi non connessi, utilizzando instradamento a datagramma, mentre un livello di rete connesso utilizza la commutazione a circuito virtuale. Questa distinzione tuttavia non è strettamente mantenuta.

Il livello di trasporto deve conoscere degli indirizzi distribuiti in modo consistente su tutta la rete, in modo da poterle identificare univocamente tra rete locale e geografica. Una primitiva di servizio non connesso offerta al livello di trasporto indica l'indirizzo livello tre, univoco, del destinatario ed il suo *payload*, questo sarà ricevuto dal livello quattro corrispondente. Il livello tre si occupa dell'effettiva trasmissione attraverso la rete.

Dal punto di vista del livello tre i sistemi possono essere *End System*, es, o *Intermediate System*, is. Ad ogni sistema viene associato un indirizzo numerico per poterlo identificare, spesso viene associato anche un nome, la cui corrispondenza all'indirizzo viene gestita da *server* appositi presenti sulla rete.

Spesso queste apparecchiature intermedia si chiamano *router* o *gateway*, contengono almeno i primi tre strati della pila ISO-OSI, talvolta per motivi di gestione devono contenere anche livelli superiori. Alcuni casi di instradamento avvengono anche a livello due, anche se principalmente a livello tre.

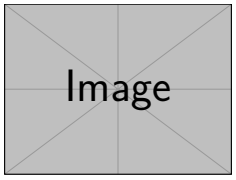

Fig. 28: *Router*, *Bridge* e *Repeater* e Modello ISO-OSI

L'indirizzo di livello due identifica un destinatario solamente all'interno di una LAN, mentre l'indirizzo di livello tre deve poter identificare il destinatario all'interno del'intera rete. Un sistema necessita di tanti indirizzi MAC quante sono le schede di rete al suo interno, ma una singola macchina può essere associata ad un singolo indirizzo di livello tre, oppure, per alcuni protocolli come IPv4 e IPv6, ad indirizzi di livello tre differenti.

Esistono protocolli appositi per gestire e stabilire la corrispondenza tra gli indirizzi di livello due e livello tre. Ma gli indirizzi di livello due, i MAC *address*, sono univoci a livello globale, quindi si potrebbe già utilizzare questi come indirizzi di livello tre. Il problema degli indirizzi MAC è che sono distribuiti casualmente, ma sarebbe utile ai fini di individuare le stazioni se gli indirizzi associati a macchine vicine avessero caratteristiche simili. Poiché gli indirizzi MAC individuano univocamente la scheda di rete, mentre gli indirizzi di livello tre possono cambiare, utilizzare solo i MAC comporterebbe problemi di *privacy*.

In linea di principio si possono individuare tre tipi di instradamento principali:

- *Routing by Network Address*: nel pacchetto c'è l'indirizzo del sistema destinatario, potrebbe non essere l'indirizzo dell'es, ma di una sua interfaccia. Si applica la commutazione a datagramma su questo indirizzo;

- *Label Swapping*: nel pacchetto non c'è l'indirizzo, ma un'etichetta che individua un cammino virtuale, con commutazione a circuito virtuale basata su queste etichette;
- *Source Routing*: nel pacchetto è indicata la lista ordinata di tutti gli is da attraversare per raggiungere la destinazione.

4.2.1 *Routing by Network Address*

Quando il pacchetto raggiunge un is, con varie linee di inoltro, questo guarda la sua tabella di instradamento locale e cerca come chiave l'indirizzo del destinatario e restituisce la linea dove deve essere inoltrato il pacchetto. La commutazione è a datagramma poiché questa tabella può variare nel tempo. Questo rappresenta il modo più semplice di instradamento, operato dagli *switch*, dove la tabella di instradamento viene chiamata *filtering database*.

Se l'indirizzo non fosse presente nella sua tabella di instradamento, allora il pacchetto verrebbe buttato, a differenza dei *bridge*, poiché su reti grandi l'inoltro di tutti i pacchetti sconosciuti su tutte le linee comporterebbero un aumento considerevole del traffico.

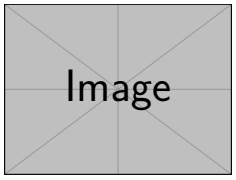

Fig. 29: Instradamento *Routing by Network Address*

Sono necessari quindi dei protocolli per poter definire gli indirizzi e conoscere la topologia della rete, essendo dinamica. Questi meccanismi permettono di compilare automaticamente le tabelle di instradamento dell'intera rete. Ma è possibile che siano compilate manualmente, anche se non è realistico. Questo processo di instradamento è indipendente da chi ha compilato la tabella.

4.2.2 *Label Swapping*

Se due stazioni vogliono comunicare, in una rete a circuito virtuale, viene stabilito il percorso che deve attraversare il pacchetto. Le apparecchiature intermedie vengono quindi informate sul percorso che deve essere effettuato. Questo percorso viene identificato da un'etichetta in modo che all'arrivo del pacchetto presso un is, si utilizza una tabella dove sono presenti le etichette ed i possibili percorsi, e quindi viene inviato su una certa linea. Questa tabella è locale e molto piccola, e determina la linea di ritrasmissione del pacchetto.

Deve essere stabilito il percorso tra le stazioni, analogamente alla tabella di instradamento precedente. La differenza tra queste due tabelle è nelle loro dimensioni, infatti è necessaria una tabella che contenga tutti i destinatari per il meccanismo precedente, mentre una tabella delle etichette deve contenere solamente i possibili percorsi. Il numero di righe non è funzione del numero di stazioni globalmente presenti nella rete, ma dal numero di circuiti virtuali che la attraversano

in ogni dato istante. È irragionevole che in una rete tutte le stazioni trasmettano a tutte le altre stazioni.

Per evitare di dover generare etichette uniche in tutta la rete, e quindi verificarne la disponibilità, ad ogni tratto del percorso viene assegnata un'etichetta diversa localmente.

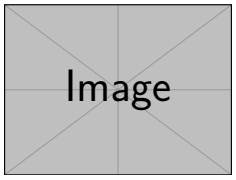

Fig. 30: Instradamento *Label Swapping*

In questo modo il numero di etichette disponibili non è un limite superiore al numero dei circuiti virtuali attivabili. Queste etichette sono contenute in un campo nell'intestazione del pacchetto.

4.2.3 *Source Routing*

Nel pacchetto viene specificata la lista delle stazioni da attraversare per raggiungere la destinazione. L'is deve solo leggere questa lista e determinare la stazione successiva al quale inoltrare il pacchetto.

Rappresenta il meccanismo di instradamento più veloce per le stazioni intermedie, ma il percorso deve essere definito a priori e non può essere modificato.

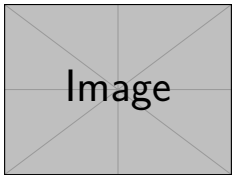

Fig. 31: Instradamento *Source Routing*

4.3 *Router*

L'architettura di un *router* è composta da un algoritmo per calcolare la tabella di instradamento, la tabella stessa ed il processo di inoltro dei pacchetti. Sono inoltre presenti almeno due interfacce su cui esegue l'operazione di inoltro. In questo corso non ci si occupa dell'algoritmo di creazione automatica della tabella di instradamento.

La tabella di instradamento ed il processo di ritrasmissione di un pacchetto costituiscono il *data plane*, mentre l'algoritmo di costruzione della tabella di instradamento costituisce il *control plane*. Questa rappresenta l'architettura logica di un *router*.

In un *router* multi-protocollo sono presenti diversi di queste pile di *data* e *control plane* per ogni protocollo, mantenendo le stesse interfacce. Un *router* può partecipare contemporaneamente

a diverse tecnologie, senza che queste diverse tecnologie e protocolli si conoscano a vicenda sulla stessa macchina. Se ad un *router* arriva un pacchetto destinato ad un protocollo che non può gestire, viene scartato, per mantenere le prestazioni.

4.4 L'*Internet Protocol Suite* TCP/IP

Il protocollo di rete IPv4, *Internet Protocol version 4*, appartenente alla *suite* TPC/IP o *Internet Protocol Suite*, è il principale protocollo di rete di livello 3. L'*Internet Protocol Suite* è la pila protocollare utilizzata dall'*internet*, implementata nella quasi totalità dei computer. Questi protocolli sono descritti nei documenti RFC, *Requests For Comments*.

Alla fine degli anni '70 nasce l'*Internet Protocol Suite* con *Transmission Control Protocol* o TCP, ed il protocollo *internet* IPv4 con una prima versione di *internet* chiamata *arpanet*. Da allora è in costante crescita. Già dai primi anni della sua storia *arpanet* permetteva connessioni tra le due coste oceaniche degli Stati Uniti. Tra protocolli di TCP/IP abbiamo IPv4, IPv6, ICMP ed ARP, protocolli di supporto, e protocolli di *routing*, di *control plane* per costruire le tabelle di instradamento automaticamente, etc.

Si possono realizzare diverse architetture utilizzando protocolli diversi dell'*Internet Protocol Suite* ad ogni livello.

Quando nasce la TCP/IP, nasce secondo i principi progettuali della semplificazione estrema del *data plane*, in modo che l'inoltro dei pacchetti sia effettuato nel modo più semplice possibile. Tutte le funzioni complesse vengono spostate sugli es. Il *routing* è effettuato dall'indirizzo di rete e viene realizzato tramite commutatone a datagramma. Il servizio offerto al livello 4 è non connesso. Il servizio è di tipo *best effort*, ovvero un pacchetto può essere corrotto, perduto o consegnato in ordine errato, ma non è il livello tre che si occupa della sua ritrasmissione.

Questi protocolli vennero progettati con il principio della vita stretta, o *narrow waist*, dove i pacchetti a livello 3 sono tutti realizzati con IPv4 in modo molto snello, per permettere un'enorme interoperabilità tra molti apparati, poiché è il protocollo parlato da tutti.

4.4.1 Il Protocollo IPv4

IPv4 è attualmente, insieme ad IPv6, il protocollo principale di livello tre, descritto nell'RFC 791. I pacchetti di questo livello si chiamano *datagram*, datagrammi. Provvede a funzioni di instradamento ed indirizzamento, il motivo per cui nasce il livello tre, ma svolge anche la funzione di frammentazione, riassemblaggio e rilevazione degli errori. Per i pacchetti la frammentazione è compito del livello quattro, ma già nelle reti WiFi i pacchetti vengono frammentati, anche se nessuno all'esterno della rete WiFi può rilevare la frammentazione. La frammentazione offerta dal livello quattro compie una riframmentazione, per i pacchetti già frammentati al livello tre poiché troppo grandi. Al contrario di una rete WiFi, nella rete di livello tre i frammenti vengono rilevati.

I pacchetti di livello tre devono essere contenuti all'interno di un pacchetto di livello due, di campo dati di dimensione 1500 byte. Per cui *datagram* più grandi di questa dimensione devono essere frammentati. Questo rappresenta un vincolo sulla dimensione del *datagram*, questi pacchetti vengono frammentati solo se strettamente necessario. Provvedere al *routing* deve essere semplice, quindi all'arrivo dei frammenti di un *datagram*, il livello tre di un *router* non può riassemblarli, e quindi inoltra i pacchetti allo stesso modo, siano frammentati o meno.

I pacchetti di livello tre di tipo IPv4 sono divisi in due parti, il *datagram header* ed il campo dati.

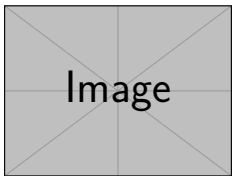

Fig. 32: Struttura IPv4 pdu

Questo formato è il linguaggio di quasi tutti gli es ed is del mondo. Non tutti poiché insieme ad IPv4 si sta affermando il protocollo IPv6.

In generale quando vengono realizzati protocolli vengono inseriti bit o gruppi di bit non utilizzati, per permettere ad evoluzioni future di cogliere circostanze non ancora rappresentate dal protocollo.

La convenzione per rappresentare i pacchetti in RFC è di rappresentare i pacchetti striscia per striscia, di lunghezza di 32 bit, chiamate *long word*. I campi versione e ihl, *Internet Header Length* sono entrambi di 4 bit. Il primo campo che si trova è la versione del protocollo che ha generato il pacchetto, è possibile che venga utilizzato da versioni successive, ma questo non è possibile per i pacchetti IPv6. Questo campo dovrebbe permettere la migrazione tra due versioni del protocollo chiara, lenta e controllata. Dato un protocollo che evolve nel tempo deve essere effettuata un'operazione di *parsing*, strutturando il pacchetto in campi, questo *parser* è diverso per ogni versione. Quest'idea di utilizzare un campo versione viene utilizzata anche per altri pacchetti di rete. Il campo ihl contiene la lunghezza dell'*header* espressa in *long word*, al minimo pari a 5. Per cui dei 1500 byte massimi del pacchetto, 20, al minimo, vengono utilizzati dall'*header*, i restanti 1480 sono disponibili a protocolli di livello superiore.

Il campo *type of service* specifica la priorità del datagramma rispetto ad altri, ed è diviso in due parti DSCP, *Differentiated Services Code Point*, e ECN, *Explicit Congestion Notification*, utilizzati per definire la priorità del pacchetto. Quando i pacchetti arrivano in una macchina sono tutti uguali, ma per differenziarli tra di loro si utilizza questo campo per indicare la priorità del pacchetto in modo che gli apparati intermedi siano in grado di effettuare le adeguate operazioni. Questi campi priorità vengono specificati da chi ha realizzato quel pacchetto di livello 3, ma dipende dal provider. Non tutti i provider sfruttano questa possibilità, se la sfruttano non permettono all'utente di indicarla. I provider che utilizzano questo strumento possono realizzare offerte commerciali utilizzando questo approccio, assegnando livelli di priorità diversi. Per l'utenza domestica questo tipo di offerta non è convenzionale, ma tipicamente è molto comune per aziende. La possibilità di rispettare la priorità può essere garantita solamente dagli apparati di loro appartenenza, i *router*, ma questo non è garantito per tutto il percorso del pacchetto attraverso la rete. Quando questo pacchetto con priorità arriva in un'area geografica gestita da un altro ISP, questo non ha obbligazioni commerciali rispetto all'utente, e quindi questi pacchetti vengono gestiti indipendentemente dalla loro priorità. Quindi queste offerte sono valide solamente per pacchetti che rimangono all'interno della zona di interesse del ISP.

I campi *ident*, *flag* e *fragment offset* vengono utilizzati frammentare il pacchetto. Il livello quattro per realizzare pacchetti di una certa dimensione, richiede al livello tre che a sua volta richiede al livello due il valore massimo di MTU, *Maximum Transmission Unit*, la dimensione massima di un pacchetto. Quindi il livello di trasporto realizza pacchetti in base alla tecnologia immediatamente sottostante. Ogni livello due attraversato da uno stesso pacchetto potrebbe avere diversi valori di MTU, quindi il pacchetto di livello tre potrebbe essere troppo grande ed è possibile sia necessario frammentarlo. Questi frammenti attraversano gli is, senza essere rilevati come tali, e verranno riassemblati solamente all'arrivo all'es. IPv6 effettua una scelta radialmente diversa per ilt trattamento dei frammenti, ed alcune di queste scelte migliori vengono implementate anche da IPv4.

In generale la frammentazione consiste nel suddividere uno stesso pacchetto in più pacchetti di dimensione minore, per essere successivamente ricostituito sulla macchina destinazione. Occorrono dei campi nel pacchetto per riconoscere i pacchetto come frammento, riconoscere frammenti generati dallo tesso pacchetto e deve essere possibile determinare l'ordine dei frammenti per poterli riassemblare. In caso il frammento di livello due deve essere nuovamente frammentato si specifica l'*offset* rispetto al pacchetto originale per ogni nuovo frammento, dato l'*offset* del pacchetto originale. Rappresenta la posizione del primo byte del frammento nel pacchetto originale. Questo valore è univoco per ogni frammento, indipendentemente dal numero di riframmentazioni. Il campo *ident*, di 16 bit, identifica un datagramma e serve all'es per determinare il pacchetto di appartenenza del frammento. Se il pacchetto non è frammentato questo rappresenta bit sprecati. Il campo *offset* di 13 bit può specificare solo un *offset* multiplo di otto, con la convenzione che tutti i frammenti sono multipli di otto, in seguito nel campo *flag* di 3 bit, si specifica se il pacchetto può essere frammentato. Il primo bit deve essere riservato e sempre pari a zero, i seguenti due bit chiamati DF, *Don't Fragment*, e MF, *More Fragment*, indicano se il pacchetto può essere frammentato, il primo, oppure se viene seguito da altri frammenti, il secondo. Il campo MF quindi permette di riconoscere l'ultimo frammento di un pacchetto. Se un pacchetto non può essere frammentato, allora si preferisce scartarlo. Oltre all'identificatore per stabilire l'univocità del pacchetto si può utilizzare l'indirizzo del mittente.

Sull'ultimo frammento si può inserire un numero di byte arbitrario, poiché non è necessario l'*offset*, ma solamente la specifica nel campo MF.

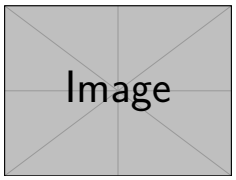

Fig. 33: Frammentazione IPv4 pdu

Il campo *total length*, di 16 bit, indica la dimensione massima del pacchetto, al massimo di 65553 byte, ma molto spesso è frammentato e minore di questo valore. Il campo *time to live*, di 8 bit, rappresenta il tempo di vita del pacchetto, e quando raggiunge zero viene scartato, ed il *router* invia una avvertimento al mittente. Il valore contenuto viene decrementato ad ogni salto o *hop*. Quindi

un pacchetto in *internet* può effettuare al massimo 255 salti, il valore massimo contenuto nel campo ttl. Nonostante la dimensione della rete *internet*, l'elevata connettività permette di effettuare pochi salti. Viene imposto un limite per impedire che alcuni pacchetti entrino in loop. Non deve essere possibile in nessun caso che un pacchetto giri in eterno sulla rete.

Il campo *protocol* indica il protocollo di livello superiore a cui è indirizzato il contenuto del campo dati del pacchetto, i possibili valori vengono definiti dalla IANA. Molto spesso il pacchetto di livello superiore contenuto è di protocollo TCP, con valore 6. Il campo *header checksum* do 16 bit è analogo al campo RCS del livello due, riguarda solamente l'intestazione e deve essere ricalcolata ad ogni hop, poiché cambia il contenuto del campo ttl. Le tecnologie più recenti permettono di non essere calcolato nuovamente ad ogni hop.

In seguito si specificano il mittente ed il destinatario del pacchetto. Si possono specificare campi addizionali relativi a possibili opzioni, questi devono obbligatoriamente costituire multipli di 32 bit, per cui si può aggiungere del *padding* per allinearsi ad una *long word*. Ogni opzione comincia con un byte che ne specifica il codice. Queste opzioni possono specificare quanto sia segreto il pacchetto, il *source routing* ovvero il cammino completo. Si può specificare una sequenza di stazioni intermedie da attraversare obbligatoriamente, opzione meno stringente del *source routing*. Si può indicare che ogni *router* attraversato debba specificare il proprio indirizzo e/o un *timestamp*. Non è obbligatorio per gli is di rispettare queste opzioni, poiché sono operazioni aggiuntive che richiedono calcolo ulteriore e potrebbero rallentare il meccanismo di inoltro.

Questi campi costituiscono l'*header* del pacchetto, in seguito vengono inseriti i dati.

4.4.2 Indirizzamento IPv4

Gli indirizzi IPv4 sono associati alle schede di rete e non alle macchine *host*, sono di 4 byte ed univoci a livello mondiale. Per comodità sono rappresentati da quattro numeri decimali, separati da punti, spesso gli vengono assegnati nomi simbolici.

Non sono assegnati in modo casuale, ma le stesse zone geografiche hanno gli stessi indirizzi, per facilitare la commutazione. Indirizzi sulla stesse rete locale condividono il prefisso, un gruppo di macchine è quindi identificato dal suo prefisso. Nei *router* quindi si possono inserire delle tabelle di indirizzi, raggruppati in *net*. Indirizzi nella stessa *net* sono raggruppati su una stessa rete fisica.

La convenzione consiste di completare i prefissi con tutti zeri per realizzare un indirizzo IPv4 legittimo da poter utilizzare. Questo valore ottenuto si utilizza per denotare l'intera *net* e LAN, e non può essere utilizzato per indirizzare le interfacce della LAN.

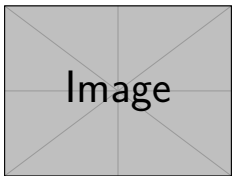

Fig. 34: Tabelle di Instradamento IPv4

L'indirizzo IPv4 ha due funzioni, di identificare un'interfaccia nella rete e di locazione di gruppi di indirizzi in posizione geografiche omogenee nella rete. Nei primi anni di *internet* gli indirizzamenti IPv4 erano divisi in cinque classi, con prefissi di diversa lunghezza, rappresentate da cinque lettere A, B, C, D, E. Questo tipo di indirizzamento viene chiamato *classfull*. Se l'indirizzo iniziava con il primo bit a zero, allora la sua parte prefisso si estendeva al primo byte, mentre i restanti consistevano in un identificatore delle macchine *host* sulla LAN. Se il primo bit è uno ed il secondo è uno zero, allora il prefisso si estende al primo ed al secondo byte. Quando il terzo bit è a zero il prefisso si estende fino al terzo byte. Questi tipi di indirizzi si chiamano di classe A, B e C. Gli indirizzi di classe D hanno il terzo bit pari a zero e individuano indirizzi di tipo *multicast*. La classe E con il quarto bit a zero venne riservata invece per usi futuri.

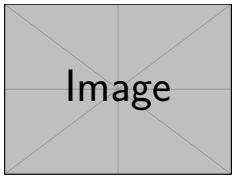

Fig. 35: Indirizzamento IPv4 *classfull*

Il problema di questo approccio consiste nello spreco di byte per la prima classe, poiché non esiste una LAN contenente 2^{24} indirizzi. Per cui si è introdotto l'indirizzamento IPv4 *classless* che utilizza un'altra sequenza di bit accanto all'indirizzo chiamata *netmask*, questo consente una lunghezza arbitraria del prefisso. Questa *netmask* è composta da una sequenza di quattro byte composta da due sequenze contigue di uno, seguita da una composta da zeri. Si specifica una *netmask* dal numero di uno che la compongono. Queste vengono normalmente scritte nella documentazione di rete come numeri decimali, con la stessa convenzione degli indirizzi IPv4.

Poiché tutti gli indirizzi di uno stesso prefisso hanno la stessa *netmask* si può parlare di *netmask* di un prefisso. La *netmask* di un prefisso si può rappresentare come un numero, dopo l'indirizzo, che indica la lunghezza del prefisso, separato da una barra `/`. Due indirizzi IPv4 che si trovano sullo stesso prefisso hanno la stessa *netmask*, generano lo stesso valore se messi in *and* bit a bit con la *netmask* relativa. Due indirizzi appartenenti a diverse LAN producono risultati diversi se messi in *and* bit a bit sulla stessa *netmask*. Questo vale solo se gli indirizzi vengono assegnati in modo che siano tutti disgiunti tra di loro. La responsabilità di assegnare gli indirizzi appartiene ad organizzazioni internazionali come la RIPE.

La configurazione delle interfacce degli es prevede la specifica della *netmask* della *net* di appartenenza, per cui mettendo in *and* l'indirizzo del destinatario, l'es è in grado di identificare se si tratta di un indirizzo locale o remoto. Se il destinatario è locale viene inviato tramite trasmissione diretta al destinatario, se è remoto invece, viene inviato al *default gateway* della LAN, un *router* particolare. Il problema in questo approccio è che non si conosce la *netmask* del destinatario e si ricava il prefisso del destinatario dalla *netmask* dell'es, infatti si potrebbe ricavare il prefisso errato applicando l'*and* bit a bit tra l'indirizzo del destinatario e la *netmask* locale. L'ipotesi che gli indirizzi vengono assegnati in modo disgiunto permette di risolvere questo problema di ricavare il prefisso sbagliato. L'unico caso che quest'ipotesi non risolve è quando il prefisso calcolato del destinatario è diverso da

quello reale, ma comunque diverso da quello locale. Questo non rappresenta un problema poiché il prefisso individuato è remoto, quindi verrà inviato indipendentemente al *default gateway*. Non da errore dal punto di vista della località o non località del destinatario di un pacchetto.

Dentro ad un *router* è presente una tabella di instradamento che controlla tramite un'azione chiamata IP *lookup*. Questa tabella contiene singoli prefissi con la relativa *netmask*, che vengono confrontati bit a bit con l'indirizzo destinatario, sulla relativa *netmask*. Non appena si rileva un riscontro il pacchetto viene inviato su quella linea, altrimenti il pacchetto viene scartato. I prefissi vengono ordinati rispetto alla loro lunghezza. Sull'ultima riga 0.0.0.0/0, chiamata rotta di *default*, non sempre presente, viene prodotto un riscontro con ogni indirizzo. Se è presente quindi nessun pacchetto può essere scartato. Si utilizza questa rotta di *default* poiché la tabella di instradamento non può contenere tutti i prefissi dell'intera rete. Oltre alla linea di inoltro rappresentata dall'interfaccia del *router*, è presente un altro campo chiamato *next-hop* che specifica a quale *router* inviare il pacchetto in base all'indirizzo del destinatario, per distinguere pacchetti destinati alla stessa linea di inoltro. Tuttavia sono presenti eccezioni notevoli su questo schema.

Le operazioni eseguita da un es per l'inoltro di un pacchetto possono essere pensate in termini di accesso ad una tabella di instradamento. Può essere considerato come una macchina con una tabella di instradamento con informazioni veramente essenziali. Su ogni macchina l'interfaccia lo di *loopback* si identifica con l'indirizzo `127.0.0.0/8`, ed il pacchetto "rimbalza" e torna indietro. Tramite quest'interfaccia due applicazioni su una stessa macchina possono comunicare disaccoppiandole fortemente, quindi senza utilizzare l'API, solamente attraverso dei pacchetti inviati sulla rete. Questo indirizzo è un prefisso, non è un vero indirizzo, l'indirizzo attribuito a quest'interfaccia di loopback corrisponde di default al primo indirizzo disponile: `127.0.0.1`. si possono inserire più interfacce di *loopback* su una stessa macchina, sempre all'interno dello stesso prefisso.

Network	Netmask	Interfaccia	Next-Hop
Prefisso dell'es	Netmask dell'es	`eth0`	Direct Connection
`127.0.0.0`	`255.0.0.0`	lo	Direct Connection
`0.0.0.0`	`0.0.0.0`	`eht0`	Rotta di *Default*

Alcuni indirizzi hanno un significato particolare e sono dedicati. Per cui non è possibile assegnare indirizzi contenenti solo `0` o `255`. L'indirizzo composto da tutti `1`, dopo il prefisso rappresenta un indirizzo *broadcast* sulla LAN, per questo non si può utilizzare solamente uno all'interno di un indirizzo. Indirizzi di prefisso `10.0.0.0/8` sono indirizzi privati non validi nella rete, che possono essere utilizzati per comunicare all'interno di una LAN, possono essere associati ad indirizzi validi per comunicare verso l'esterno. L'indirizzo `172.16.0.0/12` rappresenta macchine visibili solo localmente, analogamente indirizzi `192.168.0.0/16` sono indirizzi utilizzabili nella comunicazione tra macchine nella stessa LAN. Il provider trasforma questi indirizzi privati in indirizzi validi per comunicare in *internet*.

In un processo chiamato *sub-netting* è possibile spezzare l'indirizzo privato `10.0.0.0/8` per permettere di comunicare tra di loro a varie macchine di una stessa azienda o organizzazione.

Le organizzazioni che assegnano gli indirizzi utilizzano un meccanismo che divide gli indirizzi in grandi pezzi da `/8`, assegnati ad aree geografiche. Delle organizzazioni delegate ad ogni area geografica assegnano gli indirizzi a chi lo richiede:

- ARIN, *American Registry for Internet Numbers*: per il Nord America;
- RIPE, *(Resèaux IP Europèen) Coordination Center*: per l'Europa;
- APNIC, *Asia Pacific network Information Center*: per il Pacifico e l'Asia;
- AfriNIC, *Africa NIC*: per l'Africa;
- LACNIC, *Latin American and Caribbean NIC*: per l'America Latina ed i Caraibi.

Quando finiscono la loro sezione di spazio di indirizzi, richiedono all'IANA, *Internet Assigned Numbers Authority* l'organizzazione globale per assegnare gli indirizzi IPv4, ulteriore spazio. Attualmente questi indirizzi si stanno esaurendo quindi queste organizzazioni potrebbero non essere in grado di assegnare gli indirizzi richiesti. Attualmente l'IANA ha assegnato tutti i possibili indirizzi `/8`, quindi ulteriori indirizzi devono essere assegnati dalle organizzazioni delegate per ogni area geografica effettuando operazioni di *sub-netting*.

Gli es tendono ad avere più schede di rete, sulla stessa macchina, ma anche se sono presenti più schede l'IPv4 ne utilizza una sola alla volta, mentre le altre sono inattive. Questo poiché alla sua creazione, non era immaginabile che una stessa macchina potesse contenere più schede di rete. Per cui i costruttori di macchine inseriscono una gerarchia tra le varie schede di rete. Nonostante questo molte applicazioni cercano comunque di utilizzare più schede di rete in parallelo su un singolo es.

4.4.3 Il Protocollo ARP

Il protocollo ARP di terzo livello, definito nella RFC 826 risolve il problema di associare indirizzi MAC ad indirizzi IPv4.

Se bisogna inviare un pacchetto all'intero di una rete locale bisogna effettuare una consegna diretta, ovvero spedire il pacchetto direttamente alla macchina del destinatario. Questa consegna viene effettuata specificando al livello 2 l'indirizzo MAC del destinatario, ma è noto solamente l'indirizzo IPv4, quindi si utilizza il protocollo ARP per identificarlo. ARP invia un pacchetto *broadcast*, ARP *request*, su tutte le macchine della LAN, a questo risponderanno solamente le macchine aventi quell'indirizzo MAC. Tutte le macchine sulla LAN sono costrette ad aprire i pacchetti, quindi viene generata un'interruzione *hardware* e deve essere gestita dal sistema operativo anche se il pacchetto non appartiene alla macchina. Ogni scheda di rete è costretta ad analizzare ed aprire i pacchetti che arrivano ad essa, uno dei principi dietro agli attacchi DOS. Infatti se eccessivo potrebbe provocare la perdita di prestazioni della rete locale. Questo processo provoca un'interruzione a bordo di tutti gli *host* della LAN, anche attraverso gli *hub* e gli *switch*; i pacchetti ARP fermati solamente dai *router*.

Per evitare questo processo costantemente ed intasare la rete locale si utilizza un processo di *caching*, dove gli indirizzi noti vengono memorizzati. Quando un *host* riceve un pacchetto ARP *request* dedicato a sé stesso invia un'ARP *reply*. Gli *host* che utilizzano ARP hanno a disposizione una *cache* dove si possono salvare queste corrispondenze tra IPv4 e ARP.

Su una macchina Linux il comando `arp -a` permette di visualizzare la lista di queste associazioni.

Questi pacchetti sono contenuti direttamente nel pacchetto di livello 2, quindi sono presenti gli indirizzi IPv4 e MAC dei destinatari e mittenti.

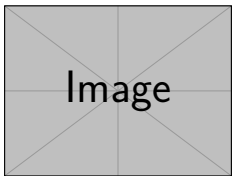

Fig. 36: Struttura ARP pdu

I campi *hardware* e *protocol* specificano il tipo di indirizzi di livello due e di livello tre, mentre i campi hlen e plen specificano la lunghezza dei due indirizzi. Venne definito in questo modo modulare poiché si pensava che la lunghezza degli indirizzi sarebbe cambiata. Si utilizzano quindi campi di lunghezza variabile per memorizzare questi indirizzi che possono variare. In IEEE 802 hlen è di 48 bit, mentre plen in IPv4 è di 32 bit.

Nel campo *operation* è specificato se il pacchetto è una *request* o *reply*. I campi *sender* ia e ha sono rispettivamente l'indirizzo livello due e tre del mittente. Il campo *target* ia contiene l'indirizzo di livello tre della macchina di cui si vuole conoscere l'indirizzo di livello due, solo nel caso di una *request*.

Mentre se l'indirizzo di livello tre è esterno alla LAN, il protocollo ARP allo stesso modo deve individuare l'indirizzo MAC del *router*. Questo protocollo ARP appartiene al livello tre della pila ISO-OSI, ed è un protocollo di supporto per l'IPv4. Quando un pacchetto viene ricevuto da un *router*, vengono effettuate le stesse operazioni descritte per gli es, per individuare l'indirizzo MAC della macchina. Ogni salto di un pacchetto comporta una ARP *request*, una ARP *reply* e l'invio del pacchetto, se l'indirizzo MAC non è contenuto nella *cache* della macchina *host*.

4.4.4 Il Protocollo ICMP per IPv4

Il protocollo ICMP, *Internet Control Massage Protocol*, per IPv4 realizza due operazioni per le reti, ed ha un approccio *best effort*, effettua dei tentativi al meglio delle sue capacità prima di scartare il pacchetto.

Provvede ad un piccolo servizio di rilevazione degli errori, inoltre consente di inviare e richiedere pacchetti ed ottenere informazioni. Mentre ARP viaggia direttamente dentro il livello 2, i pacchetti ICMP viaggiano dentro pacchetti IPv4. Questo protocollo fornisce informazioni di controllo e non offre direttamente servizi di consegna di pacchetti, si comporta quindi come un protocollo ausiliario di IPv4.

Le regole sul comportamento di questo protocollo vennero definite nella RFC 792. La prima regola impone che nessun messaggio ICMP viene generato a seguito di rilevati errori su altri messaggi ICMP. La seconda regola impone che se il pacchetto viene frammentato, solo il primo frammento può generare un messaggio di errore ICMP. La terza regola impone che i pacchetti *broadcast* e *multicast* non generano pacchetti ICMP. Si utilizzano queste regole per non generare eccessivo traffico sulla rete.

Il pacchetto ICMP viaggia all'interno del campo dati di un pacchetto IPv4, poiché se viene sollevato un errore, deve poter tornare al mittente, quindi deve necessariamente viaggiare dentro un pacchetto IPv4, per conoscere l'indirizzo di ritorno.

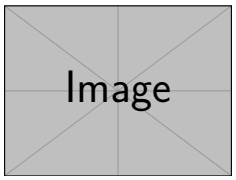

Fig. 37: ICMP pdu

Esistono diversi tipi di pacchetti di errore che ICMP può generare. I messaggi di primo tipo sono messaggi di *destination unreachable*, in questo caso i pacchetti vengono scartati poiché la destinazione non è raggiungibile per diversi motivi:

- *Network Unreachable*: un *gateway* vede la rete dov'è destinato il pacchetto a distanza infinita; ovvero un *router* sta entrando nella tabella di instradamento e l'indirizzo non effettua alcun *match*;
- *Host Unreachable*: l'*host* a cui è destinato il pacchetto non risponde ad una chiamata ARP. Il *router* a cui è arrivato il pacchetto tenta di effettuare una chiamata ARP per inviare direttamente il pacchetto, ma questa fallisce quindi scarta il pacchetto;
- *Protocol Unreachable*: dentro il pacchetto IPv4 è presente un campo che identifica il protocollo a cui va mandato il pacchetto, se questo protocollo non è conosciuto allora viene sollevato questo messaggio di errore;
- *Port Unreachable*: la *port* a cui è spedito il pacchetto non è raggiungibile;
- *Fragmentation Needed and DF Set*: se il pacchetto non può essere frammentato, ma deve essere frammentato per passare attraverso la MTU, quindi viene scartato.

Altri messaggi sono di tipo tempo scaduto, se il valore del ttl arriva a zero, il pacchetto viene scartato con un errore di tipo `TIME_EXCEEDED`. Un messaggio di redirezione può essere inviato da un *router* per indicare dove bisogna reindirizzare il traffico, indicando all'es una strada più favorevole. Messaggi di *echo* rappresentano risposte e richieste. Possono essere `ECHO_REQUEST` ed `ECHO_REPLY`, una richiesta ed una relativa risposta di *echo* per controllare la raggiungibilità di un *host*. Oppure possono essere dei `TIMESTAMP_REQUEST` e `TIMESTAMP_REPLY`, come `ECHO`, per fornire informazioni sull'orario di invio e sulla misurazione della velocità del collegamento, ed un'approssimativa sincronizzazione.

Il comando `ping`, seguito da un indirizzo, permette di effettuare quest'ultima operazione: invia un pacchetto ICMP di tipo *echo* ad una macchina e questa, se è raggiungibile, tenta di inviare i pacchetti di ritorno ICMP.

```
\prompt> ping 127.0.0.1
```

Poiché l'indirizzo specificato è l'indirizzo di *loopback* nessun pacchetto dovrebbe venire perso, e vengono ricevuto con un ritardo pressoché nullo. Questo comando è disponibile nella quasi totalità

dei sistemi operativi moderni, ed ha l'obiettivo di verificare se un dato indirizzo IP è raggiungibile ed il ritardo necessario per raggiungerlo.

Quando termina il comando di ping viene visualizzato un report che indica il numero di pacchetti inviati, persi, il tempo totale impiegato, ed il tempo di andata e ritorno in media, al massimo e la deviazione media. Il comando ping è realizzato con una `fork()`, dove il primo processo si occupa di lanciare i pacchetti, mentre il secondo riceve i pacchetti e calcola le statistiche. Quando i pacchetti vengono inviati o ricevuti vengono memorizzati. I pacchetti inviati si possono visualizzare con il comando `tcpdump` specificando solamente i pacchetti di tipo ICMP:

```
\prompt> tcpdump -n "icmp"
```

Se si effettua un ping su un indirizzo *broadcast*, considera solo la risposta della prima macchina, mentre tutti gli altri messaggi di ritorno provenienti dalle altre macchine vengono considerati come duplicati. Per terminare l'esecuzione del comando si utilizza la sequenza "Ctrl + C".

Il comando traceroute è disponile nella quasi totalità delle macchine, attraverso questo comando si vuole determinare quali sono i *router* effettivamente attraversati per raggiungere una destinazione IP. Quest'operazione viene eseguita inviando pacchetti di ttl molto basso, incrementandolo ad ogni pacchetto, in modo che i *router* che scartano il pacchetto inviano il pacchetto `TIME_EXCEEDED` alla macchina mittente. In questo modo si può ottenere progressivamente la sequenza di *router* che il pacchetto deve attraversare per arrivare all'indirizzo IP destinatario. Nella commutazione a datagramma, i diversi pacchetti potrebbero attraversare strade diverse, e quindi i vari pacchetti della sequenza potrebbero restituire *router* appartenenti a strade diverse. Per cui non è garantito che l'informazione così ottenuta sia corretta. Sono sicuramente corretti per i percorsi dei singoli pacchetti, ma non è garantito che il percorso finale ottenuto rappresenti la sequenza di *router* effettivamente attraversata da ogni pacchetto.

Il comando nei sistemi Linux si chiama `traceroute`, e `tracert` su Windows, seguito dall'indirizzo IP di destinazione. Il tempo di ritardo è variabile poiché i *router* non hanno come compito principale l'inoltro dei messaggi di errore. Quindi quando il *router* non sta inoltrando altri pacchetti può gestire i pacchetti scartati ed inoltrare i pacchetti ICMP. Di seguito una possibile implementazione in pseudocodice del comando:

```
traceroute(IP)
    for(x = 1; replyIP != IP; x++)
        ttl = x
        for(n = 1; n < 3; n++)
            sendRequest(ttl)
            if(reply == TIME_EXCEEDED)
                print(rtt, replyIP)
            else
                print("*")
```

Quando i pacchetti di risposta non vengono rilevati dopo un tempo definito di pochi secondi, vengono segnati tramite il carattere `*`. I *router* che non inviano i pacchetti ICMP sono *router* che non si vogliono mostrare alla rete, quindi scartano i pacchetti senza inviare risposte ICMP. Questi

router generalmente si trovano tra i primi hop, di accesso diretto alla rete dell'ISP. Il contenuto dei pacchetti inviati è arbitrario, poiché sono destinati ad essere buttati. Ma generalmente si utilizzano due tipi di pacchetti, una `ECHO_REQUEST`, a cui il destinatario risponde con una `ECHO_REPLY`, oppure viene inviato un pacchetto UDP, un protocollo di livello quattro.

Effettuando un'operazione `tcpdump` si possono osservare a gruppi di tre i pacchetti inviati con i rispettivi ttl. Studiare le reti corrisponde a studiare le varie apparecchiature e gli strumenti di supporto come questi appena descritti che popolano le reti.

4.4.5 I Protocolli IPv6 e ICMPv6

Gli indirizzi IPv4 si stanno esaurendo, già nel 2010 la IANA ha allocato le ultime `/8` disponibili attribuendole ai registri continentali. Alla fine del 2020 il registro europeo ha allocato i suoi ultimi indirizzi disponibili. Gli ultimi indirizzi sono essenzialmente esauriti, ma ci sono registri più avanti di altri nella distribuzione. IPv6 è alternativo rispetto a IPv4, per loro natura realizzano due reti disgiunte, che non possono comunicare tra di loro. IPv6 ha varie caratteristiche interessanti, inoltre ha indirizzi di 128 bit, e sta progressivamente sostituendo gli indirizzi IPv4. IPv5 non è mai esistito, poiché venne realizzato un protocollo specifico dal punto di vista applicativo, che utilizzava un nuovo protocollo di livello 3 il cui nome poteva sembrare simile ad IPv5 quindi per evitare confusione venne chiamato IPv6, "saltando" la versione 5. IPv4 e IPv6 sono destinati a convivere per anni, entrambi protocolli di livello tre, i loro pacchetti vengono gestiti separatamente dagli es ed is e per mantenere la rete attiva continueranno ad essere supportati ed utilizzati, anche all'avvento di nuovi protocolli di livello tre.

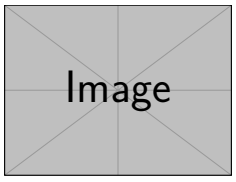

Fig. 38: Struttura IPv6 pdu

Un pacchetto IPv6 ha un *header* di struttura fissa di 40 byte, per evitare di creare problemi ad eventuali *router* attraversati. Alcune caratteristiche di IPv4 spariscono o cambiano nome e vengono introdotti nuovi campi. Viene rimosso il campo per le opzioni. Il campo *ver* versione di 4 bit, di valore 6; analogo al campo *version* nei pacchetti IPv4. Non vengono distinti in base a questo campo da IPv6, ma dal campo LLC. Il campo *traffic class* di 8 bit ha sostanzialmente le stesse funzioni di *type of service* di IPv4. Il campo *flow label* di 20 bit viene utilizzato per distinguere un flusso, ma ancora non ha uno scopo definito. I *router* attraversati dovrebbero gestire pacchetti dello stesso flusso allo stesso modo. È un campo il cui utilizzo non è ancora chiaro a livello applicativo, ma è utile per certi aspetti di sicurezza. Il campo *payload length* di 16 bit è analogo al campo *length* in IPv4. Il campo *next header* di 8 bit individua il protocollo di livello superiore contenuto, simile al campo *protocol* in IPv4. Si utilizza in IPv6 anche per specificare delle opzioni inserendo una codifica per identificarle all'interno del pacchetto. Il campo *hop limit* è il ttl, di 8 bit, come in IPv4.

Seguono i campi degli indirizzi di mittente e destinazione di 16 byte. I pacchetti IPv6 non possono essere frammentati, quindi non sono presenti i campi corrispondenti alla frammentazione in IPv4. Se questi pacchetti incontrano un *router* con un MTU minore vengono scartati. La frammentazione deve essere effettuata a livello degli es, esistono in IPv6 protocolli che individuano in un cammino la MTU minima, e quindi l'es è in grado di creare pacchetti di lunghezza adeguate per passare attraverso l'intero percorso. Si suppone che i controlli di livello 2 e 4 siano sufficienti, quindi sparisce la *checksum* di IPv4, non è presente il campo che specifica la lunghezza dell'intestazione, poiché la lunghezza è fissa in IPv6. In termini di intestazione è un protocollo semplice, rispetto ad IPv4.

Gli indirizzi IPv6 sono molto diversi da IPv4, avendo molti bit in più sarebbe più difficile rappresentarli in decimale, quindi vengono rappresentati in 8 numeri esadecimali da quattro cifre ciascuno separati da due punti, ogni numero quindi rappresenta 16 bit. Questa rappresentazione consente di non effettuare conversioni di base, da binario a decimale e viceversa.

La scrittura può essere semplificata, rappresentando zeri consecutivi con un singolo zero, solo se antecedenti da altri zeri. Generalmente si omettono gruppi consecutivi di 16 bit contenenti soltanto zeri, si omettono anche i due punti di separazione. Questa notazione è utilizzabile una sola volta all'interno di un indirizzo, per evitare ambiguità. L'interfaccia di loopback è `::1`, e rappresenta l'indirizzo `0:0:0:0:0:0:0:1`, analogo di `127.0.0.1` in IPv4.

Per i prefissi si utilizza la stessa notazione `/` usata in IPv4, con prefissi di lunghezza massima di 64 bit. Una *netmask* non è rappresentata mai nel formato dove sono espliciti i suoi bit, si usa solamente la notazione barra.

Anche in IPv6 ci sono indirizzi *unicast* e *multicast*, ma non sono presenti indirizzi *broadcast*. Si cerca di stabilire con chi si vuole dialogare in maniera più selettiva. Tuttavia questa regola ha delle interpretazioni piuttosto lasche. Sono presenti due tipi fondamentali di indirizzi *unicast* sono *Global Unicast*, indirizzi utilizzabili in *internet*, oppure indirizzi simili agli indirizzi IPv4 privati, con qualche differenza significativa, chiamati *Link-Local*, utilizzabili solo nell'ambito della LAN. In IPv6 indirizzi *Link-Local* sono usati sempre, a differenza degli indirizzi privati in IPv4, e permettono soltanto di dialogare con altre macchine sulla stessa LAN. Nel mondo IPv6 *link* e LAN sono sinonimi, quindi non vengono chiamati LAN-*Local*.

Indirizzi *multicast* importanti sono *Solicited Node*, `ff02::01`, indirizza tutte le macchine in una LAN, essenzialmente un indirizzo *broadcast*, e `ff02::02`, indirizza tutti i *router* in una LAN.

I primi 64 bit di un indirizzo *unicast* sono il prefisso, mentre i seguenti 64 bit vengono chiamati *Interface Identifier*, o *Interface ID*, IID, per identificare univocamente un'interfaccia. Questo comporta uno spreco, poiché anche se su una singola LAN sono presenti poche macchine, vengono comunque utilizzati 64 bit per indirizzarle.

Indirizzi *Global Unicast* vengono assegnati da organizzazioni e registri internazionali, analogamente ad indirizzi IPv4. Indirizzi *Link-Local* hanno un prefisso `fe80::/64`, essendo privati, non possono essere usati per accedere all'esterno della rete locale.

In IPv6 un'interfaccia di rete può avere diversi indirizzi, e gli spazi di indirizzamento attribuiti a due LAN devono essere disgiunti. Gli indirizzi possono essere attribuiti alle interfacce in modo manuale oppure automatico, tramite vari meccanismi *built-in* nel protocollo. Uno di questi è particolarmente interessante e si chiama auto-configurazione *stateless*. Con questo meccanismo una macchina si attribuisce un indirizzo *Link-Local*, senza dialogare con il *router* della LAN. L'inter-

faccia collabora con i *router* per assegnare uno o più indirizzi *Global Unicast* per poter comunicare sulla rete e sceglie i *router* di *default*. Anche in IPv4 sono presenti protocolli simili in modo che una macchina si auto-attribuisca un indirizzo IPv4, ma questi meccanismi provengono da servizi esterni, mentre in IPv6 questi meccanismi appartengono al protocollo stesso.

L'auto-attribuzione dell'*Interface ID* può essere casuale oppure basata sull'indirizzo MAC dell'interfaccia, anche se ha un impatto negativo sulla *privacy*, permette di realizzare indirizzi univoci, dato che lo sono tutti gli indirizzi MAC a livello globale. L'interfaccia divide l'indirizzo MAC in due parti, ciascuna di 24 bit, viene inserita la sequenza di 16 bit `ff:fe` tra le due parti. Al settimo bit della prima parte viene assegnato 1, poiché se l'indirizzo MAC è unico a livello globale quel bit vale 0. Questi 64 bit identificano l'indirizzo nella LAN, anteponendo il prefisso `fe80::/64` per costruire l'indirizzo *Link-Local*. Gli indirizzi MAC studiati si riferiscono ad uno standard EUI-48, e per tradurre indirizzi MAC da 64 bit si utilizza il meccanismo descritto nello standard EUI-64. Se viene utilizzato questo indirizzo per comunicare in *internet*, viene mostrato il proprio indirizzo MAC all'intera rete, questo rappresenta un aspetto molto negativo in merito alla *privacy*. Altrimenti algoritmi di scelta casuale possono essere utilizzati, ed includono meccanismi per evitare collisioni. Prima di considerare questo come indirizzo, svolge un'attività di *Duplicate Address Detection*, controllando se nella stessa LAN è presente un'interfaccia avente lo stesso indirizzo, inviando un pacchetto all'indirizzo appena costruito. Questo indirizzo è *Link-Local*, quindi deve essere unico solamente all'interno della LAN. I *router* presenti sulla LAN inviano periodicamente dei pacchetti di *Router Advertisement* destinati ad `ff02::1`, tutte le interfacce presenti sulla LAN. Le interfacce a loro volta possono sollecitare questi pacchetti tramite pacchetti *Router Solicitation*. Nei pacchetti di *Router Advertisement*, viene fornito un elenco di prefissi utilizzabili sulla LAN, un tempo di validità ed una specifica se può essere utilizzato come *router* di *default*, fornisce altre informazioni utili alle interfacce ed il proprio indirizzo MAC.

Un'interfaccia che riceve questi prefissi può attribuirsi indirizzi ulteriori rispetto a quelli *Link-Local*, anteponendo uno di questi indirizzi al suo *Interface ID*. Esistono protocolli appositi per distribuire prefissi salvati su *server*, ed attribuendoli da remoto ai vari *router* della LAN.

La spedizione di un pacchetto avviene come in IPv4, controllando se il destinatario è sulla stessa LAN, si effettua quindi spedizione diretta o spedizione al *router* di *default*. Le tabelle di instradamento dei *router* hanno lo stesso significato delle tabelle IPv4, e quindi vengono utilizzate allo stesso modo. In IPv6 non c'è il protocollo ARP per individuare l'indirizzo MAC del destinatario. Il protocollo ARP, invia continuamente richieste *broadcast* e quindi comporta un carico elevato sulla LAN. In IPv6 si utilizza invece il protocollo ICMPv6, che svolge le stesse funzioni di ICMP(v4), ed altre funzioni, sopratutto si occupa dei servizi offerti da ARP in IPv4.

Per cercare un indirizzo MAC si spedisce un pacchetto ICMPv6 ad un gruppo di *multicast*, non *broadcast*. Quando un'interfaccia si assegna un indirizzo ICMPv6, assume anche di appartenere ad un uno specifico gruppo *multicast*, chiamato *Solicited Node*. Questo indirizzo è composto da un prefisso di 104 bit seguito dagli ultimi 24 bit dell'IID. I primi 16 bit di questo prefisso sono `ff02`, i seguenti 72 bit sono pari a zero e gli ultimi 16 sono `01:ff:` `ff02:0:0:0:0:1:ff00::/104`. La probabilità di due macchine di avere lo stesso IID sulla stessa rete locale è molto bassa, quindi questi gruppi *multicast* sono molto piccoli, a volte composti da una singola macchina.

Per ottenere un indirizzo MAC di un sistema, un nodo di una macchina calcola dall'indirizzo IPv6 del destinatario il rispettivo IID, e determina il suo gruppo *multicast* di appartenenza. A

questo indirizzo invia un pacchetto ICMPv6 di tipo *Neighbor Solicitation*, specificando l'indirizzo *unicast* IPv6 a cui rispondere. Se il destinatario è presente invia un pacchetto *unicast*, con il suo indirizzo MAC specificato nella parte dati del pacchetto di tipo *Neighbor Advertisement*. Questo indirizzo viene memorizzato nella *Neighbor Cache*, equivalente alla ARP *cache*.

Si utilizza *Solicited Node* poiché un pacchetto *multicast* è considerato solo da un gruppo di macchine e non dall'intera rete. Ma richiede un gruppo *multicast* corrispondente a livello MAC, per evitare di dover inviare pacchetti *broadcast* a livello due invece che a livello tre. Visto che l'indirizzo contiene gli ultimi bit dell'indirizzo è probabile che venga processato da poche schede.

5 Livello 4: Strato di Trasporto TCP e UDP

Da questo livello in poi si possono sviluppare applicazioni, il servizio a questo livello deve essere affidabile, poiché si suppone che la rete lo sia. Le applicazioni infatti assumono sia affidabile.

Offre servizi contesi con una connessione bidirezionale e contemporanea tra le due parti. Le primitive sono di diversi tipi, offerte ad una popolazione molto ampia di utenti-programmatori:

- `listen`: Mette la macchina in attesa di ricevere una richiesta di instaurazione di una connessione;
- `connect`: Tenta di instaurare una connessione;
- `send`: Invia dati;
- `receive`: Riceve dati;
- `disconnect`: Rilascia una connessione.

Il nome della primitiva dipende dal linguaggio di programmazione utilizzato per accedere a questa primitiva.

La contemporaneità nella rete non è effettivamente garantita, è molto improbabile che nello stesso istante due es si comunichino a vicenda, ma sono in grado di ascoltare la rete ed aspettare di ricevere o inviare messaggi. Affinché la connessione sia affidabile, ogni dato inviato viene riscontrato dall'es di destinazione.

Le primitive per l'instaurazione ed il rilascio di connessioni devono essere realizzate in modo affidabile, più facile da risolvere per l'instaurazione che per il suo abbattimento.

I pacchetti scambiati in una connessione sono sempre numerati a entrambi le parti in modo sequenziale, e si possono riscontrare i pacchetti inviati tramite *acknowledgment*, analogamente ai casi precedenti. La numerazione inizia da un numero arbitrario scelto dai due es su cui si basa l'instaurazione della connessione. Questo metodo viene chiamato *three way handshake*.

Dati due calcolatori, il primo sceglie un numero di sequenza arbitrario per il numero dei pacchetti x ed invia una *Connection Request* al secondo con il numero scelto. A questo punto il secondo calcolatore ricevuto il pacchetto sceglie il suo numero iniziale di sequenza y. Questo invia un pacchetto di tipo *Connection Accepted*, contenente x e y. A questo punto il primo calcolatore suppone che questi due valori siano validi ed invia un pacchetto di riscontro al secondo calcolatore con un pacchetto contenente y. Questo pacchetto finale di riscontro dovrebbe essere numerato con il valore di x, ma questa è una questione di convenzioni, che verrà trattata successivamente. Spesso il primo pacchetto dell'*handshake* si chiama *Data*, poiché potrebbe già contenere dei dati scambiati dal primo verso il secondo calcolatore. Questo è sicuramente non vero per il primo ed il secondo pacchetto della connessione.

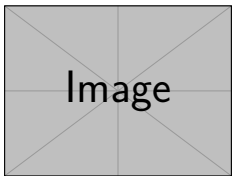

Fig. 39: Instaurazione di una connessione *Three Way Handshake*

Numerare i pacchetti è molto utile, per permettere di riscontare esattamente quali pacchetti sono stati ricevuti.

Sarebbe semplice numerare i pacchetti da zero per entrambi gli es, ma utilizzando numeri diversi è possibile distinguere pacchetti scambiati da due stessi calcolatori, anche in tempi diversi, su connessioni diverse tra di loro.

Rilasciare una connessione è un processo complesso, se non viene effettuato correttamente si potrebbe rischiare la perdita di alcuni dati. Se la connessione viene terminata in modo unilaterale, il rilascio è rudimentale, poiché il secondo calcolatore connesso potrebbe aver inviato pacchetti, nonostante la connessione sia stata interrotta a sua insaputa. Questi pacchetti verranno quindi persi. Si potrebbe utilizzare un rilascio simmetrico, dove entrambi i calcolatori si accordano per continuare ad ascoltera la connessione per un certo intervallo di tempo, per ricevere eventuali pacchetti che andrebbero altrimenti persi. Questo è un problema molto complesso, poiché non è definito in quale istante la connessione si dice completamente terminata.

Questa situazione si può rappresentare in modo semplice con il problema dei due eserciti, o due generali. Si suppone esistano due eserciti, il primo numericamente superiore al secondo, ma diviso in due parti minori, ognuna minore del secondo. Queste due parti sono disgiunte tra di loro, e per poter battere il secondo esercito devono necessariamente attaccare in contemporanea, altrimenti non riuscirebbero a vincere. I generali di queste due parti del primo esercito devono comunicare spedendo emissari tra di loro, ma questi devono attraversare il territorio dell'esercito nemico, e potrebbero essere catturati. Si potrebbero inviare emissari tra le due parti per riscontrare l'arrivo dell'emissario precedente, inviando così un numero arbitrario di riscontri; ma l'ultimo generale ad inviare il suo emissario come riscontro non saprà mai se il messaggio è stato ricevuto. Non esiste infatti protocollo di comunicazione per garantire a questi due generali una vittoria.

Si potrebbe allora inserire un timer, se entro un certo periodo non viene ricevuto alcun pacchetto, allora viene rilasciata la connessione. Si vorrebbe avere un tempo di *timeout* molto ristretto, ma questo rappresenta un'euristica, non garantisce non vengano persi pacchetti.

5.1 *Transmission Control Protocol*: TCP

Il protocollo TCP viene definito negli standard RFC 793, 1122 e 1323. TCP offre un servizio di trasmissione di dati affidabile bidirezionale e contemporaneo, chiamato *full-duplex*, oppure punto-punto, *host-to-host* o *end-to-end*. Connessioni di tipo *multicast* o *broadcast* non sono supportate, può usare protocolli di livello tre diversi, sia IPv4 che IPv6, tra gli altri.

È il primo protocollo a garantire l'affidabilità della connessione, identifica i problemi dei livelli sottostanti, infatti se un pacchetto viene scartato nel percorso lo richiede nuovamente, e se i pac-

chetti arrivano disordinati li riordina. Utilizza meccanismi di *acknowledgment*, ogni pacchetto viene seguito da riscontri. Una connessione non avviene tra due calcolatori, ma tra due processi attivi dentro una macchina. Ogni processo in una connessione è identificato dall'indirizzo IP della macchina, ma questo non è sufficiente, quindi si utilizza un numero assegnato a ciascun processo per distinguerli, chiamato numero di porta o *port*. Una volta stabilita la connessione, i dati allo strato TCP del mittente vengono inviati correttamente allo strato TCP del ricevente ed offerti al processo destinazione. Dal punto di vista dell'applicazione la rete non esiste, si interfaccia solamente con il livello TCP del destinatario. Le *port* sono i TCP sap, sono numeri da 2 byte, non possono essere minori di 1024, poiché riservati a servizi standard, anche se molti altri numeri sono riservati ad altri servizi con un *port* superiore. Un processo potrebbe chiedere al protocollo un certo numero di *port* specifico da utilizzare e se non è già utilizzato gli viene assegnato.

I pacchetti TCP si chiamano *segment*, segmenti, costituiti da un intestazione e dati opzionali. C'è un limite per i dati IP di 65553 byte, un altro limite più stringente è dato dalle MTU. I byte a disposizione dell'applicazione per ogni pacchetto sono 1460 byte, poiché al minimo 20 sono riservati all'intestazione del livello quattro, ed altri 20 riservati all'intestazione IPv4. I pacchetti non sono numerati, ma vengono numerati i byte dei pacchetti con un numero di sequenza di 32 bit, e consente di avere *reset* dei numeri di sequenza infrequenti, rappresenta una sequenza circolare. Questi vengono riscontrati byte per byte utilizzando *acknowledgment*.

I segmenti vengono utilizzati per instaurare connessioni, spedire dati, spedire *acknowledgment* o chiudere connessioni.

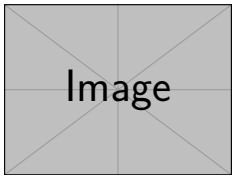

Fig. 40: Struttura TCP pdu

Ogni riga è composta da 4 byte. Nella prima sono presenti il numero di porta del sorgente e del destinatario, la seconda contiene il numero di sequenza del byte spedito, la terza il numero di sequenza di *acknowledgment*. Se si usasse un intestazione per spedire ogni byte sarebbe troppo inefficiente, quindi questo numero di sequenza di byte si riferisce al primo byte del campo dati. Questi byte vengono numerati a partire da questo numero di sequenza specificato. L'*acknowledgment* specifica l'ultimo byte che è stato riscontrato. Nella quarta riga, il campo hlen specifica la lunghezza dell'intestazione, res è un campo riservato, il campo *code* contiene la funzione del pacchetto, il campo *windows* regola il controllo di flusso, lungo 2 byte. Nella quinta riga è presente una *checksum*, importante per capire se il pacchetto è integro, segue un *urgent pointer* per specificare un dato importante da leggere appena viene ricevuto il pacchetto, entrambi di due byte. Segue una riga con le opzioni ed eventuale padding.

Quando si riscontra un numero di sequenza in un *acknowledgment* si specifica il numero del prossimo byte atteso, questo campo quindi si riferisce ai dati che viaggiano in direzione opposta.

Il numero di sequenza inoltre stabilisce la posizione del pacchetto nel flusso di dati generato in trasmissione.

Il nome del pacchetto per la richiesta dell'instaurazione della connessione si chiama *syn*, mentre *ack* per il pacchetto di riscontro. Il significato applicativo dei byte trasportati dal protocollo TCP non è noto al protocollo stesso, o a qualunque protocollo di livello quattro, questi verano interpretati da protocolli di livello superiore.

Il campo *code* determina la funzione del pacchetto in base a quali tra questi bit è attivo, ovvero assegnato ad uno:

- URG: Indica che il campo *urgent pointer* è valido;
- ACK: Indica che il campo contenente l'*ack* è valido, ovvero il riscontro contenuto deve essere considerato. Sempre valido, tranne nel primo pacchetto del *three way handshake*;
- PSH: Specifica che il pacchetto deve essere inviato velocemente;
- RST: Specifica di voler resettare la connessione;
- SYN: Indica che si vuole sincronizzare i numeri di sequenza ed il pacchetto prende il nome di *syn*;
- FIN: Indica che il mittente ha raggiunto la fine del byte stream e rilascia la connessione.

La *checksum* intessa l'intero segmento, gli indirizzi IP ed il campo *protocol* dei pacchetti al livello tre. Questa *checksum* provoca un'invasione della gerarchie sui campi del terzo livello. Le opzioni specificano la negoziazione sulla massima ampiezza del campo dati nell'instaurazione della connessione. Gli *host* sono obbligati ad accettare almeno 536 byte.

Per ogni connessione TCP utilizza dei *buffer* in trasmissione ed in ricezione. Questi *buffer* sono porzioni di memoria disponibile e vengono gestiti come delle code. Quando si utilizza una primitiva di invio, vengono inseriti i dati da inviare all'interno di questo *buffer* e verranno inviati quando il protocollo lo riterrà opportuno. In ricezione invece l'effetto della primitiva *receive* corrisponde a leggere nel *buffer* per controllare se sono presenti dati da leggere, già precedentemente ricevuti, ed in caso canella il *buffer*. Questi *buffer* occupano molta memoria, due per ogni connessione TCP, quindi è favorevole a terminare connessioni per liberare memoria disponibile per la macchina *host*.

Il campo *window* indica la dimensione corrente del *buffer* in ricezione, dopo aver ricevuto l'ultimo segmento. Specifica l'ampiezza della finestra di controllo di flusso. Pone un limite ai byte che possono essere inviati all'interlocutore, ed in caso sia zero, impedisce l'invio di ulteriori dati.

Poiché i numeri di sequenza sono degli interi a 32 bit, hanno un limite superiore prima del quale si resettano. Si possono rappresentare come interi modulo 2^{32}, quindi è possibile rappresentarli su una circonferenza, dividendola in vari spicchi per indicare quali sono attualmente utilizzati e per quale scopo.

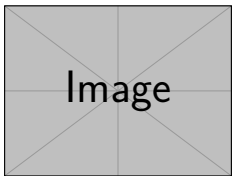

Fig. 41: Spazio dei *Sequence Number* del Ricevente

Un primo spicchio coincide con i byte ricevuti e consumati dall'applicazione, parte dal numero scelto all'inizio della connessione. Il seguente spicchio contiene i dati ricevuti e riscontrati, ma non ancora consumati, fino al numero di sequenza corrente dell'applicazione. Il successivo spicchio contiene i dati ricevuti, contenuti nel *buffer* non ancora riscontrati, quindi contengono numeri di sequenza maggiori del numero di sequenza attuale. In seguito lo spicchio si estende fino a coprire la dimensione del *buffer* in ricezione, con un massimo di $2^{16} - 1$ spazi disponibili. Questo *buffer* contiene gli ultimi tre spicchi, mentre per il mittente contiene gli ultimi due, poiché i byte inviati e non ancora riscontrati non risultano nel *buffer*, e si può basare solamente sui pacchetti precedenti per determinare la sua dimensione. Quindi questa dimensione è maggiore dello spazio attualmente disponibile nella macchina ricevente in un dato istante. Potenzialmente il mittente potrebbe inviare abbastanza byte da esaurire e fuoriuscire dalla dimensione del *buffer*. Ma questo non considera la capacità del processo ricevente di consumare byte.

Per rilasciare una connessione TCP, uno dei due processi invia un segmento con il bit `FIN` attivo, non avendo più dati da trasmettere. Quando questo processo riceve un riscontro, ritiene la sua connessione terminata. Ma i dati possono ancora fluire in verso opposto, vero questo processo che ha chiuso il suo flusso. La connessione si definisce rilasciata solamente quando entrambi i processi hanno rilasciato la connessione. Quando il secondo processo ha terminato i dati da trasmettere, invierà anch'esso un segmento `FIN`, ed al suo riscontro terminerà la connessione. Sono quindi richiesti quattro segmenti per terminare una connessione TCP, ma il primo pacchetto di riscontro, per il rilascio iniziato dal primo processo, può coincidere con il secondo pacchetto di rilascio, iniziato dal secondo processo.

Per risolvere il problema dei due eserciti si aspetta un intervallo di tempo, *timeout* definito come il doppio del ttl stimato per un segmento.

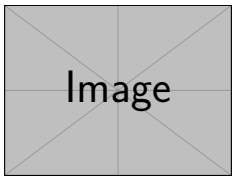

Fig. 42: Diagramma di Transizione di Stato di una Connessione TCP

In una connessione TCP i due processi si definiscono *client* e *server*, generalmente il processo ad instaurare una connessione è il *client* mentre il *server* si rende disponibile. Il processo a terminare

la connessione è il *client*, anche se durante la connessione si era comportato come *server*.

Il processo *client* per instaurare una nuova connessione si deve trovare nello stato `closed`. Inviando un pacchetto `syn` al *server*, che si trova nello stato `listen`, entra nello stato `syn sent`. Il *server* alla ricezione di questo segmento entra nello stato `syn received` ed invia un riscontro. Quando questo viene ricevuto dal *client* entra nello stato `established` per segnalare che ha instaurato la connessione ed invia un riscontro al *server*. Quando viene ricevuto dal *server* anch'esso entra nello stato `established`.

In questo macro-stato corrisponde a molti più stati, come la corretta ricezione dei pacchetti, tramite riscontri e la gestione della finestra di controllo di flusso. In spedizione invece può corrispondere a stati di scelta della spedizione dei pacchetti, per massimizzare la banda e controllare la congestione della rete, e stati di spedizione di copie dei pacchetti in caso di mancato riscontro.

5.2 *User Datagram Protocol*: UDP

Il protocollo UDP fornisce ai livelli superiori un servizio di trasmissione non connesso e non affidabile. Non vengono infatti inviati riscontri di alcun tipo sull'esito della ricezione di pacchetti. Ma al compenso è molto veloce, si utilizza quando non è importante il riscontro dei pacchetti, preferendo l'efficienza, oppure se il riscontro viene gestito a livello applicativo. I pacchetti inviati con questo protocollo prendono il nome di datagrammi.

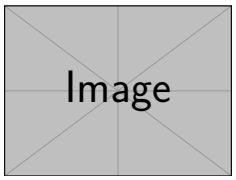

Fig. 43: Struttura UDP pdu

L'intestazione di un pacchetto UDP viene diviso in quattro campi di 16 bit ciascuno. Questi specificano la porta del mittente e del destinatario, la lunghezza del datagramma ed il *checksum*. Queste porte sono analoghe alle porte per il protocollo TCP, e può utilizzare gli stessi protocolli di livello tre.

6 Livelli Applicativi: DNS, HTML, HTTP

6.1 Il *Domain Name System*: DNS

Il DNS o Domain Name System è un applicazione del livello ISO-OSI sopra al livello quattro. Dal livello quattro in poi le applicazioni possono essere programmate dall'utente e non è presente una divisione così netta tra i vari livelli applicativi.

Poiché è scomodo utilizzare un numero per identificare un indirizzo, si vuole assegnare ad ogni interfaccia un nome, più riconoscibile, questo protocollo si occupa della mappatura tra questi nomi assegnati all'indirizzo IP corrispondente. Agli albori dell'*internet* questo "Spazio dei Nomi" o *namespace* era *flat*, ovvero l'insiemi dei nomi ammissibili non aveva restrizioni, quindi poteva essere assegnato qualsiasi nome arbitrario, e la mappatura veniva tracciata da una persona a mano su un *file* testuale. Al crescere della dimensione della rete tuttavia potevano facilmente sorgere conflitti. La possibilità di aggiungere e modificare i nomi necessitava di un accesso a questo *file* di testo centrale costante.

Per migliorare questo servizio si è pensato di decentralizzare l'assegnazione dei nomi e la responsabilità del *mapping* tra nome ed indirizzo. Inoltre la più importante aggiunta è la possibilità di accedere al *mapping* con tecniche *client-server*.

Si utilizza un *database* distribuito per contenere queste corrispondenze, la robustezza e la resistenza di questo algoritmo sono favorite da meccanismi di *caching* e replicazione di questo *database*. Il *namespace* viene partizionato per garantire un controllo efficiente non centralizzato delle assegnazioni. I nomi non sono più sequenze di caratteri, ma sequenze di caratteri separati da punti. I nomi che hanno un suffisso comune possono essere rappresentati attraverso un albero. Il livello più alto della gerarchia, l'autorità *top level* delega la gestione dei nomi ad autorità assegnate a diverse partizioni dello spazio dei nomi. L'autorità *top level* non si occupa delle partizioni interne, e si suppone tutti questi domini siano figli di un dominio radice, cui corrisponde una stringa vuota. Questa partizione può somigliare all'indirizzamento IPv4, ma questo meccanismo di delega applicabile a più livelli può espandere lo spazio, mentre in IPv4 lo spazio è finito e definito a priori.

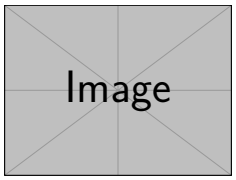

Fig. 44: Gerarchia del *Namespace*

Un dominio è un sottoalbero del *namespace*, il cui nome è il nome della radice del sottoalbero. Le foglie di questo albero sono in corrispondenza di indirizzi IPv4 e IPv6, e rappresentano interfacce. Inoltre è possibile che i nodi intermedi rappresentino ulteriori interfacce. Anche un singolo *host* è un dominio. In questo momento esistono 1500 domini *top level*, all'inizio questa distribuzione era molto ristretta. I domini nazionali sono stati standardizzati con sigle da ISO 3166, e recentemente questa restrizione è stata resa più lasca, ed esistono vari domini di recente definizione, come il

dominio del `cern`. I domini su cui nasce *internet* sono principalmente domini nord americani, come `com`, `edu`, `gov`, etc.

L'organizzazione dei nomi di *internet* viene chiamata *Domain Name System*, i sistemi che realizzano il *mapping* tra indirizzi e nomi sono chiamati *Name Server* (ns). Alcuni hanno la delega per una porzione del *namespace*, e questi sono in grado di dialogare tra di loro. Un ns ha informazioni complete su una parte del *namespace*, detta zona, sulla quale ha autorità. Una zona è un sottoalbero del *namespace*, eventualmente privato di alcuni sottoalberi.

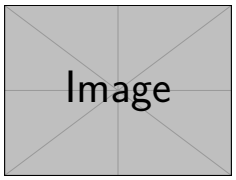

Fig. 45: Zona del *Namespace*

Questi sottoalberi su cui non ha autorità sono altri domini delegati ad altri ns dal *top level*. Un ns può essere autorità di varie zone, inoltre per la stessa zona possono essere associati più ns autorità, per motivi di resistenza ai guasti. Poiché questo sistema è nevralgico per la gestione di *internet*, ed è necessario il suo funzionamento per l'accesso alla rete.

I ns possono essere primari o secondari, i *server* primari contengono le informazioni aggiornate sul *mapping* della zona, mentre i ns secondari richiedono periodicamente il *mapping* della zona al *server* primario. Si lavora su modiche o assegnazioni dei nomi solamente sul ns primario. Un ns può essere contemporaneamente primario per certe zone e secondario per altre. Si hanno vari gradi di libertà sul numero dei ns, sulla possibilità di essere primari o secondari, oppure sulla loro posizione, possono essere anche lontani dalle macchine delle quali conservano i nomi.

6.1.1 Risoluzione

I *client* che usano i ns si chiamano *resolver* e sono all'interno delle macchine *host*. Hanno la capacità di interrogare i ns ed interpretare le loro risposte. Sopratutto sono in grado di inviare le informazioni ricavate ai programmi che li utilizzano. Non necessariamente è un processo autonomo, i *resolver* si possono trovare all'interno di comuni librerie contenute in alcuni *browser*. Il *resolver* dato un nome assegnato ad un certo indirizzo ne restituisce l'indirizzo IP.

Se il *resolver* ha già le informazioni necessarie le fornisce al *client*, altrimenti le chiede direttamente alla radice autorità dello spazio dei nomi, la quale fornisce al *resolver* un ns più specifico, scendendo l'albero fino a raggiungere l'indirizzo specifico. Questi ns assegnati alla radice sono costantemente sotto pressione, poiché l'intera rete si basa interamente su di loro per effettuare il processo di risoluzione.

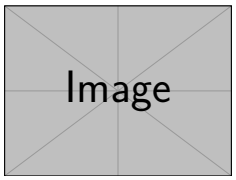

Fig. 46: Processo di Risoluzione

Per la risoluzione sono possibili vari atteggiamenti, può essere ricorsiva oppure iterativa, anche se una risoluzione solamente iterativa non viene mai utilizzata sulla rete. Nella risoluzione ricorsiva il *client* chiede al *server* una risorsa, e pretende come risultato l'indirizzo, se il *server* non la conosce, è suo compito contattare altri *server* per ottenerla. Mentre nella risoluzione iterativa, è il *client* a dover contattare ulteriori *server*, specificati dal *server* iniziale. Nella risoluzione iterativa i *server* non devono memorizzare lo stato della rete, quindi è un approccio più efficiente per i *server*. Questo processo risolutivo può essere completamente iterativo o ricorsivo, oppure sono possibili atteggiamenti misti, dove il *resolver* effettua una *query* ricorsiva al ns e questo effettua delle *query* iterative a vari ns, appunto per rendere più efficiente questo processo. Questo corrisponde a quanto descritto nell'immagine precedente.

Le informazioni memorizzate tra le vari *query* vengono salvate su diverse *cache* dentro i ns, che svolgono il ruolo di autorità per le varie zone, quindi in alcuni casi non è necessario scendere l'intera gerarchia della zona, la *cache* è presente anche al livello di applicazione, come i *browser* quindi può essere non necessario utilizzare il processo *resolver*. Alcuni *server* possono memorizzare informazioni negative, quindi si suppone l'informazione sulla *cache* scade dopo un certo intervallo di tempo. Il tempo di vita delle informazioni in *cache* si chiama ttl, *Time To Live* e viene scelto con opportune metriche per un compromesso tra consistenza ed efficienza, non è in relazione con il ttl dei pacchetti sulla rete.

Alcuni ns non sono autorità per nessuna zona, e sono solamente a disposizione dei *client* per accettare *query* ricorsive ed effettuare *query* iterative ad altri ns. Non sono solo a disposizione dei *client*. Normalmente ogni ISP ne mette a disposizione uno per i propri clienti, sono chiamati a volte *Local* ns, o *Caching* ns, oppure *Recursive* ns.

Le informazioni DNS sono memorizzate su *Resource Record*, ed ogni dominio è associato ad uno o più *Resource Record*, questi contengono gli indirizzi IP, ma anche altre informazioni che permettono di associare ai nomi altri servizi. Uno di questi servizi sono i servizi di posta elettronica. Hanno la seguente struttura:

```
<resource record>
    <domain name><time to live>
    <class><type><value>
```

Questi *record* contengono campi per il nome del dominio, il ttl, la classe, in *internet* è sempre `IN`, il tipo di *record* ed il valore, che dipende dal tipo. I tipi principali sono

- `SOA`: *Start of Authority*, contiene informazioni amministrative della zona;
- `A`: Contiene nel valore l'indirizzo IPv4 dell'*host*;

- `MX`: Specifica il nome dell'*host* che accetta le *mail* indirizzate al dominio del *record*;
- `NS`: Il ns per una zona;
- `AAAA`: Contiene nel valore l'indirizzo IPv6 dell'*host*;
- `CNAME`: *Canonical Name*, il nome del dominio specificato corrisponde ad un altro nome.

Si possono assegnare dei nomi alle macchine che si comportano come ns. Le radici contengono *record* che indicano i ns assegnati alle varie zone, e per permettere la risoluzione contengono anche un *record* di tipo `A` (o `AAAA`) con il loro indirizzo. Essendo *record* contenenti indirizzi di una zona dovrebbero trovarsi nel ns relativo a quella zona, quindi questa può essere considerata un'invasione di gerarchia da parte della radice. Questi *record* prendono il nome di *Glue Record*.

I messaggi DNS si realizzano in due tipi di stesso formato, richiesta e risposta. Tra i campi dell'*header* si hanno:

- `QR`: *Query*, indica se il messaggio è una domanda o una risposta;
- `RD`: *Recursion Required*, indica che la query è di tipo ricorsivo.

Tra i campi della *question section* si hanno:

- `NAME`: Nome della risorsa;
- `TYPE`: Tipo della risorsa.

Questi messaggi devono viaggiare su un protocollo di livello quattro affidabile, quindi sembra bisogna essere assegnato al protocollo TCP, ma in realtà viene assegnato ad UDP. È la logica applicativa del *resolver* che si prende carico del meccanismo dei riconto, infatti in caso le richieste non ricevono risposta, questo invia nuovamente la richiesta dopo un certo intervallo di tempo. I *client* si rivolgono alla porta UDP `53` *Well Known*, la richiesta e la risposta viaggiano entrambe su singoli pacchetti UDP. Per dialoghi che richiedono risposte di grande dimensioni si può usare TCP, nel caso in cui *server* secondario si rivolge al primario per richiedere le informazioni aggiornate.

Sotto il DNS non ci sono solo messaggi UDP, ma ci sono protocolli di sicurezza per evitare vengano letti da terze parti, la risposta viene quindi considerata autentica. Questi meccanismi non sono trattati in questo corso.

Il comando `dig` permette di effettuare il processo di risoluzione comportandosi come un *client* ed osservare a schermo le risposte ottenute. Prende come argomento il nome o indirizzo IP del ns a cui si possono aggiungere parametri opzionali come il nome della risorsa che si vuole accedere, il tipo di *query* che si vuole effettuare, la porta a cui si vuole effettuare questa richiesta, etc. Su Windows si può usare il comando analogo `nslookup`.

Il comando restituisce la lista dei *record* ottenuti come risposta ed il loro tipo:

```
\prompt> dig google.com
; <<>> DiG 9.18.28-0ubuntu0.24.04.1-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
```

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40999
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             103     IN      A       142.251.209.14

;; Query time: 10 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Wed Jan 15 12:36:21 UTC 2025
;; MSG SIZE  rcvd: 54
```

I nomi vengono rappresentati con un punto alla fine per convenzione per indicare che sono sottoalberi di una radice comune avente stringa vuota. Il *server* di risposta coincide all'interfaccia di lo, dato che questo *record* era già presente nella *cache* del *resolver*. Se viene eseguito il comando dopo un certo periodo di tempo il ttl restituito è minore, dato che decrementa al passare del tempo per diminuire la validità nella *cache*. Se viene effettuata questa ricerca su una machina locale, viene restituito un *record* contenente il suo *alias*, seguito dai *record* che utilizzano questo nome per identificare l'indirizzo. Si utilizzano questi *alias* poiché il nome del servizio deve esistere per fornire informazioni agli utenti, viene realizzato a scopo di *marketing*, nella fase di risoluzione infatti viene connesso al nome effettivo di quel servizio.

Per effettuare una *query* non ricorsiva si utilizza la *flag* `+norecurse`, restituisce molti nomi dei ns e relativi indirizzi. Inoltre con `+trace` è possibile seguire il cammino delle deleghe che viene effettuato per individuare l'indirizzo, fornendo i *record* analizzati da ogni ns nella sequenza per individuare l'indirizzo. I primi ns in questa lista rappresentano le radici, `root-servers`, di varie zone, identificati da una lettera come prefisso per indicare il gestore. I `root-servers` europei hanno tradizionalmente il prefisso `k`.

6.2 Il Linguaggio HTML

Il linguaggio HTML è il linguaggio che specifica le pagine web, si interessa il protocollo che porta queste informazioni attraverso la rete, ovvero HTTP. HTML, *HyperText Markup Language*, è un testo che può contenere riferimenti ad altri ipertesti. Un marcatore è un codice che segnala l'inizio o la fine di una primitiva di formattazione del testo.

Le caratteristiche dell'ipertesto sono la concisione, dato che il contenuto di una singola pagina può essere ristretto all'essenziale; la completezza, l'informazione distribuita sulle varie pagine può essere arbitrariamente dettagliata; la condivisione, i dati non vengono replicati ma condivisi, evitando quindi le ridondanze. Si può arrivare a livelli di completezza arbitrari affidando a subordinate distribuendo l'informazione nelle pagine. L'ipertesto costituito dall'insieme di tutte le pagine presenti su *internet* viene denominato *World Wide Web*, rete estesa al mondo intero. Bisogna distinguere tra *internet* e WWW. La parola *internet* è ulteriormente ambigua poiché il nome del protocollo di livello quattro *Internet Protocol* e l'insieme dei protocolli di rete *Internet Protocol*

Suite condividono lo stesso nome. Inoltre si riferisce alla struttura fisica, ma spesso si riferisce a tutti i servizi disponibili in questo ambiente distribuito. Mentre WWW si riferisce alla rete logica costituita dalle pagine *web* e dai loro *hyperlink*, queste costituiscono solo uno dei servizi offerti da *internet*.

HTML è un linguaggio di *markup*, è un linguaggio che codifica delle informazioni tramite marcatori. Un linguaggio di marcatura ha senso quando si vuole rappresentare un'informazione che può essere strutturata in termini sequenziali.

Si possono identificare linguaggi di marcatura fisica o logica. I linguaggi di marcatura fisica descrivono dettagliatamente come apparirà il testo formattato. Queste hanno una filosofia WYSIWYG, *What You See Is What You Get*. Si vuole poter descrivere testi per schermi di dimensione e risoluzioni differenti, quindi specificare direttamente la dimensione degli oggetti di una pagina è sconsigliato. Si utilizzano quindi linguaggi di marcatura logica o semantica descrive il significato strutturale degli oggetti da formattare con una filosofia WYGIWYM, *What You Get Is What You Mean*.

HTML usa una marcatura logica e permette di realizzare pagine *web* senza doversi preoccupare delle caratteristiche del dispositivo sul quale quel testo sarà offerto in consultazione. Ogni marcatore in HTML viene identificato all'interno di parentesi acute `< ... >`. I marcatori di fine formattazione hanno uno *slash* prima della parentesi angolata aperta `</ ... >`. Alcuni marcatori ammettono valori, specificati dopo un uguale: `<marcatore attributo1 = valore1 ... attributoN = valoreN>`. I commenti si scrivono nel seguente formato `<!-- ... --->`. Le parentesi angolate non sono quindi disponibili come caratteri si utilizzano caratteri speciali per indicare questi caratteri che non possono essere scritti direttamente. Iniziano con `&` o `;`:

- `&lt`: <;
- `&gt`: >;
- `&amp`: &;
- `&_grave`: lettera _ con accento grave.

Esistono marcatori di metadati che non necessariamente servono per la formattazione, ma possono inserire marcatori per identificare certe parole all'interno della pagina: `<meta ...>`.

Un *file* HTML viene strutturato all'interno del marcatore `<html> ... </html>` e si può dividere in due sezioni `<head> ... </head>` che contiene informazioni relative all'intera pagina, e `<body> ... </body>` che contiene descrizioni relative agli oggetti da visualizzare nella pagina. Nell'intestazione si può trovare il titolo con `<title> ... </title>`, metadati `<meta ... = ... >`

Si può formattare il testo con i seguenti marcatori:

- `<center> ... </center>`: Centratura;
- `<h1> ... </h1>`: Titolo;
- `<h2> ... </h2>`: Sottotitolo;
- `<b> ... </b>`: **Bold**;

- `<i> ... </i>`: *Italic*;
- `<u> ... </u>`: Sottolineato;
- `<blink> ... </blink>`: Lampeggiante.

Si definiscono ancore con il marcatore `<a href="..."> ... </a>`, in base all'attributo si possono creare *hyperlink* specificando un URL, oppure un punto di arrivo o un indirizzo ad un punto di arrivo a questa o ad altre pagine.

Si possono inserire immagini con `<img src="\textcolor{SpringGreen}{URL}">`, a capi di un paragrafo `<p>` o *break* `<br>`, linee orizzontali `<hr>` di cui si può modificare lo spessore con `size=...`. Si può cambiare il colore del font con `<font color=...></font>`. Si possono realizzare liste non ordinate:

```
<ul>
    <li> ...
    <li> ...
</ul>
```

Oppure ordinate:

```
<ol>
    <li> ...
    <li> ...
</ol>
```

Si possono definire tabelle con:

```
<table border=...>
    <th> ... </th> <!-- titolo cella-->
    <tr>  <!-- inizio-fine linea-->
        <td> ... </td> <!-- inizio-fine cella-->
        ...
        <td> ... </td> <!-- inizio-fine cella-->
    </tr>
</table>
```

Si possono definire ulteriori marcatori ed estensioni, ma se il *browser* non conosce il marcatore non produce errore, ma semplicemente lo ignora. HTML è un linguaggio estremamente basilare che si è evoluto nel tempo, attualmente siamo alla versione HTML 5. Il codice HTML viene scandito riga per riga, partendo dall'alto ed eventualmente viene formattata la pagina. HTML viene spesso accompagnato da altre tecnologie come javascript o CSS, *Cascading Style Sheets*, fogli di stile, elementi di formattazione che possono essere utilizzati per rendere la pagina al meglio.

6.3 URL ed il Protocollo HTTP

Prima di descrivere il protocollo HTTP, bisogna descrivere l'URL, *Uniform Resource Locator*, o l'URI, *Uniform Resource Identifier*. Poiché in *internet* non si differenzia tra la risorsa e la sua posizione.

Un URL è una rappresentazione testuale e compatta di una risorsa disponibile, in modo sintetico realizzata da uno schema, ed una parte dipendente dallo schema, separati da `:`. Lo schema descrive il tipo di risorsa acceduta, e la parte seguente la individua secondo le caratteristiche di un dato schema. Tipicamente lo schema è `http` o `https`, questi sono sia risorse che protocolli per acquisire risorse. Schemi come `\textit{file}` sono risorse presenti sul proprio calcolatore.

Alcuni schemi utilizzano la stessa sintassi per la parte dipendente, questa sintassi si chiama *Common Internet Scheme Syntax*, CISS. Si utilizza un doppio *slash* iniziale seguito da un campo con l'*userid* e la *password* seguita da una chiocciola, ma questo è molto raramente utilizzato negli URL moderni. Il secondo campo dopo la chiocciola corrisponde all'indirizzo dell'*host*, il suo numero IP, o ad una stringa identificativa di questo indirizzo sulla rete. Il campo successivo specificato dopo `:` indica la porta dove è disponibile quella risorsa, si suppone la risorsa sia un livello quattro che dispone di porte, questa può essere omessa, ed in caso si utilizza una porta di quelle note, associate a protocolli HTTP, come la porta `80`. L'ultimo campo individua un *file* all'interno del *filesystem*, del *server* con `/` seguito dall'indirizzo del *file*. Alcuni sistemi presentano variazioni nella sintassi di questo campo, come HTTP e HTTPS. Questo identifica una risorsa, tipicamente attraverso l'indirizzo dell'*host*, ma l'indirizzo IP ha anche il ruolo di locazione, quindi è possibile localizzare la risorsa dato l'IP.

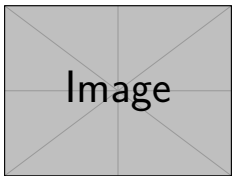

Fig. 47: Struttura Schema di HTTP

Quando un browser deve accedere ad una risorsa individuata da un URL etra i campi di indirizzi e la porta, se non è specificata utilizza la porta `80`, per il protocollo HTTP. Questo lavoro lo effettua qualsiasi applicazione che tenta di accedere ad una risorsa. Il campo *userid* e *password* vengono utilizzati per gestire un colloquio con il *server*, senza coinvolgere l'utente. In coda al percorso della risorsa può esserci una stringa preceduta dal carattere `#` per individuare la posizione all'interno della pagina HTML, utilizzata dal *client* e non inviata al *server*. Oppure una stringa preceduta da `?` viene inviata al *server*, per specificare parametri o altre opzioni.

Il protocollo HTTP, *HyperText Transfer Protocol*, è stato progettato come un protocollo di livello applicativo per realizzare sistemi distribuiti, basati su ipertesti, anche se da molto non è più esclusivo per HTML. Questo protocollo si basa su uno schema richiesta-risposta tra il *client* ed il *server*; uno di questi colloqui tra *client* e *server* si chiama sessione.

HTTP venne inventato da Tim Bernes-Lee, un ricercatore del CERN nel 1989, e da HTTP/3, nel progetto QUIC, c'è una variazione molto grande rispetto alle reti precedenti, con lo standard RFC 9114.

Una sessione di HTTP/1.0 è composta da quattro fasi: apertura, richiesta, risposta e chiusura. Nel codice del *server* è stato specificato ad un processo di ascoltare sulla porta `80`, oppure una porta specificata per ascoltare eventuali richieste di connessione. Un cliente quindi può effettuare una connessione TCP con questa porta TCP del *server*. Una volta instaurata una connessione TCP viene inviata una richiesta al *server*, un pacchetto HTTP di richiesta al *server* con la specifica della risorsa alla quale si vuole accedere. Il *server* poi restituisce la risorsa, tipicamente un *file*, contenente una descrizione HTML di una pagina che si vuole visualizzare. Dopo aver terminato la trasmissione, il *server* chiude la connessione TCP. Avviene quindi un singolo scambio *client-server* in ogni sessione, questa sezione è completamente *stateless*.

Nella richiesta viene specificato uno di possibili metodi tra questi seguenti, per specificare quale azione da eseguire alla risorsa:

- `GET`: Richiede la risorsa indicata;
- `POST`: Invia dati alla risorsa indicata;
- `HEAD`: Chiede informazioni sulla risorsa indicata;
- `PUT`: Copia dei dati inviati sulla risorsa con una sostituzione di *file*;
- `DELETE`: Richiede la cancellazione della risorsa:
- `OPTIONS`: RIchiede di conoscere le opzioni disponibili per il trasferimento della risorsa specificata.

Un pacchetto HTTP di richiesta è strutturato da un'intestazione composta da uno di questi metodi, la risorsa e la versione di protocollo da adottare seguita da informazioni aggiuntive, *request fields*, sul collegamento instaurato. Una riga vuota quindi divide l'intestazione dal corpo del pacchetto dove sono presenti i dati effettivamente inviati dal *client* al *server*.

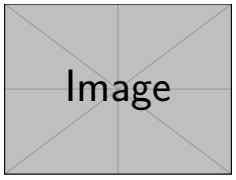

Fig. 48: Struttura HTTP pdu di Richiesta

I pacchetti di risposta hanno nell'intestazione la versione, il codice di stato ed in linguaggio naturale la spiegazione per il codice. Questi codici di stato possono essere positivi, di tipo `2xx` o `3xx` oppure negativi `4xx` o `5xx`. Non è necessario conoscere questi codici di risposta poiché nel pacchetto è presente loro spiegazione. I codici di risposta sono tipicamente sempre leggibili, per

molti protocolli che definiti da standard RFC. Nell'intestazione sono presenti anche i *response fields* su trasferimento dal *server* al *client*, seguiti da una riga vuota per separare l'intestazione dal corpo del pacchetto, contenente i dati inviati.

I codici di stato più comuni sono, riguardo a richieste soddisfatte del *client*:

- `200`: Il pacchetto contiene ciò che è stato richiesto;
- `201`: È stat creata una nuova risorsa come richiesto;
- `202`: È stata accettata la richiesta, ma non è stata creata la nuova risorsa.

Possono essere riguardo ulteriori azioni richieste per completare l'operazione:

- `301`: La risorsa è disponibile ad un altro indirizzo;
- `304`: La risorsa non è stata modificata.

Possono specificare che la richiesta errata e non può essere soddisfatta:

- `400`: La sintassi della richiesta è errata;
- `401`: È richiesta autorizzazione;
- `403`: La risorsa non è disponibile;
- `404`: La risorsa non esiste;
- `405`: Il metodo specificato non è permesso per la risorsa indicata;
- `414`: Il nome della risorsa è troppo lungo.

Oppure possono indicare che non può essere soddisfatta una richiesta legittima:

- `500`: È stato sollevato un errore interno al *server*;
- `501`: La funzionalità richiesta non è implementata;
- `505`: La versione del protocollo specificata non è supportata.

Nelle versioni successive i pacchetti vengono inviati allo stesso modo, ciò che ca,bia sono le informazioni addizionali. Possono essere presenti alcuni campi di richiesta:

- `accept-charset`: Insieme di caratteri accettabili;
- `cookie`: Un *cookie* inviato precedentemente dal *server*, usando `set-cookie`;
- `content-length`: Lunghezza in byte del corpo;
- `host`: Il *domain name* del *server* e il numero di *port* TCP, utile per servire vari domini sullo stesso *server*. Su uno stesso *server* possono essere presenti servizi diversi, si utilizza il campo `host` per specificare a quale di questi si vole accedere;

- `if-modified-since`: Permette al *server* di inviare una risposta `304` *not modified* se la risorsa è invariata dopo la data specificata;
- `if-none-match`: Permette al *server* di inviare una risposta `304` *not modified* se la risorsa è invariata.

Alcuni campi di risposta sono:

- `ETag`: Un identificatore assegnato da un *server* ad una specifica versione della risorsa, può essere ottenuto con una funzione di *hash*;
- `set-cookie`: Un *cookie* HTTP.

Il *server* può offrire informazioni aggiuntive rispetto ad un semplice valore temporale. Un *cookie* è un dato spedito dal *server* e memorizzato nel *client*, viene re-inviato dal *client* al *server* ogni volta che accede a certe risorse, definite dallo scope del *cookie*. Viene usato per gestire uno stato nelle sessioni HTTP. Il *server* può quindi riconoscere attività precedenti dello stesso *client* in base ai *cookie*. L'applicazione più evidente è quella dell'autenticazione degli accessi ad un *server*, questo *cookie* permettere al *server* di identificare lo stesso *client*, quindi attribuisce i diritti per poter eseguire le operazioni successive. Può anche permettere di tracciare le pagine visitate da uno stesso utente. Un *cookie* può essere utilizzato anche da altri servizi nonostante non sia stato rilasciato per un certo sito. Nell'EU all'accesso di un sito, questo è obbligato di specificare il motivo dei *cookie* che vengono utilizzati, il loro scopo o finalità, ed il modo in cui vengono salvati e mantenuti nel tempo. Inoltre è possibile accedere comunque ad un sito senza accettare all'utilizzo dei *cookie*.

Il protocollo HTTP 1.1 permette di effettuare diverse sessioni sulla stessa connessione TCP, ma questo dipende comunque dal *server*, diminuendo la latenza.

Questa possibilità è particolarmente importante se insieme a HTTP e TCP si utilizza un protocollo per rendere sicure le connessioni. Attualmente si utilizza il protocollo TLS, *Transport Layer Security*, dopo l'instaurazione della connessione TCP i due *host* si scambiano informazioni per mantenere la sicurezza delle connessioni con un altro *three-way-handshake*. Quindi all'instaurazione della connessione TCP si effettua uno scambio per determinare il protocollo di sicurezza utilizzato. Questo permette di offuscare i pacchetti trasmessi tra queste due macchine mantenendo la connessione persistente. Se fosse possibile effettuare una singola connessione TCP per ogni richiesta verrebbero scambiati 7 pacchetti di apertura-chiusura per una singola sessione, 10 considerando anche i protocolli di sicurezza. Quindi in versioni precedenti alla 1.1 l'*overhead* per l'invio di un singolo pacchetto è molto rilevante.

HTTP quindi rappresenta una sequenza di transizioni richiesta-risposta tra un *client* ed un *server*, tipicamente originate da un singolo utente, per accedere a risorse tra loro correlate.

Se si usano HTTP e TLS separatamente, è opzionale l'utilizzo di TLS, mentre il protocollo HTTPS presenta l'utilizzo obbligatorio del protocollo di sicurezza TLS. Cambia anche la porta di ascolto nota e diventa la porta `443`.

Nonostante la possibilità di inviare più richieste su una stessa connessione, queste vengono effettuate sequenzialmente, e quindi una nuova richiesta può essere inviata solamente dopo aver ricevuto la risposta della precedente, indipendentemente dal tempo di attesa. Per cui dalla seconda

versione del protocollo si inseriscono meccanismi di parallelismo, inviando varie richieste sulla stessa connessione, in un processo chiamato HTTP *pipelining*.

HTTP/3 invece utilizza un ulteriore protocollo di trasporto chiamato QUIC e non TCP. QUIC utilizza pacchetti UDP, basato su implementazioni nell'*user-space* e quindi non sono legate all'aggiornamento del sistema operativo. Si intreccia quindi l'evoluzione dei protocolli di trasporto e l'evoluzione di HTTP. Questo rappresenta un fenomeno di evoluzione molto recente.

In QUIC il *three-way-handshake* è mescolato con l'*handshake* del protocollo TLS 1.3, con un minore *overhead* della connessione. Per utilizzare questo protocollo, anche il *server* deve poterlo utilizzare, sicuramente supportato da *server* appartenenti a Google, avendo creato questo protocollo. Anche se il *server* non appartiene a Google, allora è possibile che utilizzi il protocollo QUIC, circa il 20% dei siti *web* infatti utilizza il protocollo HTTP/3, mentre tutti i *browser* lo utilizzano, perché vogliono offrire sempre migliori prestazioni ai loro utenti.

7 Posta Elettronica

La descrizione di questo servizio *web* offre la definizione di un criterio per la progettazione o modifica di servizi. La progettazione è divisa in quattro fasi, un'analisi dei requisiti, definendo le caratteristiche dei servizi. La specifica delle primitive del servizio, che rappresenta la sua interfaccia. La definizione dell'architettura e delle operazioni, definizione delle applicazioni coinvolte nella sua gestione, eventuali ambienti di supporto e le operazioni dei flussi di informazione. Laa definizione dei protocolli per ogni tipo di comunicazione, bisogna definire la struttura delle pdu, degli stati e delle procedure del protocollo. Non verrà trattato il dimensionamento o la dislocazione del servizio sulla rete.

I servizi di posta elettronica possono essere forniti da ISP, aziende come Google, che offrono molti servizi *web*.

Un servizio di posta deve poter gestire messaggi in partenza ed in arrivo, spedire e ricevere messaggi ad uno o più destinatari. Richiede di preservare la riservatezza dei dati, e si vuole avere una certezza sulla consegna, è anche possibile avere una consegna differita nel tempo, per essere garantita. Quest'approccio duale rispetto ad altri servizi *web*. Si vuole che la configurazione sia poco onerosa da parte degli utenti finali con un'interfaccia utente elementare ed intuitiva.

Definiti i requisiti si definiscono delle primitive di servizio offerte agli utenti, per i messaggi in partenza:

- Composizione di un messaggio;
- Memorizzazione di un messaggio;
- Cancellazione di un messaggio;
- Caricamento di un messaggio.

Per i messaggi ricevuti:

- Lettura dei messaggi;
- Memorizzazione di un messaggio;
- Cancellazione del messaggio;
- Stampa di un messaggio.

Queste sono le primitive di base per poter almeno gestire la posta elettronica. Dopo queste due fasi bisogna definire l'architettura del servizio.

Si considera l'architettura odierna della posta elettronica, descrivendo ulteriori architetture non realizzabili valutando variazioni. Si considera una possibile architettura realizzata da una connessione diretta tra il mittente ed il destinatario. Sono quindi presenti due processi su questi due *host*, attivi, che devono poter ascoltare e inviare dati su una connessione, potrebbero utilizzare TCP. Per definire la porta di ascolto si potrebbe definire uno standard globale, su una porta ben nota alla definizione dell'architettura. Questo approccio è molto semplice, il messaggio viene recapitato in tempo reale e si ha anche la certezza della consegna, basata sull'affidabilità di TCP. Ma il mittente

non può spedire il messaggio se il processo sul destinatario non è attivo, quindi non rappresenta un'implementazione realizzabile. Inoltre il mittente si deve ricordare il nome della macchina del destinatario, può essere il nome del DNS o dell'indirizzo IP, ma questo deve essere noto per poter instaurare la connessione. Questo stesso approccio non è ragionevole per molti servizi basati sulle reti. Inoltre non viene definito come il destinatario viene autenticato, poiché viene riconosciuto solamente dalla macchina.

Si considera quindi una variante dove un *server* intermedio disaccoppia le due macchine *host*, questo *server* rappresenta il dominio del destinatario, ovvero un nodo dell'albero del *namespace*. Si ipotizza di legare i destinatari a domini di *internet*, sotto qualche forma. Questo *server* di ricezione deve essere ben noto. I processi di invio e ricezione sono disaccoppiati, ed il destinatario può essere autenticato dal *server*, dove sono presenti servizi di autenticazione. Il mittente non deve ricordare la specifica macchina del destinatario, ma solamente il dominio. Ma questo *server* può essere impegnato, guasto o sovraccarico in un certo istante e questo non garantisce di ricevere il messaggio dal *server*. Dopo aver inviato il messaggio infatti il processo del mittente può essere chiuso e quindi il messaggio non è in grado di arrivare al *server*. Poiché non si ha autorità sulla gestione del *server*.

L'architettura moderna contiene un *server* mittente per memorizzare il messaggio, con funzione di inoltro, e molti *server* duplicati, definendone uno primario ed altri ausiliari, in ricezione. In questo modo se il primo *server* non è raggiungibile si può tentare di raggiungere uno dei *server* secondari, per inoltrare il messaggio al destinatario.

Questa rappresenta un'architettura diversa da quella che si potrebbe aspettare. Bisogna quindi specificare come si definiscono indirizzi di posta elettronica, in rifermento a questa architettura. Una prima parte dell'indirizzo identifica un'entità che riceve il messaggio, ed una seconda parte definisce il dominio dell'utente, dopo una chiocciola `@`, per ricavare il *server* che deve ricevere il messaggio.

La prima applicazione coinvolta nel servizio si chiama MUA, *Mail User Agent* e viene mandata in esecuzione quando si vuole accedere al servizio di posta elettronica. L'utente può chiudere questa applicazione quando lo ritiene opportuno, viene chiamata anche *mailer*. L'applicazione MTA, *Mail Transmission Agent*, al contrario della MUA, è accessibile in modo stabile nel tempo, poiché si trova su di un *server* di inoltro per permettere al messaggio di essere trasmesso da una sorgente ad una destinazione.

Esempi di MUA sono applicazioni Desktop come Mozilla Thunderbird o Microsoft Outlook, altri servizi si basano sul *web* che trasformano il browser in un MUA, come Gmail, a cui bisogna però autenticarsi sul *browser*. Sono inoltre disponibili molte applicazioni mobile Android ed iOS.

MUA e MTA sono applicazioni, che vengono eseguite su delle macchine, quelle che ospitano gli MTA sono dei *server* che possono avere diversi ruoli nell'architettura appena definita. Un primo tipo di *server* si chiama OMS, *Outgoing Main Server*, alla quale MTA si riferisce direttamente il processo MUA del mittente. Un *server* di tipo *Mail eXchanger*, per ogni dominio DNS infatti esiste una lista di *host*, in ordine di priorità che ospitano gli MTA incaricati di ricevere posta per quel dominio. Un ulteriore *server* chiamato *Incoming Mail Server* contiene la MTA che comunica direttamente con la MUA del destinatario, generalmente coincide con il Mail eXchanger primario del dominio.

Si vuole che l'utente possa configurare facilmente il servizio della posta elettronica. In quest'architettura l'utente deve ricordarsi semplicemente due *server*, il suo MUA si riferisce infatti ad un OMS ed un IMS. Questi *server* invece di specificarli con un indirizzo IP si possono specificare con un nome, in modo che se dovesse cambiare l'indirizzo IP della macchina non sarebbe necessario modificare il servizio.

Le MTA ospitate sull'OMS sono suddivise in due processi chiamate MSA, *Message Submission Agent*, quello a cui si rivolge la MUA per spedire la posta, ed il secondo è il vero e proprio MTA che cura la spedizione al MTA del dominio del destinatario. Lo stesso avviene per un IMS, dove esiste sempre un processo MTA, ed un ulteriore processo MDA, *Mail Delivery Agent*, responsabile del recapito all'utente.

Definita l'architettura si definiscono una serie di azioni, a più livelli di dettaglio, scomposte in varie operazioni. Tutte le primitive di gestione dei messaggi in partenza e dei messaggi ricevuti sono delegati al MUA. Quando l'utente richiede il salvataggio della MUA, quando l'utente richiede il salvataggio del messaggio, il MUA accede al *filesystem* locale, per salvare la nuova *mail*, in caso non esista, accodandola al *file* delle *mail* salvate, con il rispettivo testo.

Quando si spedisce un messaggio il MUA invia il messaggio al suo OMS, che cerca il MX del dominio del destinatario utilizzando il DNS, se il primario non è disponibile si rivolge al secondario, richiedendo al proprio *server* primario la lista dei MX del dominio destinazione, in ordine di priorità, tentando di trasmettere il messaggio ad intervalli regolari. Se fallisce per tre giorni consecutivi viene notificato l'utente del fallimento. Se viene inviato ad un MX secondario, questo tenta ad intervalli regolari di inviare i messaggi salvati al MX primario.

Sono presenti anche servizi ausiliari, ovvero il DNS, senza il quale la posta elettronica non funzionerebbe, inoltre è necessario un *filesystem* distribuito. I *server* MX infatti devono poter accedere alle *mail* tramite un sistema distribuito, poiché potrebbero non trovarsi sullo stesso *server*.

Esiste un comando chiamato `dig` che permette, sulle macchine Linux, di comportarsi come *resolver*, nel mondo Windows è presente un comando analogo chiamato `nslookup`, che fornisce *record* MX. In questo modo è possibile ottenere le informazioni relative ai *server* ed i loro indirizzi IPv4. Utilizzando questo comando bisogna specificare ulteriori parametri per leggere i *record* MX, e specificare l'indirizzo. Questo va effettuato anche con il comando `dig` specificando dopo la *flag* `-f` il tipo di *record* che si vuole leggere e l'indirizzo corrispondente.

```
\prompt> dig -f MX gmail.com
```

Dato un MUA configurato per spedire *mail* bisogna conoscere l'indirizzo IP dell'OMS, questo si ottiene specificando un nome ed invocando un DNS, questo effettua la risoluzione dell'indirizzo, e rappresenta il primo punto di contatto tra la posta elettronica ed il DNS. Un punto di contatto diverso rappresenta l'OMS che notifica al DNS di voler conoscere l'indirizzo MX del destinatario. Questa seconda operazione infatti non rappresenta una risoluzione canonica. Il messaggio viene inviato dall'OMS al MX relativo all'IMS del destinatario. Il MX e l'IMS comunicano tra di loro tramite una memoria di massa condivisa sulla quale può scrivere il MX e leggere l'IMS.

Si possono distinguere due tipi di flussi di informazione, tra il primo MUA e l'MTA dell'OMS, che si trasmettono messaggi fino a quando non arrivano al MX primario del destinatario. Questo primo flusso di informazione è realizzato da un *client* che intende trasferire uno o più messaggi, tramite un servizio che il *server* potrebbe accettare o rifiutare di concedere. Non occorre autenticare gli

interlocutori poiché o entrambi hanno i privilegi di amministratore, oppure uno di essi è il mittente, e non si considera cruciale assicurarsi che il mittente corrisponda a quanto dichiarato. Poiché l'IMS ed il MX sono due macchine associate a compagnie diverse e non sarebbe realizzabile permettere a tutti i possibili OMS le autenticazioni del *server* MX, analogamente per il riferimento ad utenti che inviano messaggi da spedire ai proprio OMS. Si suppone non ci sia una verifica di autenticazione in questo passaggio, e questo è la causa dello *spam* che invade la posta elettronica.

Mentre l'ultimo passaggio dall'IMS dal destinatario probabilmente è realizzato da un'interazione di richieste e risposte, e sembra essere intrinsecamente più complicato. Utilizza quindi un protocollo più articolato. Il *client* si informa sul numero e la dimensione dei messaggi contenuti nel *server*, e destinati allo specifico utente. In questo passaggio l'autenticazione è indispensabile, e si possono trasferire o non trasferire messaggi. Chi utilizza questi messaggi si deve autenticare poiché consente di leggere solo i messaggi espressamente indirizzati all'utente con le opportune credenziali. In questo passaggio bisogna permettere all'utente di cancellare messaggi senza trasferirli o di leggerli senza cancellarli.

Poiché questi due flussi di informazione sono così diversi, si utilizzano due sistemi di comunicazione diversi per ognuno di essi. Dato che la quantità di informazioni da scambiare non è determinabile a priori, si utilizza un protocollo connesso TCP per instaurare le connessioni. Bisogna definire due protocolli, definendo le operazioni, le procedure, i pacchetti e gli stati attraversati. Questi protocollo utilizzano un meccanismo *client-server*, dove il *client* chiede al *server* un servizio. Bisogna analizzare la differenza tra queste richieste e le risposte. In HTTP la richiesta si effettua specificando un metodo, ed il *server* invia al *client* una risorsa in base al metodo specificato. Anche in questi protocolli si utilizzano dei metodi, e bisogna definire i loro metodi d'uso, e le relative risposte dal *server*. Il primo tipo di flusso di informazione è supportato da un protocollo chiamato smtp *Simple Mail Transfer Protocol*, molto vecchio e modificato molte volte. Questo protocollo usa una porta TCP *well-known* `25`. Ed è interamente *text based*, quindi di facile comprensione. Ci sono varie RFC che definisco il corpo di un messaggio di posta elettronica, tra cui 821, 5321 e 5322. Tipicamente la struttura del protocollo è rimasta invariate, solamente la struttura del corpo del messaggio è stata modificata. Per iniziare una conversazione in smtp si utilizza il comando `HELO`, seguito da un dominio. Il comando `MAIL FROM` inizia il trasferimento di una *mail* specificando uno o più destinatari. `RCPT TO` individua un destinatario, se sono presenti più destinatari, e non sono presenti nel dominio locale, viene richiesto all'applicazione ricevente di inoltrarli al MX del dominio, tramite un'operazione di *relaying*. Il comando `DATA` indica l'inizio del corpo della *mail*, e verrà trattato tale fino all'arrivo della combinazione `<CRLF>. <CRLF>`. Un punto ad inizio di riga ed un accapo, poiché tutte le righe del messaggio vengono processate, ed ai punti di inizio riga vengono sostituiti dei doppi punti, il processo inverso viene effettuato anche in ricezione per visualizzare correttamente il messaggio. Il comando `RSET` interrompe una connessione, terminando il trasferimento corrente. Per autenticarsi si utilizzano i comandi `VRFY` che richiede al ricevente di confermare che il destinatario esiste. `EXPN` richiede al ricevente di confermare che l*alias* di posta specificato corrisponde ad una lista di utenti, eventualmente ritornata al ricevente. Il comando `HELP` restituisce tutti i comandi disponibili, se viene seguito da un comando, fornisce informazioni specifiche per quel comando. Il comando `NOOP` non effettua alcuna operazione, ma richiede una risposta affermativa dalla controparte. Per terminare una comunicazione si utilizza il comando `QUIT`.

Le risposte del *server* sono dei codici numerici, seguiti da una loro spiegazione in linguaggio

naturale, analogamente per HTTP.

Una transazione viene iniziata quando il *client* inserisce il comando `MAIL`, che deve essere seguito da `RCPT`, eventualmente iterato, e poi dal comando `DATA` per specificare il corpo del messaggio. A questo punto il *server* entra in uno stato di accettazione del messaggio che termina quando riconosce la sequenza `<CRLF>. <CRLF>`. Durante il trasferimento non vengono accettati comandi, dopo un trasferimento avvenuto con successo si può chiudere la transazione con il comando `QUIT`.

Per la transazione tra l'IMS e la MUA del destinatario si possono utilizzare molti protocolli, uno di questi è pop3, *Post Office Protocol 3*, utilizza la porta *well known* `10`, specificato nelle RFC 1725 e 1939. Ormai non è più utilizzato e si utilizza il protocollo imap, o variazioni sempre su di imap. I comandi di pop3 sono composti da quattro lettere e le risposte dal *server* ISM sono precedute da `+OK` se positive e da `-ERR` se negative. I comandi `USER` e `PASS` permettono di autenticare l'utente, specificando queste informazioni. Queste passano in chiaro sulla rete, se non è specificato un protocollo di cifratura. Alcuni comandi sono gli stessi come `NOOP`, `RSET` e `QUIT`. Mentre per effettuare le operazioni specifiche di questo flusso di informazione si utilizzano i comandi `LIST`, che fornisce una lista dei messaggi disponibili, ordinati e numerati, affiancati da un numero che rappresenta la dimensione del messaggio in byte. Il comando `STAT` mostra il numero di messaggi totali e la loro dimensione complessiva. I byte vengono chiamati ottetti, *octets*, gruppi di 8 bit. Per visualizzare un messaggio si utilizza il comando `RETR` seguito dal numero del messaggio che si vuole visualizzare. Per eliminare un messaggio si utilizza `DELE`, sempre seguito dal numero relativo al messaggio.

Il primo stato di pop3 è l'autenticazione, dopo essere autenticato entra in uno stato di transazione, dove è possibile effettuare richieste di servizio del *client* al *server*. Dopo aver inviato tutto entra in uno stato di *update* e vengono rilasciate le risorse acquisite, e la connessione TCP viene chiusa.