Infrastrutture e Protocolli – BETA VERSION

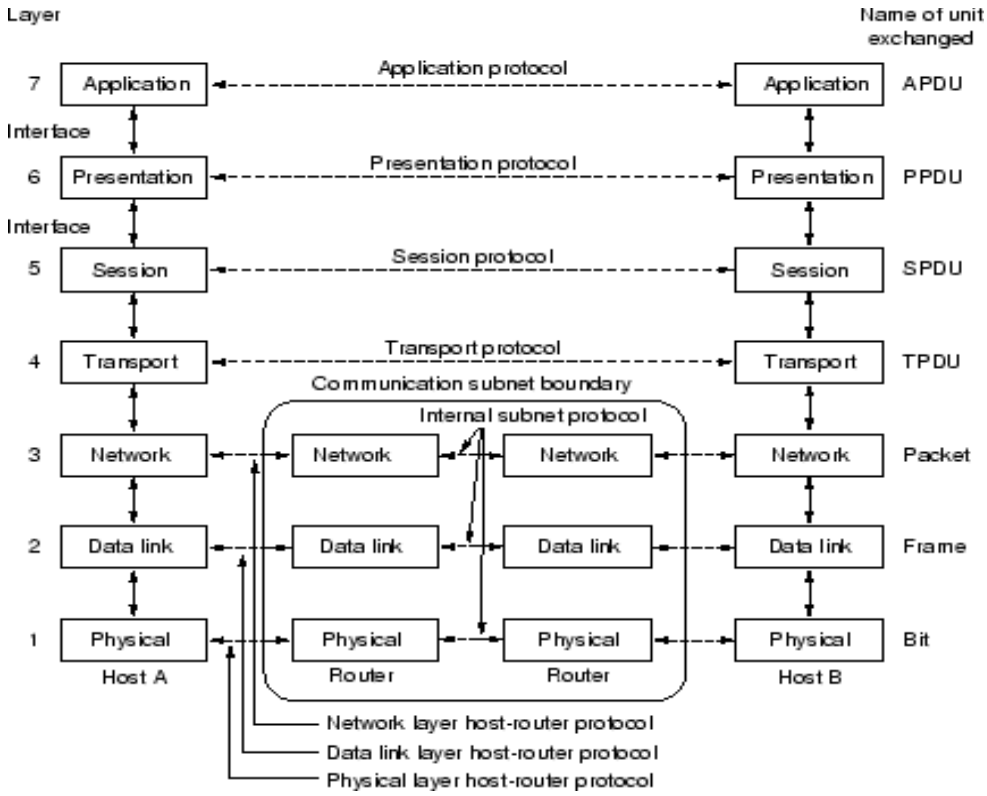
Il seguente documento è rilasciato sotto licenza Creative Commons

Attenzione: i seguenti appunti non sono nè completi nè coprono tutte le parti trattate dal corso. Sono sicuramente presenti anche numerosi errori ed omissioni. Vi prego di segnalarmi i problemi su http://www.vortexmind.net

1 Servizi di comunicazione

Modello funzionale: ho due o più **entità remote** che devono comunicare (effettuare un *colloquio* attraverso un *servizio di comunicazione*) tramite uno scambio di *messaggi*. Il sistema che effettua lo scambio di messaggi è detto fornitore del servizio di trasporto dell'informazione.

Si trasferiscono **unità informative** (bit, gruppi di bit, files ...). Questo servizio è descritto dalle chiamate con cui è realizzato, dette **primitive di servizio**. Esse servono a descrivere, a richiedere e a ricevere informazioni sul servizio. Sono caratterizzate da parametri (info da trasferire, destinatario, caratteristiche ...). Il servizio di comunicazione si basa su un canale bidirezionale.



Le **modalità di comunicazione** possono essere a connessione (più fasi: **instaurazione,trasferimento,chiusura**) oppure senza connessione (una fase, non ho il concetto di sessione, avviene in modo autonomo).

Le entità colloquianti possono anche offrire un servizio di comunicazione a entità tere, dette di

livello superiore (si modificano le funzionalità del livello esistente).

Il **protocollo** è l'insieme di regole che guida il colloquio tra entità dello stesso livello (formato messaggi, informazioni di servizio, algoritmi di trasferimento). Un protocollo utilizza unità di trasferimento dati dette **PDU** (Packet Data Units, dette anche trame). Le PDU possono contenere un header di controllo e l'informazione vera e propria ricevuta dai livelli superiori.

Architettura a strati: si articolano i servizi di comunicazione stratificandoli su più livelli via via più complessi e ricchi di funzioni (dal trasporto dei bit su mezzo fisico ai servizi applicativi complessi)

Ogni servizio offerto da uno strato è rappresentato da un **SAP** (Service Access Point).

Funzioni di rete: si dice che tale funzione è implementata su un livello quando è possibile il colloquio tra più di due entità dello stesso livello. Si introduce la funzione di **instradamento** (selezionare il SAP d'uscita), ovvero il problema di individuare il partner in un colloquio. Questo problema può essere risolto ad un livello inferiore se si introduce la tecnica di **indirizzamento** (identifico N-SAP di destinazione). Anche se viene attribuita al livello 3, in realtà la funzionalità di rete può essere implementata ad ogni livello tramite apposite funzioni.

Indirizzamento: ho un indirizzo (univoco sui livelli) identificativo del SAP da cui raggiungere l'entità. Tipologie *unicast*, *multicast*, *broadcast*. **Forwarding/Commutazione**: inoltro che un entità fornisce ad altre entità sullo stesso livello (effettuazione di un passaggio versos un SAP già scelto). Si utilizzano *tabelle di instradamento* generate da *protocolli di instradamento*.

IEE 802 definisce reti a due livelli: **physical layer** (MAC e fisici 802.3 ethernet 802.11 wireless) e **data link layer** (LLC). Ci sono anche meccanismi di interconnessione tra i vari livelli (802.1 bridge e switch, non è un layer ma è un meccanismo trasversale tra i livelli).

Ethernet 802.3 (CSMA/CD): trasmettere ad alta velocità su un mezzo condiviso, senza nodi d'interconnessione. Problema dell'accesso multiplo (necessità di arbitraggio dell'accesso, collision detection). Ethernet non si preoccupa di evitare le collisioni, se si verificano cercano di correggere il problema (nessuna garanzia deterministica). Provo a trasmettere, vedo se c'è qualcuno che usa il cavo, se c'è attendo e aspetto che finisca poi trasmetto. Mentre trasmetto ascolto sul canale per capire se qualcun altro sta trasmettendo. Se c'è la collisione mando un segnale di collisione in broadcast, poi attendo un tempo random di backoff per ritrasmettere. Attenzione ai ritardi di propagazione. Il meccanismo è stato esteso fino a reti a 10Gb/s, anche se sono stati introdotti dispositivi di interconnessione (switch) che partizionano le reti e riducono la possibilità di una collisione. (*** trama ethernet: PAD, serve a raggiungere la lunghezza minima per una trasmissione ottimale in caso di payload troppo corto)

1.1 Interconnessione di reti locali

1.1.1 Livello 2 Transparent Bridging

Utilizzo di bridge/switch livello 2 (nodi di interconnessione). Ricevono trame di livello 2 e sono in grado di indirizzarle su altre porte di uscita (implementano la funzionalità di rete al livello 2). Riceve dati attraverso il livello fisico, le compone in trame MAC ethernet: il MAC Relay, che è il "motore intelligente" del bridge la prende e decide se e come inoltrarla su un'altra porta. Operano in modalità **filtering** (sorgente e destinazione in LAN1) oppure **relay** (sorgente LAN1 e instradamento su un altra partizione della rete)

modalità **store&forward**: ricevo tutta la trama e poi decido se metterla in una coda d'uscita (non faccio le cose bit a bit perchè lavoro già con pacchetti e non con singoli bit)

Il bridge divide una rete locale in più domini di collisione: collidono solo trame trasmesse contemporaneamente sulla stessa partizione della rete (LAN1, LAN2 ...). Tuttavia non partizionano

le trasmissioni in modalità broadcast (con tale trama si comportano esclusivamente in modalità relay).

A differenza dei bridge, gli hub (ripetitori fisici di livello 1) ripetono bit a bit e non possono ovviare alle collisioni: il bridge invece può accorgersi della collisione e mandare il pacchetto che collide in una coda d'uscita in modo da non avere una collisione.

Il bridge associa una porta ad ogni indirizzo MAC di destinazione che conosce (FDB, forwarding database). Se la destinazione è la porta da cui è arrivata, butto via la trama (filtering), altrimenti la mette sulla porta corrispondente per l'instradamento (relay) (slide 47). Se la tabella è vuota ho una trasmissione broadcast eccetto sulla porta di provenienza. I bridge, tramite il **backward learning**, possono costruire la topologia della rete in modo autonomo. Controllando gli indirizzi sorgente delle trame, associo indirizzi MAC alle porte di provenienza. Le entry nel FDB sono temporizzate: se non c'è traffico da una macchina, dopo un po'lo switch cancella l'entry relativa nella tabella. Questo per supportare mobilità interna dei terminali.

Il bridge mi permette di utilizzare meglio la banda della rete. Se il traffico fosse tutto locale nelle partizioni, avrei una capacità doppia rispetto alle stesse lan con ripetitori e senza bridge.

Se introduco dei cicli il meccanismo di bridging trasparente non funziona più. Bisogna “tagliare” il ciclo per far tornare tutto a gerarchia ad albero. Si utilizza l'algoritmo di **spanning tree**: si elegge un bridge come “radice dell'albero” e si crea un albero decidendo, per ogni bridge, quali sono le porte attive e quali sono le porte in blocco (che realizzano il taglio dei cicli). Se non faccio così alcuni pacchetti rimangono a circolare indefinitamente nella rete finchè non collassa.

Come eleggere la radice? Si elegge il bridge con un bridge ID minore tramite scambio di messaggi. Lo scambio di messaggi pesa anche i lati dell'albero e permette di trovare l'albero di supporto minimo.

In realtà adesso non ci sono più reti a mezzo condiviso perchè dato il basso costo degli switch ormai ho degli alberi di host senza collisioni. Il problema ora è capire quando conviene cambiare gli switch con i router (sicurezza). Questo mi permette anche di avere collegamenti fisicamente più lunghi e collegamenti full-duplex.

2. Indirizzamento e inoltro

Internetworking: per consentire il colloquio occorre tra reti, bisogna aggiungere dispositivi (*gateway* o *router*) capacid di colloquiare con calcolatori della propria rete e con altri router.

IP: si richiedono funzionalità locali minime(indirizzamento locale, trasferimento di pacchetti locali, indirizzamento broadcast).

Standard Internet: documenti pubblici **RFC** (Request For Comments), coordinati da IETF (www.ietf.org)

Ci sono organismi che si occupano di indirizzi e di nomi (IANA,ICANN), assegnando indirizzi IP e nomi simbolici globali.

2.1 Indirizzi IP

IPV4: gruppi da 32 bit raggruppati in byte (4x8 bit). Notazione decimale di ogni Byte (0-255).L'indirizzo è diviso in due parti, **netID** (identificazione di rete locale) e **HostID** (identificazione della macchina). A parità di rete locale ho condivisione del netID tra i vari host. I router lavorano indirizzando su netID. Esistono classi di indirizzamento differenti per utilizzi diversi:

Classe A 8 NetID, 24 HostID. Header 0 (0-127)

Classe B 16 NetID, 16 HostID. Header 10 (128-191)

Classe C 24 NetID, 8 HostID. Header 110 (192-223)

I primi bit (header) permettono di distinguere la classe. I rimanenti sono indirizzi riservati per scopi particolari

Classe D 32 Multicast. Header 1110 (224-239)

Canali multicast, per applicazioni speciali

Classe E 32 Future Use. Header1111 (240-255)

Riservato per usi futuri.

Indirizzi Speciali:

- *HostID tutti a 0*: indica la rete nel suo complesso, il cui indirizzo è contenuto nella parte NetID (ed è usato solamente nelle tabelle di instradamento).
- *HostID tutti a 1*: è l'indirizzo broadcast della rete indicata nella parte NetID (funzionalità usualmente non supportata).
- *Tutti i 32 bit a 1*: indirizzo broadcast limitato. E'il broadcast all'interno della stessa sottorete dell'origine del pacchetto. Il pacchetto non oltrepassa i router.
- *NetID a 0*: indica l'host il cui indirizzo è specificato in HosTID, host sulla stessa rete del mittente
- *Tutto a 0*: indica il mittente stesso del pacchetto. Serve quando l'host non conosce ancora il suo indirizzo.
- *Primo ottetto a 127*: indirizzo loopback verso se stesso, usato dai sistemi operativi per testare gli stack protocollari.

I router hanno più interfaccie verso più reti (almeno 2), per cui hanno un indirizzo IP per ogni interfaccia: gli indirizzi IP identificano quindi le interfaccie dei nodi, non i nodi stessi. Un nodo ha almeno tanti indirizzi IP quante sono le sue interfaccie di rete (a differenza delle reti in standard OSI che identificano i nodi e non le interfaccie). Inoltre gli indirizzi IP non hanno una corrispondenza geografica (non è vero che indirizzi numericamente adiacenti siano anche geograficamente adiacenti). Normalmente ai router vengono assegnati gli ultimi indirizzi disponibili nella rete.

Le reti private accedono ad internet accedendo a nodi di accesso forniti da ISP (Internet Service Provider) locali. Normalmente anche gli ISP sono organizzati gerarchicamente (ISP Locali, regionali, nazionali, internazionali).

Intranet: reti autonome e private basate su tecnologia IP (AS, autonomous system). I router all'interno della intranet sono detti IG (Interior Gateway) mentre quelli connessi ad internet sono detti EG (Exterior Gateway) (Nelle intranet si cerca di lavorare a livello 2 con i bridge invece che a livello 3 con i router perchè mantengo la *mobilità*...non devo preoccuparmi, se mi sposto, di mantenere il mio indirizzo IP perchè a livello 2 mi baso sul MAC address che è invariabile).

In generale ho corrispondenza 1:1 tra un NetID e una rete locale. Può succedere (ma mai viceversa) che ad una rete locale siano associati più di 1 NetID (utile ad esempio per sovrapporre classi di indirizzamento in una rete con più host di quelli disponibili nella classe di partenza).

Inoltro (forwarding) e **instradamento** (routing)**:** l'inoltro rappresenta le regole con cui un pacchetto viene inoltrato da una porta di ingresso ad una porta di uscita in un router, basandosi sulla *lettura* di una tabella di instradamento. L'instradamento invece è un algoritmo che sceglie il percorso in rete di un pacchetto, andando a *scrivere* le tabelle di instradamento dei router della

rete. Nei bridge la funzione di routing viene svolta intrinsecamente dallo spanning tree e dai relativi meccanismi.

Il protocollo di instradamento descrive gli scambi di pacchetti tra i router per scrivere in modo automatico le tabelle, implementando la logica di instradamento descritta nell'algoritmo di routing.

Inoltro diretto ed indiretto: applicate sia agli host che ai router. Diretto, quando la destinazione è contenuta all'interno della stessa rete, indiretto quando la destinazione si trova in un'altra rete locale.

Diretto: ho due host A e B (ciascuno con indirizzi IP e MAC). B deve mandare un pacchetto ad A. Se ha una tabella di corrispondenza IP<-->MAC , può costruire un pacchetto di livello 2 con il MAC del destinatario come destinazione, e i corrispondenti IP nell'overhead di livello 3. La trama spedita ha come sorgente MAC di B e destinazione MAC di A, e i relativi indirizzi IP nei campi sorgente/destinazione del pacchetto IP.

Indiretto: B deve spedire a A, ma la destinazione A è fuori dalla rete. L'host quindi passa il pacchetto al router. Il router può, grazie alle tabelle di inoltro, mandare avanti un pacchetto fino a destinazione. La distinzione diretto/indiretto si fa confrontando i campi NetID sorgente e destinazione. B costruisce il pacchetto andando a recuperare l'indirizzo MAC del router. A livello 3 il router vedrà che l'indirizzo IP destinazione non è nella stessa rete (e non ho corrispondenza sul MAC destinazione, che è quello del router), per cui instraderà il pacchetto all'esterno verso la destinazione.

I router seguono criteri simili, ma solitamente hanno più interfaccie per effettuare un inoltro diretto. Gli host hanno un default gateway (router), i router invece hanno una tabella di instradamento di più router e sanno qual'è la via migliore per instradare. Guardando il NetID capiscono se possono fare un inoltro diretto oppure se devono fare un inoltro indiretto verso un altro router. L'host deve conoscere il MAC adress del router, e lo recupera dalla tabella di corrispondenza.

La trama livello 2 ha come indirizzo MAC destinazione l'indirizzo del router. Dentro il pacchetto IP invece ho il pacchetto che volevo spedire fuori dalla mia rete locale, infatti la destinazione del pacchetto IP è quella originale.

Inoltro nei router

Inoltro IP basato su indirizzo destinazione, e la decisione riguarda solo il prossimo passo verso la destinazione. Tabella ha un campo *net-hop* che dice qual'è il prossimo nodo cui passare il pacchetto. Non è in grado di calcolare l'intero percorso fino alla destinazione. Il router ha più interfaccie: si controlla l'indirizzo destinazione con gli indirizzi corrispondenti alle interfaccie (confronto dei *net-id*). Questo avviene per i pacchetti generati nel router, ma anche per quelli che vi vengono instradati. Nell'inoltro indiretto, il router deve instradare il pacchetto verso uno dei tanti router cui è collegato. Si decide qual'è il prossimo passo dell'instradamento (*next hop*, facendo il check del campo *net-id*), utilizzando le tabelle di routing.

Le **tabelle di routing** sono un elenco di reti di destinazione (net-id significativo, la parte host-id è posta a 0 perchè è ininfluente) in cui sono indicati gli indirizzi IP dei router adiacenti a quello considerato (ovvero raggiungibili con inoltro diretto e appartenenti alla sottorete cui il router è collegato).

In realtà questo è il meccanismo base, ci sono delle complicazioni dovute alle modalità di assegnamento degli indirizzi (che non sono più legate alle classi canoniche). Es: organizzazioni private con indirizzo di rete di classe B suddividono la loro intranet in sotto-reti IP locali (partizionate con router interni).

Si consente di usare una parte dell'**host-id** per indirizzare la sottorete.

[	Net-ID	][	Host-ID		]	|				
[	**Network**	**][**	**SubNet**	**][**	**Host**	**]**	**|**	32bit		
[1 1			**.. 1 1]**	**[0 0 .... 0]**		|			

La divisione è indicata da un campo ulteriore **netmask**, che è una sequenza di 1 che indicano la parte Network+Subnet ed una sequenza di 0 che indicano la parte Host. Questa modifica ha influito sulla politica di instradamento. Ad ogni modo questa cosa viene fatta all'interno di reti private, che generalmente sono comunque raggiungibili da 1 sola via: all'esterno della rete si vede un unico blocco di indirizzi della vecchia rete di classe B. Sarà compito dei router interni per distinguere le sottoreti interne. La netmask rappresenta un confine mobile tra Net-ID e network.

Si può indicare la netmask in valori binari e decimali, oppure specificando il numero di 1 consecutivi (*prefisso di rete*) ovvero sapere dove si trova il confine (**121.175.21.0/24**). La netmask non viaggia nei pacchetti IP, è una configurazione che risiede nelle interfaccie e può essere scambiata solo grazie ai protocolli di routing.

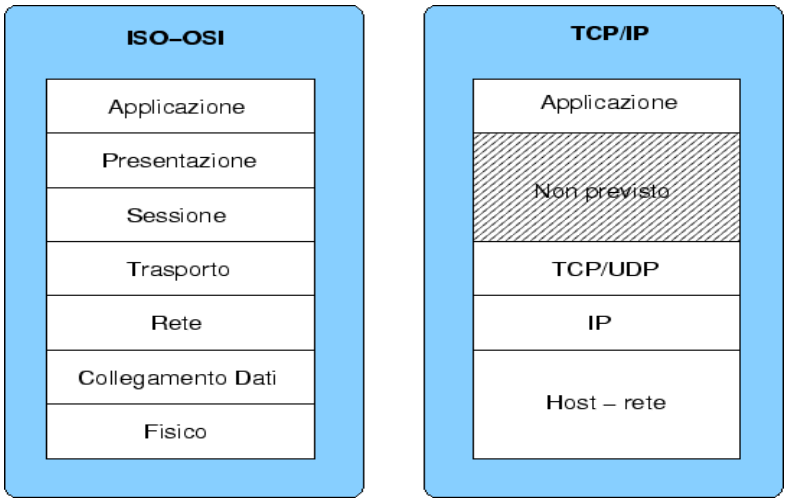
Per operare con le regole di inoltro definite prima, ciascuna interfaccia deveessere configurata sia con indirizzo IP sia con la netmask. Inoltre le tabelle di instradamento devono contenere un ulteriore attributo indicante la netmask per ogni tupla.

Per inoltrare un pacchetto bisogna verificare se i pacchetti appartengono a una sottorete delle interfaccie, facendo un AND bit a bit tra indirizzo dell'interfaccia e netmask e tra indirizzo destinazione e ip. Se il confronto è lo stesso, si procede all'inoltro.

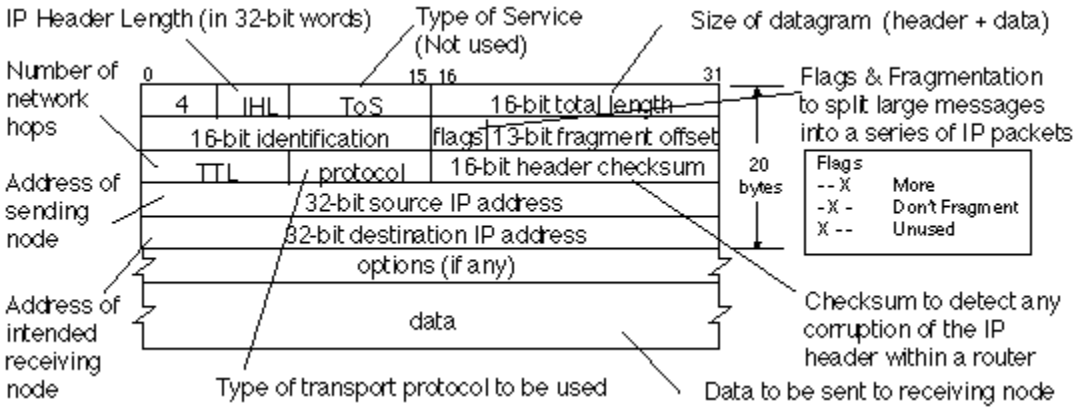
Il router itera l'operazione su ogni interfaccia, effettuando il confronto riga per riga. Se ho un esito positivo per più righe, lo mando nella riga con il prefisso più lungo (perchè rappresentano accessi migliori verso sottoreti). Si aggiunge in tabella anche un entry con network e netmask a 0, che instrada verso il default router.

Supernetting: la netmask in generale definisce un *confine mobile* tra la parte rete e la parte host. Per cui posso sfruttare questa possibilità per implementare altre cose. Ad esempio nel supernetting si usa il netmask per raggruppare 2^n sottoreti in una rete più grande. Questo è stato introdotto a causa della scarsità di indirizzi di Classe B disponibili.

Routing classless: instradamento basato soltanto sulle netmask e non più legato alle vecchie classi (dovuto all'introduzione di subnetting e supernetting). Si possono compattare le tabelle di routing raggruppando (all'interno dei router) più reti adiacenti associate allo stesso router di next-hop (indotto dall'assegnamento su base geografica degli indirizzi). La riga rimanente indicherà la rete fittizia che raggruppa le sottoreti. Si può applicare supernetting e subnetting in sequenza.



3 Internet Protocol (IP)



Lo stack IP si colloca ai livello 3 della pila ISO/OSI (network layer). Il suo funzionamento è garantito dai protocolli ICMP, ARP e di routing. Rende trasparenti i livelli inferiori, e comunica ai livelli superiori costituiti dai protocolli di trasporto (TCP e UDP).

Il pacchetto IP è costituito da un header (20 byte), un campo opzioni e un payload dati. L'header è costituito di numerosi campi.

Ver (4 bit): Versione del protocollo, studiamo la versione 4

HLEN (8 bit): indica la lunghezza dell'header, espressa in parole da 32 bit.

TOS (8 bit): Type of Service (DS Field), può essere usato per gestire la priorità nelle code dei router.

TL (16 bit): Total Length, indica la lunghezza totale del pacchetto in byte: MAX 2^{16} = 65536. TL – (HLEN*4) = dimensione Payload (in byte)

Frammentazione dei pacchetti: alcuni protocolli cui IP si appoggia richiedono dimensione massima

di pacchetto (MTU) inferiore a 65536 bytes (esempio Ethernet: max 1500 bytes/pacchetto). Se il pacchetto è troppo grosso, prima di essere propagato ai livelli inferiori viene spezzato in frammenti, ciascuno con il proprio header. Questi frammenti verrano ricomposti (deframmentati) dall'entità IP del destinatario.

Per operare la frammentazione vengono usati i campi **ID, Flags, Frag.Offset**

Identification (ID): identifica in modo univoco tutti i frammenti di uno stesso pacchetto. Scelto dall'entità IP che effettua la frammentazione.

Frag Offset: I byte del pachetto originali sono numerati da 0 al valore totale di lunghezza. Il campo Fragment Offset, in ogni frammento, riporta il numero di sequenza del primo byte di frammento.

Flags: 3 Bit M,D e uno non utilizzato. M (More) posto a 0 solo nell'ultimo frammento. D viene posto a 1 quando non si vuole che venga applicata la frammentazione lungo il percorso. Se essa fosse necessaria, non viene applicata ma si genera un messaggio di errore.

TTL (Time To Live): viene settato ad un valore numerico elevato da chi genera il pacchetto, e viene decrementato da ogni router attraversato. Se un router decrementa il TTL fino a 0, il pacchetto viene scartato e viene notificata la sorgente tramite un messaggio di errore (ICMP Time Exceeded)

Checksum: ... ogni router decrementa il TTL e ricalcola il checksum

Protocol: serve a fare multiplazione (offro un servizio di comunicazione a più entità di livello superiore in parallelo).

Campi opzionali: i primi 20 bytes dell'header sono sempre presenti, la lunghezza dell'header non deve superare i 60 byte. Ho quindi a disposizione 40 bytes per aggiungere campi opzionali. Hanno una struttura simile a un protocollo.

[	**8bit**	**|**	**8bit**	**|**	**variabile**	**]**
	code		length		data	

Il campo code è suddiviso

[	**1bit**	**|**	**2bit**	**|**	**5bit**	**]**
	copy		class		option	

Di seguito alcune opzioni.

- **record route:** usato se voglio ricostruire il percorso utilizzato alla destinazione. Ho 9 campi da 32bit, e ogni router che gestisce il pacchetto scrive il suo indirizzo. In realtà con ICMP c'è un metodo più semplice per scoprire laposssibili route di un pacchetto.
- **Source routing**: la sorgente del pacchetto scrive nel pacchetto stesso la sequenza di nodi da attraversare. Strict source routing**:** il pacchetto attraversa soltanto i router nella sequenza, nello stesso ordine. Loose source routing**:** insieme minimo di router da attraversare.
- **Timestamp**: può essere utilizzato per sincronizzazione remota di apparati. Normalmente però i protocolli di sincronizzazione esistono, ma sono realizzati a livello applicativo (es. NTP, Network Time Protocol)

La corrispondenza tra indirizzi IP e indirizzi fisici è realizzata tramite le tabelle di corrispondenza ARP ed il protocollo ARP (che si preoccupa di generare dinamicamente ed automaticamente la tabella ARP in ogni host).

ARP (Address resolution protocol) protocollo di livello 3 incapsulato in trame ethernet. Basato su

meccanismo richiesta/risposta. Chi ha bisogno di conoscere un'assocazione genera un messaggio ARP request in broadcast, l'host interessato risponde alla richiesta con un messaggio ARP response. La risposta viene registrata da chi l'ha richiesta in una tabella di ARP locale (cache ARP). Ogni entry ha un tempo di timeout. La necessità di un indirizzo broadcast in una rete è dovuta proprio alla necessità di implementare queste funzionalità (altrimenti IP e ARP non funzionerebbero).

[source:source-MAC|destin:broadcast][ARP REQUEST]

Arp Request:

[IP-S: ... | MAC-S: ...][IP-D: ... | MAC-D: xxxxxxxxxxxxx]

Il campo MAC-D è quello da completare. Tutti gli host leggono la richiesta, tutti gli host leggono la risposta, solo la macchina IP-D risponde e scrive nel pacchetto il suo MAC nel campo MAC-D. Nella risposta l'indirizzo MAC è scritto sia in MAC-D che in MAC-S, questo perchè il livello ARP legge solo la parte ARP ma per mandare il pacchetto in rete ho bisogno di completare anche il livello ethernet per far circolare il pacchetto sulla rete.

La necessita del timeout sulle righe della ARP cache è dovuta al fatto che non è assicurata la corrispondenza eterna tra un IP e un indirizzo MAC (esempio, rete con DHCP con assegnamento IP dinamico).

Per IP è fondamentale che ci sia una corrispondenza 1:1 tra i domini di broadcast e le sottoreti, questo per come è configurato ARP. Non possono coesistere più domini di broadcast in un'unica sottorete IP.

E'possibile invece che ad uno stesso dominio di broadcast corrispondano più sottoreti IP. A e B sono sullo stesso dominio di broadcast, ma hanno indirizzi appartententi a sottoreti diverse (però con ugual netmask). A manda ARP request a B per inoltro indiretto, il router capisce che A e B sono sulla stessa sottorete. Per poter funzionare, il router deve avere un indirizzo per entrambe le sottoreti. E'lecito quindi assegnare più indirizzi IP ad una stessa interfaccia.

I router di nuova generazione consentono di operare con modalità **Proxy ARP**: in questa modalità tutti gli host vengono configurati con netmask poste a 0. B cercherà quindi di fare un ARP request per collegarsi in modo diretto verso qualunque destinazione. B riceve il pacchetto ARP e risponde. Cosa faccio però se devo uscire dalla LAN? B vuole uscire dalla LAN, invia una ARP Request (perchè non può distinguere la richiesta da quella locale). Il router riconosce che la ARP request non è compreso tra quelli definiti in quella interfaccia, e mediante un processo di Proxy ARP risponderà a B con il suo MAC. In questo modo, senza che B se ne accorga, i pacchetti di B verso l'esterno saranno automaticamente inviati al router che saprà che li deve inoltrare all'esterno (mascherando l'indirizzo del gateway agli host!). Il router deve sapere su che indirizzi applicare il proxy ARP.

ICMP: protocollo per gestire messaggi di servizio fra host e routers, o per gestire errori di protocollo a livello rete. ICMP è incapsulato in IP (nel payload), anche se logicamente è un protocollo di livello superiore.

Echo: echo-request, echo-reply. Messaggi ICMP scambiati tra due host, richiesta e risposta. Serve a capire se tra i due host c'è un percorso di rete funzionante, e se l'host destinazione è attivo. Se volessimo conoscere qual'è anche il cammino, possiamo usare le opzioni di record route di IP. Oppure si usa un metodo alternativo (implementato ad esempio dall'applicazione traceroute): vengono elencati tutti gli indirizzi dei router attraversati. Viene usato il TTL: si mandano pacchetti verso la destinazione con TTL da 1 a MAX. In questo modo scopro tutti i nodi, perchè mi risponderanno con Time exceeded.

Destination Unreachable: quando un router scarta un pacchetto, genera un messaggio di errore che invia alla sorgente del pacchetto. Contiene informazioni sulla causa dell'errore.

Time exceeded: quando il TTL viene settato a 0, si scarta il pacchetto e si genera un pacchetto ICMP Time exceeded da inviare alla sorgente.

Redirect: quando si conosce una route migliore, un router puo dire alla sorgente di usare il gateway migliore con questo messaggio.

Timestamp request/reply: farsi inviare il timestamp da un particolare nodo della rete.ss

Address mask request/reply: scambio di netmask tra due nodi della rete.

RARP (Reverse ARP): serviva per gestire macchine non configurate: gruppi di workstation (senza disco, ovvero senza spazio per salvare la configurazione di rete) che avevano l'immagine del SO messa in modo centralizzato in un server: queste macchine dovevano effettuare il bootstrap via rete. Le macchine, sapendo il loro indirizzo fisico, richiedevano al server il proprio IP con una richiesta RARP. Non è più usato.

Necessità, legata all'ingrandimento degli ISP, di fornire indirizzi IP al momento della connessione dell'utente (configurando la connessione all'atto dell'instaurazione della stessa). Scopo (iniziale): gestione di un numero elevato di utenti usando un insieme ridotto di indirizzi pubblici utilizzabili (pochi utenti contemporaneamente online). In ogni caso c'è necessità di uso dinamico degli indirizzi assegnati alle varie macchine in una rete (configurazione centralizzata su un server, piuttosto che distribuita sui singoli host).

Tipi di associazione

- **statica:** a parità di macchina (MAC) si riceve sempre lo stesso IP. Si realizza con una tabella di corrispondenza MAC <-> configurazione.
- **automatica**
- **dinamica:** insieme degli IP minore degli host che possono usarlo (usato per gli indirizzi pubblici, che sono scarsi). Utilizzo di timeout o di procedure di rilascio esplicito. E'possibile negare la richiesta.

Protocollo DHCP (Dynamic Host Configuration Protocol): evoluzione di BOOTP. Protocollo di tipo client-server.

Configurazione di rete: IP, Gateway, Netmask, DNS. Opzionalmente si indica un file da scaricare dopo la configurazione (boot SO in remoto, soprattutto per applicazioni embedded – es. Telefoni VoIP).

I protocolli client-server non sono simmetrici (il formato dei messaggi in una direzione è diverso dal formato nel messaggio opposto. Modello TCP/IP è client-server.), i protocolli P2P sono simmetrici (non vi è differenza nel formato dei messaggi, perchè i nodi sono paritari. Modello OSI è intrinsecamente P2P). TCP è simmetrico, ma c'è sempre una delle due parti che inizia le connessioni, per questo non è P2P. Nei pacchetti che seguono il modello ISO/OSI ho indicazione dei Source-SAP e Dest-SAP, mentre nei pacchetti che seguono il modello TCP/IP ho solo l'indicazione del tipo di protocollo.

Funzionamento: basato su scambio di messaggi. Il client invia un messaggio in broadcast (DHCP Discovery), allegando il proprio indirizzo fisico. I server (possono essere più d'uno) rispondono con un messaggio DHCP Offer. Il client analizza l'offerta e invia una richiesta specifica per quella particolare configurazione (DHCP Request, in broadcast), e il server scelto risponde con la configurazione (DHCP Ack). Alla fine dell'utilizzo, il client manda un messaggio DHCP Release per cancellare l'allocazione (oppure la configurazione scade automaticamente allo scadere del **lease time**). Se non ha più risorse, il server DHCP invia un messaggio di rifiuto (oppure non risponde).

E'possibile utilizzare più server DHCP, oppure usare dei Relay. Infatti il server deve essere

fisicamente collegato alla rete che serve. Se avessi tante sottoreti, dovrei avere tanti server DHCP per ogni sottorete (prima dell'assegnamento ho solo accesso al livello 2!!). Si usano i Relay: i DHCP Relay ricevono in broadcast i messaggi, li impachettano con il protocollo IP e li spediscono ai server deputati a servire quelle richieste. In questo modo ho più flessibilità.

DHCP si appoggia su UDP per il trasporto (pacchetti di livello applicativo incapsulati in pacchetti di trasporto). I messaggi dei client hanno IP sorgente 0.0.0.0 e destinazione 255.255.255.255 (porta sorgente 68, porta destinazione 67). Questo finchè il client non riceve la configurazione (necessario per permettere al client non configurato di usare il livello 3).

4 Transport Layer (TCP/UDP)

Compiti: instaurazione di collegamenti logici tra applicazioni su host differenti. Presente solo sugli host e non sugli apparati di interconnessione. Rende trasparente il trasporto fisico alle applicazioni (si segmentano i messaggi applicativi in PDU di livello 4). E'possibile gestire più applicazioni contemporaneamente (multiplexing/demultiplexing: implementate tramite le **porte**)

Porte: indirizzi di 16 bit, assegnati ai processi applicativi per permettere la comunicazione (0 – 65535).

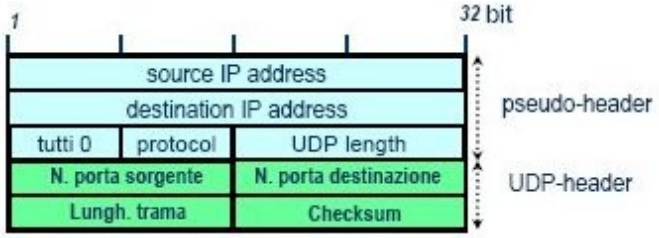
L'unione di indirizzo IP e # di porta prende il nome di **socket**, e permettono l'identificazione univoca di un processo applicativo in esecuzione su un host.

Il **servizio di trasporto** può essere di vari tipi:

- **TCP**: servizio affidabile, connection-oriented
- **UDP**: servizio non affidabile, di tipo datagram (connectionless)

Essi sono implementati come moduli nei sistemi operativi, che forniscono l'API per l'utilizzo. Il sistema operativo associa due code (IN/OUT) ad ogni processo associato ad una porta.

Protocollo UDP: rispetto ad IP non aggiunge niente se non indirizzamento delle applicazioni (source e destination port) e controllo di errore sull'header. E'un protocollo datagram, non garantisce consegna, non effettua controlli di flusso o errore.



Protocollo TCP: a differenza di UDP si implementa il trasporto affidabile, in corretta sequenza e senza errori sui dati. Implementa un meccanismo di trasporto affidabile dei dati, e protegge la rete dai sovraccarichi (controllo di congestione, controllo di flusso). L'affidabilità del servizio è strettamente correlata con la modalità a connessione: la connessione viene instaurata esplicitamente, si effettua il trasferimento e infine si decide esplicitamente la terminazione della connessione con una *segnalazione esplicita* (setup, fase dati, tear-out). Le connessioni TCP instaurate sono solo di tipo **full-duplex**.

Il controllo di flusso è regolato sulla capacità del ricevitore di ricevere i dati: è basato su una **sliding window.**

Vi sono anche meccanismi di controllo di congestione: il flusso dati si regola anche sulla situazione

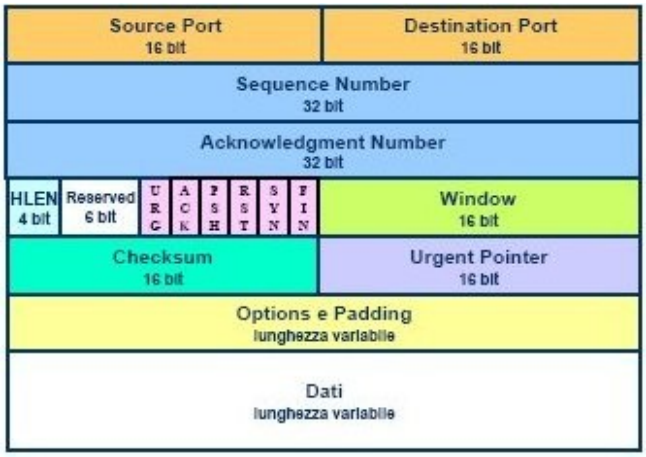
del traffico di rete. Si riduce il troughtput se si profilano situazioni congestionate. Queste vengono scoperte da TCP studiando gli eventi di perdita pacchetti.

Basato sul meccanismo della sliding window, che è dimensionata automaticamente a seconda dello stato di traffico della rete. Principio di equità (*fairness*) dell'accesso alle risorse di rete. Si garantisce un minimo di servizio a tutti gli utilizzatori, anche nelle condizioni pessime.

Il flusso dati su una connessione TCP è suddiviso in segmenti che possono essere trasmessi in IP. I segmenti hanno dimensioni variabili (e arbitrarie). Il layer TCP memorizza i dati in un buffer, e periodicamente questo buffer viene svuotato e con i dati ricavati si crea un segmento. Siccome le dimensioni del segmento sono critiche per la prestazioni, generalmente si aspetta un minimo di quantità informativa prima di creare e spedire il segmento su rete. E'possibile forzare la creazione del segmento in applicazioni con particolari requisiti temporali.

Il ricevitore è sempre in grado di sapere il livello di riempimento del buffer di ricezione: in questo modo, tramite il campo window del pacchetto, è in grado di indicare al mittente la dimensione consigliata della finestra in modo da evitare overflow del buffer.

Il controllo d'errore è realizzato tramite un meccanismo sui pacchetti di tipo **go-back-n**. I byte trasmessi sono numerati in sequenza, l'header di ogni segmento riporta il numero del primo byte che ne fa parte. Chi riceve riscontra i dati ricevuti, inviando il numero di sequenza dell'ultimo byte che ci si aspetta di ricevere correttamente ed in sequenza. Il mancato riscontro forza un meccanismo di timeout e l'invio ripetuto dei dati mancanti. I segmenti sono quindi numerati, ma relativamente al byte più recente. In questo modo si ritrasmettono solamente i dati che servono (quelli mancanti) e non l'intero segmento.



- **Source, Destination port**: 16 bit
- **SN**: # di sequenza del primo byte presente in payload
- **AN**: # prossimo byte che si aspetta di ricevere (se ACK valido)
- **HLEN**: lunghezza dell'header (multiplo di 32 bit)
- **Window**: il valore della finestra comunicato da ricevitore a trasmettitore
- **Checksum**: CRC su header virtuale con aggiunti IP sorgente e destinazione
- **Options/Padding** : campi opzionali e riempimento in multipli di 32 bit.
    1. MSS (Maximum segment size): decisa dal mittente a tempo di setuè
    2. Fattore di scala: definisce unità di misura della finestra (default: 1 Byte, se no

2 ^ fattore di scala)

- **Flags**
    1. URG: dati urgenti, puntatore al primo byte dei dati urgenti
    2. ACK: valore 1 se ACK valido -> AN ha un SN significativo
    3. PSH: comando di PUSH richiesto dal trasmettitore
    4. RST: reset della connessione senza tear-down esplicito
    5. SYN: sincronizzazione, comunica i numeri di sequenza iniziali
    6. FIN: chiusura esplicita di una connessione

I socket permettono l'identificazione univoca anche in caso di più connessioni alla stessa porta (di un server di posta ad esempio)

Fase di setup: applicazione server fa una *passive open*, comunicando allo stack TCP del SO che è pronto ad accettare nuove connessioni su una data porta. L' applicazione client fa una *active open*, richiedendo allo stack TCP l'effettuazione di una connessione verso un dato socket.

Client TCP: Esso genera anche un numero di sequenza iniziale e manda un messaggio SYN contenente questo SN (*Sequence Number*) (per evitare problemi in caso in cui il setup non va a buon fine per packet loss)

Server TCP: riceve SYN, manda SYN/ACK con altro SN generato a caso, e AN (*Acknowledgment Number*) uguale a quello ricevuto dal client (*riscontro*)

Client TCP: riceve SYN/ACK, invia ACK con AN uguale all'SN ricevuto dal server. Nel payload inserisce i primi dati della connessione con SN pari a quello iniziale generato. Notifica all'applicazione client che la connessione è instaurata.

Server TCP: riceve ACK, notifica all'applicazione server che la connessione è instaurata.

Successivamente chi chiude la connessione, invia un messaggio FIN con gli ultimi dati nel payload. Chi riceve FIN, riscontra con un ACK. Nell'altra direzione la connessione rimane comunque aperta, con la finalizzazione dei dati da inviare. Alla fine anche dall'altra parte si invia un FIN, replicato con un ACK per concludere definitivamente la connessione.

Se si invia un messaggio con flag RST, esso causa la chiusura della connessione senza ulteriore invio di dati.

Controllo di errore TCP: recupero di pacchetti persi (a causa di overflow delle code dei router, ad esempio). Meccanismo di tipo **go-back-N con Timeout**. Finestra di trasmissione è dimensionata dal meccanismo di controllo di flusso e congestione (valore di N). Il timeout viene iniziato al momento della trasmissione, si ha ritrasmissione quando scatta il tempo massimo.

Controllo di flusso TCP: ricevente responsabile dei dati in ingresso. Decide quanta roba vuole ricevere, e la comunica al trasmittente. Esso indica esplicitamente la dimensione della finestra di ricezione in ogni trama verso il trasmittente.

RCVWND (*Receive Window*): spazio di buffer disponibile per la ricezione dati. Ha dimensione pari allo spazio ancora libero.

SNDWND (*Send Window*): buffer di trasmissione, tiene traccia dei dati trasmessi ma non riscontrati e della dimensione RCVWND. Ha dimensione pari alla distanza tra il primo byte non ancora riscontrato e l'estremo della finestra.

Silly Window Syndrome:

1. Ricevitore svuota lentamente, invia segmenti con finestra piccola, trasmettitore invia segmenti corti con molto ovh. Soluzione: ricevitore "mente", indicando una finestra nulla finchè il suo buffer non è vuoto per una porzione pari almeno a MSS o a metà del buffer *min(1/2*RCV_BUF,MSS)*
2. Trasmettitore genera dati lentamente, invia segmenti piccoli man mano che vengono prodotti.Soluzione: TCP sorgente invia prima porzione dati, anche se corta. Altri segmenti inviati sse buffer di uscita contiene dati almeno per MSS, oppure alla ricezione di un'ACK del segmento precedente.

Meccanismo PUSH: alterazione del normale funzionamento quando ci sono dati che richiedono la consegna immediata all'applicazione ricevente. Applicazione invia comando PUSH allo stack TCP, si setta il flag PUSH nel pacchetto per informare anche l'altra applicazione di comportarsi così.

Meccanismo URGENT: si possono marcare i dati come urgenti, identificati all'interno del flusso dati e non seguono le regole del controllo di flusso.

Come stabilire il valore ottimo del timeout? Esso dipende in modo molto marcato dal ritardo della rete. TCP stack calcola un valore opportuno stimando RTT (*Round Trip Time*): algoritmi di Karn e Jacobson

$\{RTT^{(i)}\}$ (campioni di RTT): tempo che passa tra trasmissione segmento e ricezione del relativo ACK.

Smoothed RTT, calcolato dal trasmittente con l'algoritmo di Jackobson

$$SRTT^{(i)} = (1-\alpha)\, SRTT^{(i-1)} + \alpha\, RTT^{(i)}$$

α generalmente compreso tra 0 e 1 (tipicamente 1/8)

Viene anche stimata la deviazione standard, di cui si calcola un valore filtrato

$$DEV = |RTT^{(i)} - SRTT^{(i-1)}|$$

$$SDEV^{(i)} = 3/4*SDEV^{(i-1)} + 1/4*DEV$$

Il timeout viene quindi calcolato come $TOUT = SRTT + 2*SDEV$

All'inizio SRTT è posto a zero, e SDEV = 1.5s . Il valore iniziale del timeout è quindi 3s

Se è avvenuta una ritrasmissione si passa all'algoritmo di Karn: RTT non è aggiornato, il timeout è moltiplicato per un fattore fisso (tipicamente 2), si fissa un tetto massimo per il timeout, e dopo un numero massimo di ritrasmissioni la connessione viene chiusa.

Persistenza: Se il destinatario azzera RCVWND, la sorgente interrompe la trasmissione. Essa riprende quando destinatario invia un ACK con finestra diversa da zero. Se si perde l'ACK ho un deadlock. Soluzione: si usa un timer di persistenza attivato all'arrivo di un segmento con finestra nulla. Se scade (timeout = ritrasmissione) si invia un *segmento sonda* (probe). Se viene ricevuto un ACK si esce dallo stato critico, altrimenti si invia un altro probe al timeout.

Controllo di congestione: si implementa in modo efficace usando RCVWND e SNDWND. Nella rete non ci sono meccanismi sofisticati di controllo congestione a livello di rete, questo è delegato a TCP. Il controllo è di tipo end-to-end (secondo le caratteristiche stesse di TCP).

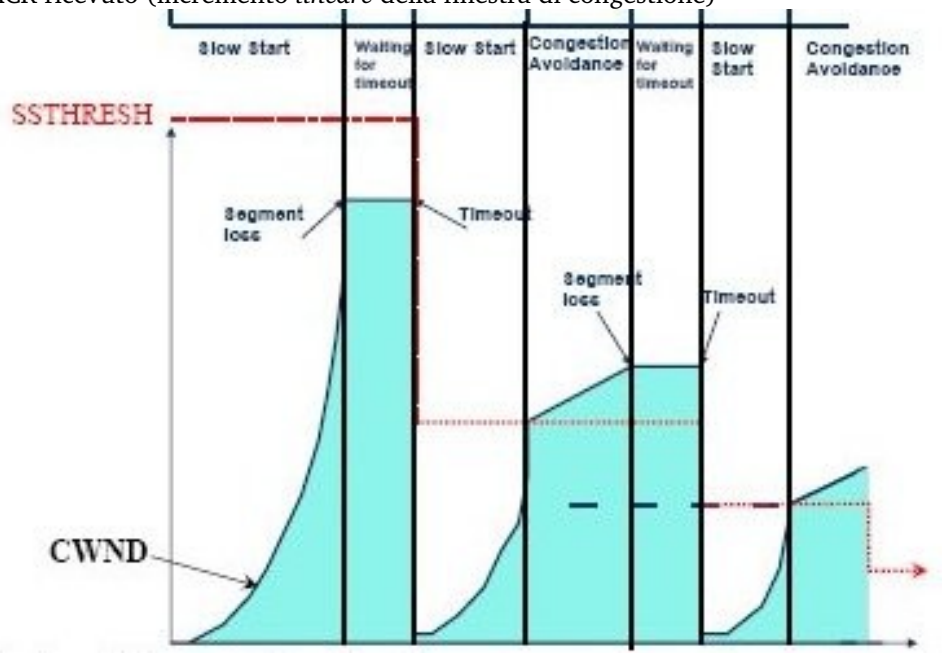
Il trasmettitore mantiene una *Congestion Window* (CWND) che varia in base agli eventi osservati (ACK, timeout). Trasmettitore non può trasmettere più di min(RCVWND,CWND).

Si interpretano le perdite di segmenti (scade il timeout di ritrasmissione): ad ogni evento di congestione, si riduce CWND. Questo valore viene aggiornato dal trasmettitore TCP tramite un algoritmo che si comporta a seconda dello stato del trasmettitore (identificato dalla variabile

SSTRESH)

- **Slow Start** (CWND < SSTRESH): inizialmente si pone CWND a 1*MSS. SSTRESH ad un valore molto elevato di default. Si incrementa CWND di 1 per ogni ACK ricevuto. (incremento *esponenziale*: si raddoppia ogni RTT). Il *Rate* di trasmissione è identificato da CWND/RTT [bit/s]
- **Evento di congestione**:
    1. si pone SSTRESH = max (2*MSS,0.5*FlightSize)
    2. Si pone CWND = 1
    3. FlightSize: byte trasmessi ma non riscontrati (di solito pari a CWND)

  Ne risulta che CWND < SSTRESH, e si torna nella fase di SS. Il trasmettitore trasmettte tutti i segmenti a partire da quello per cui è fallito il timeout (go-back-N). Il valore di SSTRESH è una stima della finestra ottimale che eviterebbe eventi futuri di congestione.
- **Congestion Avoidance** (CWND > SSTRESH): Si incrementa CWND di 1/CWND per ogni ACK ricevuto (incremento *lineare* della finestra di congestione)



Fast Retrasmit / Fast Recovery: implementati nella versione TCP Reno: se TCP riceve pacchetti fuori sequenza, invia subito un ACK con AN pari al segmento atteso. Alla ricezione del 3°ACK consecutivo duplicato (stesso AN), si pone

SSTRESH = max (2*MSS, 0.5*FlightSize)

Poi si ritrasmette il pacchetto indicato da AN, ponendo **CWND = SSTRESH + 3*MSS** . Per ogni ulteriore ACK duplicato si incrementa di 1 la CWND. Si trasmettono nuovi segmenti, se consentito dai valori CWND e RWND. Appena arriva un ACK di riscontro dei nuovi dati, si esce dalla fase di *fast recovery* e si pone di nuovo **CWND = SSTRESH = max (2*MSS, 0.5*FlightSize)**. Questo perchè se arrivano ACK duplicati, significa che sarà stato perso un pacchetto, ma quelli successivi a quello perduto sono arrivati, quindi non c'è congestione. Se non c'è congestione, si può incrementare CWND del numero di pacchetti sicuramente arrivati.

In condizioni ideali, TCP è in grado di limitare la congestione di rete, e di dividere in modo equo la capacità dei link tra i diversi flussi. In realtà ci sono numerosi fattori che discostano dal caso ideale:

- RTT differenti per flussi differenti
- buffer nei nodi minori del prodotto banda*ritardo
- valori medi sulla capacità dei flussi
- ritmo di trasmissione fortemente variabile

Algoritmo di calcolo FAIR-SHARE: vedere slide 76 capitolo 4 degli appunti per l'algoritmo)

5 Procolli Applicativi

Vi sono processi in esecuzione su sistemi remoti, necessitano di scambiare informazioni. I protocolli applicativi sono le regole ed i formati per la costruzione e lo scambio dei messaggi. Si utilizzano i servizi offerti dai livelli inferiori tramite i SAP (Service Access Point). Nell'architettura TCP/IP i SAP sono le **porte** e l'insieme porta/indirizzo IP viene denominato **socket**.

Architettura client/server: l'interazione avviene tra due strutture software, una delle quali (**server**) fornisce servizi, mentre l'altra (**client**) consuma i servizi offerti. Il client fa richieste per i servizi ed interpreta le risposte, mentre il server fornisce risposte e interpreta le richieste. I protocolli applicativi per questa architettura seguono il modello **request-response**.

Normalmente più client possono accedere allo stesso server, ed un client può effettuare più richieste contemporanee verso un server. I software client/server possono essere eseguiti in modalità parallela o seriale. Generalmente ,applicativi UDP funzionano in modo seriale, applicativi TCP funzionano in modalità parallela (applicativi multi-threaded).

HTTP: protocollo applicativo, basato su modello client/server. Il client richiede file identificati da URL, il server restituisce i files richiesti. E'un protocollo *stateless* (non c'è memoria sulle richieste effettuate). E'basato su TCP per il trasporto dei messaggi.

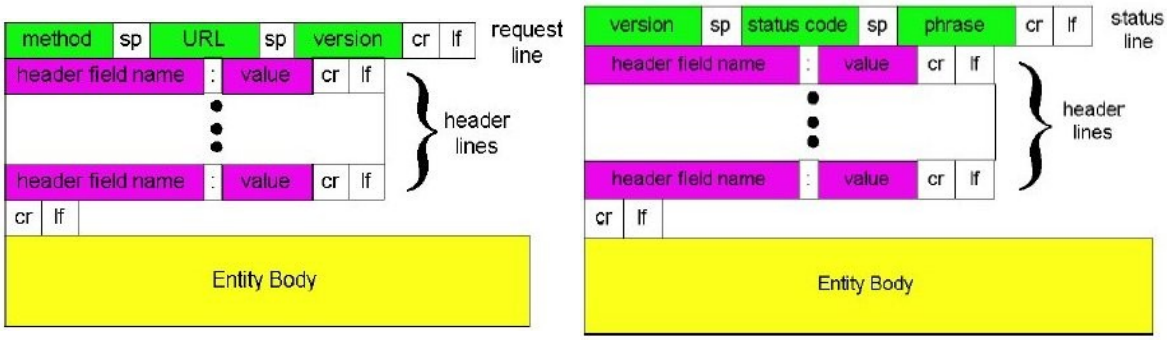
URL (Uniform Resource Locator): *method://host:port/path*

Il trasferimento di più oggetti (richiedendo ad esempio una pagina web) può essere effettuato in due modalità: *non-persistent connection* (HTTP 1.0) e *persistent connection* (HTTP 1.1)

1. **Non persistent**: si apre una connessione per ogni request/response. Si possono aprire un parallelo più richieste per ogni oggetto. Il server chiude la connessione dopo l'invio dell'oggetto
2. **Persistent**: la connessione al server rimane aperta e può essere utilizzata per trasferire oggetti relativi alla pagina oppure ad altre pagine. La connessione è chiusa basandosi su un timeout. Si può effettuare *pipelining* per velocizzare la risposta.

Messaggi HTTP:

- Richieste: GET – HEAD – POST – PUT
- Risposte: 1xx (Informational) 2xx (Succcess) 3xx (Redirection) 4xx (Client Error) 5xx (Server Error)
- Header: sono coppie in formato **name:value**, utilizzate per scambiare informazioni aggiuntive di servizio. Sono possibili più linee di header per ogni messaggio HTTP



Richiesta HTTP **Risposta HTTP**

Utilizzo di Proxy: i proxy forniscono una memoria cache, per fornire più velocemente documenti molto richiesti. Sono degli *application gateway*, ovvero instradatori di messaggi di livello applicativo (client/server).

Autenticazione: HTTP è stateless, non può riconoscere richieste successive dallo stesso utente. Supporta però un meccanismo di autenticazione tramite username e password (in chiaro)

Vedere slides.

6 Routing

6.1 Instradamento

Instradamento, funzionalità base implementata a livello 3. Permette il collegamento indiretto tra due nodi A-B tramite la collaborazione dei nodi intermedi. Commutazione dei pacchetti avviene grazie a indirizzo ed etichette poste nei pacchetti. Il SAP di uscita è correlato all'indirizzo nella **tabella di routing**. I criteri di scelta dei cammini sono definiti nella **politica di routing**. Il tipo di rete (*datagram, virtual circuit*) definisce il tipo di tabella utilizzata e i gradi di libertà nella scelta dei cammini.

In una rete broadcast non avviene instradamento e il traffico massimo smaltito dalla rete è pari al più alla capacità massima del canale. In una rete a maglie la trasmissione di pacchetti può essere effettuata in parallelo su più apparati, per cui diventa fondamentale una buona politica di instradamento per massimizzare il troughtput della rete.

Nelle reti IP il forwarding è *destination-based, next-hop based*. I pacchetti verso D che giungono da un router R seguono lo stesso percorso da R a D indipendentemente da come sono arrivati in R. Ne segue il vincolo: *l'insieme dei cammini da ogni sorgente verso D è un albero, per ogni possibile D* (Non si può instradare in modo indipendente ogni coppia sorgente-destinazione).

Calcolo dei cammini minimi: calcolato su un grafo (ad archi pesati opportunamente). Questo permette la ricerca dell'albero dei cammini minimi, trovato grazie ad algoritmi che funzionano anche in modo distribuito sui nodi di rete. Inoltre la ricerca dei cammini minimi ha complessità polinomiale sul numero di nodi.

Dato un grafo G(N,A) e due nodi (i,j) trovare il cammino di lunghezza minima tra tutti quelli che permettono di andare da i a j. Si ha una proprietà: *se k è attraversato dal cammino minimo tra i e j, allora il sotto-cammino fino a k è anch'esso minimo*.

6.2 Calcolo dei cammini minimi

Algoritmo di Bellman-Ford: (pesi positivi e negativi, no cicli con lunghezza negativa) si trovano i

cammini minimi tra sorgente e tutti gli altri nodi, oppure da tutti i nodi ad una destinazione. Vedere slide 06.16 per l'algoritmo. Esso converge in un numero finito di passi anche in forma distribuita. Ogni nodo invia periodicamente l'ultima stima del cammino minimo ai vicini, aggiornando la propria stima secondo il criterio di iterazione.

Nella pratica: ogni nodo ha 2 etichette (n,L)

- **n** è il primo nodo sul cammino minimo
- **L** è la sua lunghezza

Si aggiornano le etichette guardando quelle dei vicini. Quando le etichette sono stabili, si ricostruisce l'albero dei cammini minimi ripercorrendole.

Algoritmo di Dijkstra: (pesi positivi) si trovano i cammini tra un nodo sorgente e tutti gli altri nodi. Vedere slides 06.20 per l'algoritmo. Nella pratica si applica lo stesso criterio di Bellman-Ford, distinguendo tra etichette temporanee e permanenti. Ad ogni iterazione, l'etichetta temporanea con la lunghezza minore diventa permanente.

Le complessità sono pari a $O(N^3)$ per Bellman Ford, e $O(N^2)$ per Dijkstra.

6.3 Protocolli di Routing IP

Identificano due funzionalità diverse: *scambio di informazioni di raggiungibilità tra i router* e *costruzione delle tabelle di routing*. Le **Tabelle di routing** sono costituite da un elenco di *route*: ognuna comprende rete di destinazione, netmask, first-hop. La metrica è generale tra tutti gli apparati. Si può indicare solo il primo router del cammino grazie alla proprietà per cui anche i sotto cammini di un cammino minimo sono minimi.

Esistono due famiglie principali di protocolli: **Distance Vector** e **Link State**

6.3.1 Distance Vector

Distance Vector rappresenta l'informazione sulla raggiungibilità: [indirizzo,distanza], dove la distanza è una stima posseduta dal nodo. Il DV è inviato ai nodi adiacenti, e la stima delle distanze è fatta grazie a Bellman-Ford distribuito. L'invio del DV è periodico, oppure se il risultato di un *ricalcolo* differisce dal precedente. Si esegue il **ricalcolo** se si riceve un DV diverso da quello memorizzato precedentemente, o se cade/nasce una linea attiva cui è connesso.

Ricalcolo: $D_j' = \min_k [D_k + d_{kj}]$

Come funziona: ogni nodo è identificato da un indirizzo. Si inizializza la rete attivando tutti i nodi contemporaneamente.

Cold Start: ogni nodo ha informazioni iniziali permanenti (*conoscenze locali*: il suo indirizzo e a quali link è connesso direttamente). La tabella coniene una route verso se stesso a costo 0. Si propaga l'informazione agli altri router adiacenti, che allargano le loro conoscenze (aggiornando il costo e inserendo l'informazione nella tabella). Le informazioni vengono propagate agli altri router, finchè ognuno non conosce la topologia della rete (ovvero quando le tabelle non si aggiornano più)

Caduta di un link: i nodi agli estremi monitorano e riscontrano la caduta del link. Aggiornano le loro tabelle, assegnando costo infinito al link caduto, e inviano i nuovi DV ai vicini. Questo causa l'aggiornamento e la successiva stabilizzazione di tutte le tabelle di routing negli altri nodi.

Distance Vector è molto facile da implementare, ma ha problemi

- velocità di convergenza,

- *counting to infinity*: quando le cadute di link causano partizionamenti della rete, può accadere che gli aggiornamenti dei DV non facciano convergere a stati stabili all'interno delle partizioni. Risolvibile introducendo **hop count limit** (rappresentando l'infinito con un valore finito, maggiore del più lungo percorso di rete. Se si raggiunge questo valore, il nodo è dichiarato irraggiungibile). Durante il *counting to infinity* la rete si trova in uno stato congestionato con perdita di pacchetti (e anche dei DV). Altro rimedio è **split-horizon**: se A manda a D i pacchetti destinati a X, non ha senso che A annunci a D la raggiungibilità di X nel suo DV. Il nodo A non annuncia quindi a D con quale costo raggiunge X. In forma semplice: il nodo omette ogni informazione sulle destinazioni che raggiunge tramite quel link. Con poisonous reverse: il nodo include nel messaggio tutte le destinazioni, ma pone a distanza infinita quelle raggiungibili tramite quel link. Split-Horizzon non funziona con certe topologie
- limitato dal nodo più lento
- i cambiamenti inducono cicli che sussistono anche per lungo tempo
- difficile da stabilizzare su grandi reti

Il problema di *counting to infinity* si può ripresentare anche in caso di Split-Horizon. Un ulteriore metodo di soluzione è rappresentato dall'utilizzo dei contatori (*Hold Down*): dopo un tempo $T_{invalid}$ in cui non ricevo DV di una route dal nodo del primo hop, la si dichiara invalida (*no announce, invalid DV*). Dopo un tempo T_{flush} la route è rimossa. Si calibra l'intervallo tra i due tempi in modo da permettere la propagazione dell'informazione relativa al guasto sulla rete.

Le route non valide sono annunciate con distanza infinita (valore di soglia), i nodi riceventi tale informazione mettono la route in hold-down. I cambiamenti di topologia sono annunciati immediatamente e distinti dagli altri (*Triggered Update*). Aumenta la velocità di convergenza e si scoprono prima i guasti.

6.3.2 Link State

Ogni nodo impara topologia, nodi adiacenti e distanze verso di esse. Si inviano queste informazioni in *flooding* con pacchetti LSP (*Link State Packet*). Grazie a questi pacchetti si costruiscono database LSP locali ai nodi con topologia completa della rete. Vengono quindi calcolati i cammini minimi verso tutte le destinazioni.

Link State è flessibile in quanto si ha un routing ottimale (ogni nodo conosce la rete), e l'informazione LSP può essere inviata solo dopo un cambiamento nella rete. Tutti i nodi vengono informati subito dei cambiamenti (particolarmente quelli topologici). Tuttavia:

- E'necessario un protocollo dedicato (Hello)
- E'necessario il *flooding*
- E'necessario riscontro sui pacchetti LSP
- E'difficile da implementare

Flooding: si ritrasmette il pacchetto su tutte le porte diversa da quella di ricezione. Si prevengono i loop e la congestione con appositi *Sequence Number* e database di SN ricevuti in ogni nodo, evitando di ritrasmettere più di una volta. Esiste anche un meccanismo simile a TTL di IP.

Quando arriva un pacchetto LSP, se non è mai stato ricevuto o SN > SN precedente, allora memorizzo LSP e lo ritrasmetto in flooding; se SN è uguale a quello memorizzato, non faccio nulla; se LSP è più vecchio di quello memorizzato, viene ritrasmesso LSP più recente al mittente.

Al variare dell'archivio degli LSP si verifica se varia anche il grafo pesato: in tal caso si ricalcola

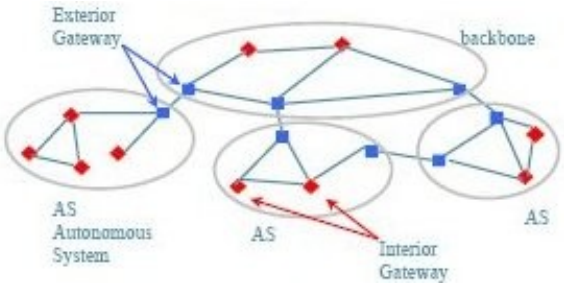
l'albero dei cammini minimi, inserendo nella tabella di routing per ogni possibile destinazione il primo hop [dest, nodo_vicino]

Nelle reti broadcast ogni nodo è logicamente connesso a tutti gli altri. In questo caso si elegge uno *pseudo-nodo* cui tutti gli altri sono connessi (la topologia diventa star invece di mesh).

6.4 Routing Internet

Vengono fatte le seguenti definizioni:

- **AS (Autonomous System):** porzione di rete gestita da diverse autorità
- **Protocollo di Routing:** protocollo di scambio informazioni di routing tra gateway
- **EGP (Exterior Gateway Protocol) IGP (Interior Gateway Protocol)**



Dominio di Routing (RD): è una porzione di AS in cui è implementato un unico protocollo di routing. Se i RD comunicano, alcuni router appartengono a più domini di routing e devono implementare più protocolli di routing. I router su più domini possono *ridistribuire* le informazioni di un dominio nell'altro e viceversa.

6.4.1 Protocolli usati (Distance Vector, Link State)

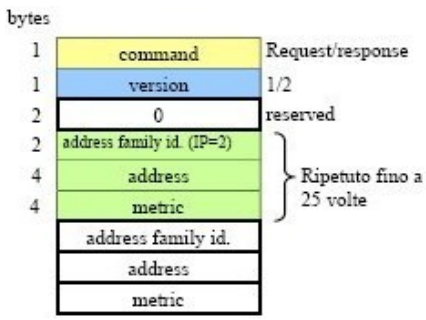
IGP

- **RIP (Routing Information Protocol) v1,v2**
- **IGRP (Interior Gateway Routing Protocol) CISCO**
- **IS-IS (Intermediate System – Intermediate System)**
- **OSPF (Open Shortest Path First)**

EGP

- **BGP (Border Gateway Protocol)**

RIP: protocollo IGP di tipo Distance Vector (RFC 1058). La sua metrica è il numero di HOP. Diffuso e semplice. **Routing Update Timer**: intervallo di tempo per l'invio dei DV (default 30s).



Route invalid timer: intervallo di dichiarazione di invalidità di una route in assenza di annunci dalla sua interfaccia (default 90s)

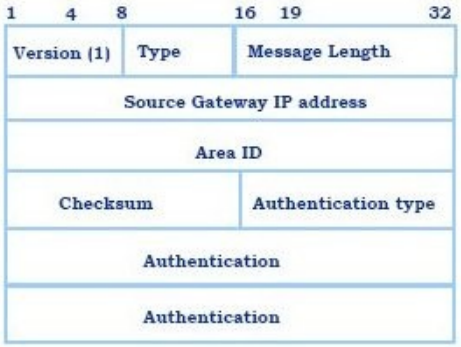
Route flush timer: intervallo di rimozione di una route invalida dalla tabella.

Versione 2 (RFC 1723): aggiunte le netmask

OSPF: protocollo di tipo Link State (RFC 1247,1583). Supporto di routing gerarchico, utilizzo di protocollo Hello, ID unico per i router, LSA (Link State Advertisement). Router classificati come **AS boundary router**, **area border router**, **backbone router** e **internal router.** In particolare gli *area border* diffondono in ogni area un riassunto delle informazioni raccolte nell'altra.

Tipi di LSA:

1. **Router Links advertisement**: all'interno della stessa area (LSP classico)
2. **Network links advertisement**: generato dallo pseudo nodo (LR) di una LAN
3. **Network summary link advertisement**: generato da ABR per riassumere info
4. **Boundary routers summary link advertisement**: generato da ABR, indica la presenza di un AS boundary router nell'area e il suo costo
5. **AS external link advertisement**: generato da un AS boundary router, propagato a tutti i router di tutte le aree (destinazioni esterne e costi)



- Il campo *Type* specifica il tipo di messaggio (HELLO, DB DESCRIPTION, LINK STATUS REQUEST, LS UPDATE, LS ACKNOWLEDGE)
- Il campo *source gateway IP address* è l'indirizzo IP del mittente e l'*Area ID* codifica l'area di appartenenza
- Il campo *Authentication type* può essere 0 o 1

OSPF invia periodicamente messaggi di HELLO per verificare la raggiungibilità dei vicini. I messaggi DB Desc servono per inizializzare il db topologico dei gateway. I dati di metrica sono passati sui messaggi LS.

BGP: protocollo EGP più diffuso. Routing di AS è diverso dal routing interno perchè:

1. I criteri di scelta sono difficilmente traducibili in metriche per il calcolo dei cammini
2. Gestori di AS instradano secondo una propria politica
3. Scelta basata sulla conoscenza dell'intero percorso verso la destinazione

DV non va bene (non ho conoscenza del percorso), LS non va bene (occorrerebbero info topologiche sull'intera rete mondiale). Si usa **Path Vector**: nei DV inviati dai nodi non è indicata la distanza da una destinazione, ma l'intero percorso verso la destinazione (sequenza di attributi). Vi sono attributi obbligatori (interpretati da tutte le implementazioni BGP - *Origin*: protocollo IGP da cui proviene l'informazione; *AS_PATH*: sequenza di AS attraversati; *NEXT_HOP*: prossimo router)

6.6 Multicasting

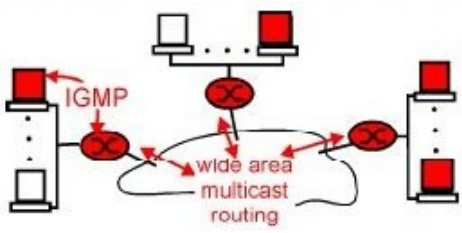
Certe applicazioni richiedono l'uso di collegamenti punto-multipunto. Può essere implementato anche da una sorgente su rete unicast (replicando i pacchetti per ogni sorgente). Se la rete supporta multicast, posso inviare un solo pacchetto ed essere supportato da nodi con ruolo attivo (router)

Si richiede: definizione dei destinatari, indirizzamento, definizione dell'albero di routing

Non è efficente identificare un gruppo di ricevitore tramite i loro indirizzi IP. IP definisce una classe di indirizzi per il multicast (11110...multicast addressess – da 224.0.0.0 a 239.255.255.255)

L'uso di questi ID riduce overhead ma pone problemi: come istituire un gruppo, come aggiungere membri, come controllare l'ingresso nel gruppo, chi conosce l'elenco dei membri.

IGMP (Internet Group Management Protocol): vi sono specifici router in rete che gestiscono il servizio multicast. IGMP usato nel colloqui tra host e multicast routers. Periodicamente il router IGMP manda messaggi broadcast (224.0.0.1); gli host rispondono con l'elenco dei gruppi in uso da qualche processo applicativo (**membership queries: general / specific ; membership report ; leave group)**



Multicast routing: i protocolli di multicast routing costruiscono un albero di instradamenti (Spanning Tree) che consente l'inoltro dei pacchetti senza effettuare cicli. I router senza utenti nel gruppo possono essere esclusi dall'albero. Si può costruire un albero che distribuisce pacchetti di tutte le sorgenti (**Group Shared Tree**) oppure un albero differente per ognuna delle sorgenti attive (**Source based trees**)

GST: la ricerca dell'albero di costo minimo è un problema NP-Completo. Si cercano soluzioni sub ottime (*center based approach*). Si elegge un router centrale, si inviano messaggi di *join* in unicast al router centrale. I messaggi tracciano i rami finchè non raggiungono il centro o un router già associato all'albero.

SBT: basata sull'albero (unicast) dei cammini minimi. Si usa però l'albero dei cammini al contrario. *Reverse Path Forwarding*: i pacchetti in arrivo sul cammino minimo verso la sorgente sono inoltrati, tutti gli altri sono scartati. I pacchetti raggiungono anche nodi senza host associati: questo si risolve con la tecnica del *pruning*. I router che non hanno host associati infatti possono inviare messaggi di **prune** al contrario lungo l'albero. Bisogna però segnalare ai router a valle l'evento multicast, e permettere il join di nuovi utenti (msg. Unprune o timeout sul prune)

DVMRP (Distance Vector Multicast Routing Protocol): è il più usato RPF. Utilizza DV per costruire l'albero dei cammini minimi. Ogni router ha una lista di router dipendenti. Un msg di pruning è inviato solo se tutti i router nella lista l'hanno fatto. Messaggio esplicito di unprune (*graft*), e timeout su info di prune.

Siccome in internet solo una piccola parte dei router sono multicast router, si può usare il tunnelling tra router multicast isolati (usata in Mbone)

7 Evoluzione network layer: MPLS e IPv6

7.1 MPLS

Acronimo di Multi-Protocol Label Switching. Unire i vantaggi di IP e ATM nelle *Backbone Network* (routing IP e switching ATM), eliminandone al contempo i difetti. Architettura di gestione flussi, a circuito virtuale (FEC – *Forward Equivalence Class*) predeterminati dal gestore on-demand, con meccanismi di setup e prenotazione.

Ottimizzazione dei flussi statica o dinamica, instradando su un ricco set di parametri (sorgenti, porte, applicazioni)

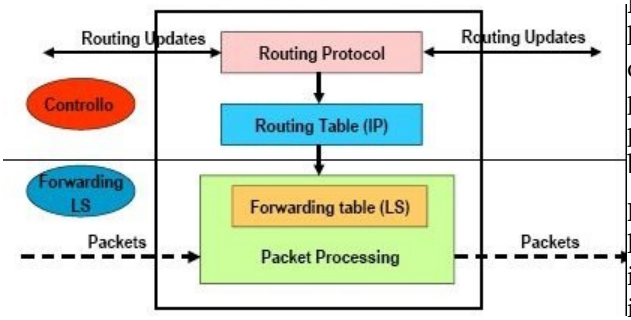
LS Forwarding: IP è incapsulato in un header LS

[	**Label**	**][CoS][S][**	**TTL**	**]** 4 Byte
	20 bit	3b 1b	8b	

CoS: Class of Service. **S**: Stack. **TTL**: Time To Live. Label a 20 bit è compatibile con VC ATM

La **Label** è usata per la commutazione (Label Swapping – Commutazione), ha significato Locale come in ATM e FR: esse vengono determinate e aggiornate al momento del set-up del cammino. All'*Ingress Router* vi è corrispondenza tra IP destinazione (e altri parametri opzionali) e la Label del cammino scelto. Se vi sono flussi con cammini in comune, il router di ingresso al cammino comune incapsula i due flussi con identica label, che vengono poi frammentati dal router dove si abbandona il cammino comune. Non c'è limite all'incapsulamento, l'instradamento avviene in base alla label più esterna e in base a indicazioni di *push* e *pop*. In questo modo si realizza una forte *scalabilità*.

Disaccoppiamento tra routing e forwarding:



I pacchetti di controllo seguono un inoltro hop-by-hop simile a IP tradizionale. Essi creano un nuovo *label switched path* (VC). I pacchetti per i quali è stato creato il path possono essere poi inoltrati direttamente in base alla label.

E'possibile anche instaurazione manuale del label switched path, in questo caso si omette il controllo. Il disaccoppiamentto permette di introdurre economie di scala, la prenotazione delle risorse e l'uso di tecniche di ingegneria del traffico.

Si introduce un database di traffic engineering (TED) e procedure avanzate di signaling. **TED** contiene informazioni topologiche (link-state, derivate dai routing protocol), informazioni sulle risorse di rete (derivate da estensioni dei routing protocol) e dati amministrativi (derivate dai dati di configurazione utenti). Consente ai *border router* di determinare un cammino.

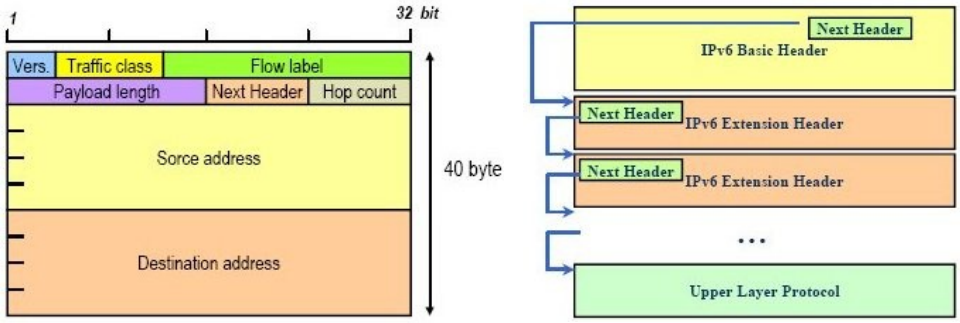
Il cammino viene instaurato determinando l'*Egress Router* in base al next-hop IBGP. Possono essere determinati i cammini anche con calcoli offline (permette ottimizzazioni globali a flussi noti) oppure online (tenendo conto dei vincoli utente). Viene usato un meccanismo di segnalazione per coordinare la distribuzione dei label, instaurare un cammino desiderato (*Explicit Route*), riservare e riassegnare le risorse e prevenire i loop.

- **LDP (Label Distribution Protocol)**: hop-by-hop, segue cammini IGP, non supporta TED
- **RSVP**: estensione per route esplicite. La sorgente invia un messaggio di PATH verso la destinazione, la destinazione accetta la richiesta e invia un messaggio di RESV che distribuisce la label e riserva le risorse.
- **CRLDP (Constrained Routing LDP)**: estende LDP per route esplicite

7.2 IPv6

Ipv6 è la nuova versione di IP. Mantiene l'impostazione di IPv4 ma cambia molti aspetti, aumentando nel contempo la lunghezza degli indirizzi da 32 a 128 bit.

- Gestione opzioni, frammentazione, identificzione, flussi, classi di traffico, no checksum
- ICMPv6: funzionalità aggiuntive rispetto a versione precedentemente, include ARP che è stato eliminato, include alcune funzioni di DHCP
- DHCPv6: modificato e aggiornato
- RIPng / OSPFv6: protocolli di routing



- **Version**: versione del protocollo (6)
- **Traffic Class**: distinguere più tipi di traffico nelle reti *Differentiated Services*
- **Flow Label**: Identifica un flusso di pacchetti (come MPLS)
- **Payload Length**: Lunghezza del pacchetto (eccetto basic header)
- **Next Header**: ID del tipo di header che segue il basic header (livello superiore o extension header)
- **Hop Limit**: come TTL
- **Source Address**: indirizzo sorgente
- **Destination Address**: indirizzo destinazione

IPv6 Extension Header:

1. **Hop-by-Hop option**: interpretato dai router, opzioni per pacchetti lunghi e gestione di allineamenti a 32 bit
2. **Source Routing**: obbliga i pacchetti a seguire un particolare percorso
3. **Fragmentation**: implementa la frammentazione, esegita solo dal mittente che conosce max MTU del path (tramite i messaggi MTU Path discovery di ICMPv6)
4. **Autenticazione**: del mittente
5. **Encrypted security payload**: serve per crittare il payload (pacchetto IP o livelli superiori)

7.2.1 Indirizzi IPv6

Ci sono $7x10^{23}$ indirizzi IPv6 per metro quadrato di superficie terrestre!

A gruppi di 2 byte in hex, si possono omettere gli zeri.

8000:0000:0000:0000:8965:0678:A45C:87D3 --> 8000::8965:678:A45C:87D3

Per IPv4 ho una notazione speciale ::131.175.21.173

Gli indirizzi possono essere di vari tipi, e normalmente un'interfaccia può avere più di un indirizzo associato. Destinatari *unicast*, *anycast, multicast* (uso globale o locale). I prefissi IPv6 servono come in IPv4 a individuare il campo che identifica l'interfaccia (stessa notazione). I tipi diversi di indirizzi sono identificati dalla prima parte del prefisso (*Format Prefix*)

Indirizzi speciali:

- Unspecified (0:0:0:0:0:0:0:0) è usato come indirizzo sorgente quando il nodo non conosce altri suoi indirizzi (non può essere usato come destinazione)
- Loopback (0:0:0:0:0:0:0:1) analogo al 127.x.y.z di IPv4
- IPv4-compatible IPv6 (::IPv4_addr) utilizzato per far comunicare host IPv6 quando occorre attraversare una rete IPv4
- IPv4-mapper IPv6 (::FFFF:IPv4_addr) utilizzato per far comunicare host IPv6 con host IPv4

Aggregatable Global Unicast Address: formato unicast globale. Struttura gerarchica per ridurre i problemi di scalabilità delle tabelle di routing. 3 Macrolivelli: Public Topology, Site Topology, Interface_ID

```
[001][  TLA  ][ Res ][    NLA    ][    SLA    ][         Interface ID         ]
 3b       13b     8b         24b          16b                   64b
```

1. **TLA (Top Level Aggregation)**: assegnato su base geografica o agli ISP backbone
2. **Res (Reserved)**: per future espansioni
3. **NLA (Next Level Aggregation)**: ogni ISP con un TLA può strutturare gerarchicamente le sue reti con diversi NLA
4. **SLA (Site Level Aggregation)**: sottoreti
5. **Interface ID**: formato derivato da IEEE EUI-64

I livelli NLA e SLA possono essere ulteriormente divisi in modo gerarchico.

Link-Local Unicast Address: identificati da FP = 1111 1110 10 Sono indirizzi utilizzabili per l'indirizzamento su un singolo link (sottorete). IpV6 prevede che ogni interfaccia disponga di almeno un link-local unicast address (assegnato per autoconfigurazione a partire dall'indirizzo fisico IEEE EUI-64). Fondamentali nel processo di *Neighbor Discovery*.

Site-Local Unicast Address: identificati da FP = 1111 1110 11 Destinati a uso locale, definiscono uno spazio di indirizzamento privato. (16 bit di *subnet*)

Multicast Address: identificati da FP 1111 1111. Diversi sottotipi (*Multicast global, Multicast link-local, Multicast site-local*). All'interno hanno indirizzi con usi speciali (4 bit di *flag*, 4 bit di *scope*). Se il bit T del flag [000T] è posto a 1 l'indirizzo è temporaneo, altrimenti è permanente. Scope ha vari valori (0: reserved; 1: node-local scope; 2: link local-scope; 5: site local-scope; 8: organization-local scope; E: global scope)

Indirizzi speciali multicast:

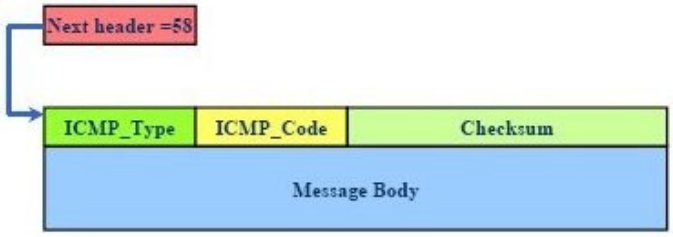
- FF01::1 >> *all systems node-local scope*
- FF02::1 >> *all systems link-local scope*
- FF01::2 >> *all routers node-local scope*
- FF02::2 >> *all routers link-local scope*
- FF05::2 >> *all routers site-local scope*

Solicited-Node Multicast Address: ogni sistema IPv6 deve avere uno di questi indirizzi per ogni indirizzo unicast/anycast configurato. E'costruito automaticamente concatenando il prefisso FF02::1::FF00:0/104 con gli ultimi 24 bit dell'indirizzo unicast/anycast.

IPv6 prevede l'uso di processi di autoconfigurazione. Un nodo

1. Si autoconfigura un link-local-address a partire dall'indirizzo fisico IEEE EUI-64
2. Si autoconfigura un solicited-node-multicast-address per ogni indirizzo
3. Può autoconfigurarsi altri indirizzi (ICMP , DHCP)

7.2.2 ICMP version 6



Molte più funzoni rispetto a v4

- Error reporting / diagnostica
- Risoluzione indirizzi link-level
- Individuazione router corretto
- Check IPv6 addressess

- Autoconfigurazione IPv6 addressess
- Calcolo PATH-MTU per la frammentazione

Tipi di messaggio ICMPv6 (più usati)

- Type 1: Destination Unreachable
- Type 2: Packet too big
- Type 3: Time exceeded
- Type 4: Parameter Probe
- Type 128: Echo Request
- Type 129: Echo Reply

Procedure di Neighbor Discovery: **Address Resolution** (come ARP di IPv4), **Router Discovery** (segnalare e scoprire presenza sul link), **Redirection** (opzione redirect di IPv4), **Neighbor Unreachability Detection** (irraggiungibilità di host noti)

Vengono utilizzati molti indirizzi speciali (*link-scope*)

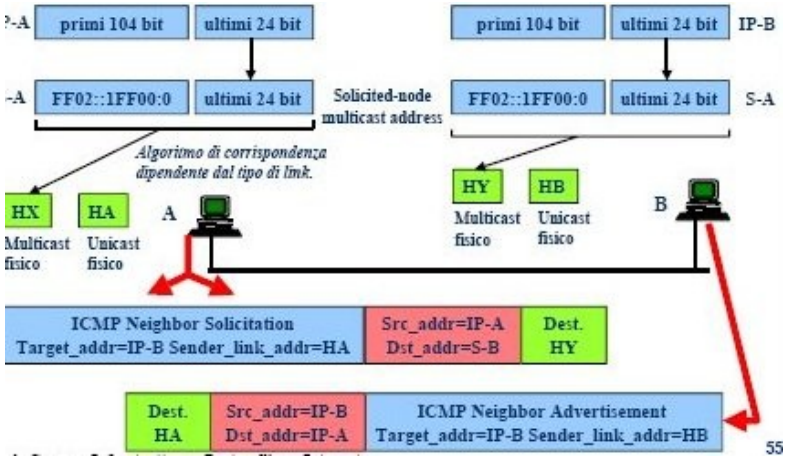
- All-systems Multicast Address (FF02::1)
- All-routers Multicast Address (FF02::1)
- Solicited-node Multicast Address
- Unicast Link-Local Address
- Unspecified Address (0::0)

Più 5 nuovi tipi di messaggio

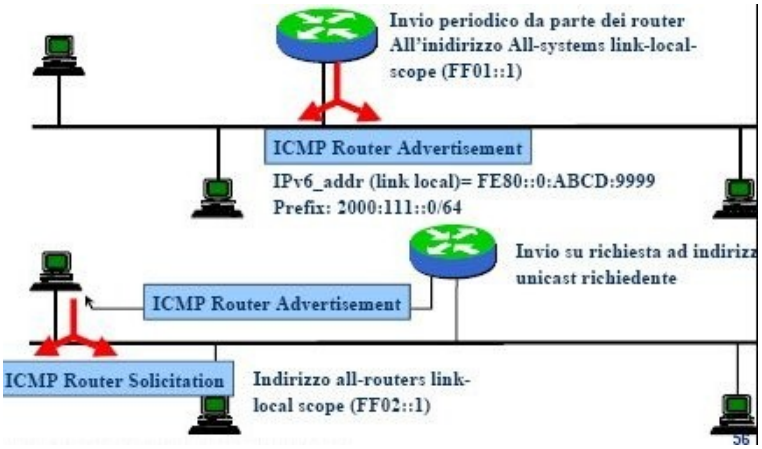
- Router Solicitation (type=133)
- Router Advertisement (type=134)
- Neighbor Solicitation (type=135)
- Neighbor Advertisement (type=136)
- Redirect Message (type=137)

ICMPv6 Address Resolution: come ARP, servono indirizzi multicast/broadcast sul livello inferiore. Esiste un mappaggio tra indirizzi multicast IPv6 e multicast/broadcast a livello link, e si usano *"Neighbor Solicitation"* e *"Neighbor Advertisement*"

Il messaggio *NS* viene inviato all'indirizzo *node-solicited multicast* che può essere ricavato anche dal richiedente. Il messaggio *NA* viene inviato all'indirizzo sorgente della richiesta



Address Resolution



Router Discovery

E'possibile autoconfigurare anche indirizzi globali.

- Stateful configuration: tramite DHCPv6
- Stateless configuration: tramite ICMP (noto il prefisso annunciato dal router)

MTU path discovery: mittente deve conoscere la MTU più piccola sul percorso. Si invia un pacchetto lungo quanto MTU del primo link, se arriva ICMP "Packet too big" riduco MTU e continuo finchè non arrivano più ICMP error.

La migrazione da sistemi IPv4 a sistemi IPv6 è basata sull'utilizzo di sistemi *dual stacked* (con implementazioni di IPv4 e IPv6 contemporaneamente), sull'utilizzo di *tunnel* (attraversamento di porzioni di rete IPv4) e su *header translation* (in entrambi i formati)

8 Intranet

Una intranet è una rete privata che utilizza tecnologia di interconnessione IP, normalmente collegata con la rete pubblica *internet* tramite un ISP e dotata di servizi per gli utenti internet. Sono sorte problematiche come:

- Sicurezza
- Gestione degli indirizzi
- Distinzione tra servizi offerti verso utenti intranet oppure accessibili anche da internet

Vi sono problemi anche negli indirizzi utilizzabili (per questo si spinge alla diffusione di IPv6), risolta su reti IPv4 con gli indirizzi privati. L'indirizzamento privato si basa su classi IP che non sono usate nella rete pubblica e i cui pacchetti (sorgente o destinazione) non possono transitare nella rete pubblica. **Classe A** (16 milioni di indirizzi)**, Classe B** (16 reti da 65536 indirizzi)**, Classe C** (256 reti). I server con accesso alla rete pubblica devono avere indirizzi pubblici, mentre gli altri possono avere indirizzo privato. L'esigenza di scambiare pacchetti tra host con indirizzo pubblico e indirizzo privato viene risolta tramite il **natting** ed i **proxy**.

8.1 NAT

Meccanismo disponibile su router/gateway. Consente di effettuare l'associazione tra indirizzi pubblici e indirizzi privati. I pacchetti verso l'esterno vengono riconosciuti e inviati al NAT, che provvede ad inoltrarli verso la corretta destinazione esterna modificando l'indirizzo sorgente del pacchetto e ritraducendo le risposte. Si mantiene l'associazione tra indirizzo privato e pubblico in una tabella di NAT (statica e dinamica). L'assegnamento dinamico si basa sulle sessioni (identificata dal socket per TCP e UDP, per ICMP dalla terna IP sorgente/destinazione e ID) essa viene instaurata creando un'associazione al passaggio del primo pacchetto, che viene rilasciata al termine della sessione.

Inzio sessione: pacchetto SYN per TCP, UDP/ICMP in vari metodi dato che sono connectionless

Fine Sessione: pacchetto FIN per TCP, metodi vari per altri pacchetti

Occorrono meccanismi di timeout per recovery errore e packet loss

ALG (Application Level Gateway): funzionalità aggiuntive, si preoccupano di modificare i messaggi applicativi in transito e/o adattare i segmenti TCP (per le applicazioni che trasportano indirizzi IP e porte all'interno del payload)

Traditional NAT (Outbound NAT): permette solo sezioni iniziate dall'interno. Le informazioni di routing possono solo entrare all'interno. Suddiviso in *Basic NAT* e *NAPT* (Network Address and Port Translator)

- **Basic NAT**: si traduce solo l'indirizzo IP. Corrispondenza un a uno nell'assegnamento degli indirizzi durante una sessione. Due host non possono usare contemporaneamente lo stesso indirizzo. Possibilità di blocco per traffico elevato.
- **NAPT**: viene tradotto il socket. Molti indirizzi interni possono usare lo stesso indirizzo esterno. Si hanno problemi con flussi diversi da UDP e TCP, e con pacchetti frammentati.

Bi-Directional NAT: è possibile aprire una sessione in entrambi i versi; l'host pubblico deve far riferimento a nomi simbolici ed il servizio DNS deve usare un'unico namespace.

L'utilizzo del *natting* comporta il ricalcolo dell'header checksum, la sostituzione degli indirizzi in messaggi ICMP, il ricalcolo dei checksum TCP e UDP con il nuovo pseudo-header. Problemi anche a carico degli ALG per quanto riguarda i messaggi applicativi.

Application Proxy: sono degli *application gateway*. Ogni richiesta viene inviata al Proxy che la inoltra con il proprio IP address pubblico. Occorre avere un proxy per ogni applicazione usata.

8.2 Connessione di intranet remote

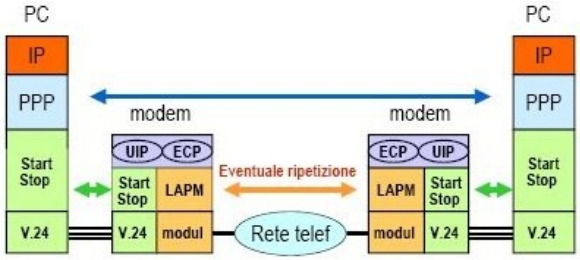
I problemi principali sono *costi*, *uso degli indirizzi privati, sicurezza* e *prestazioni*.

- Si possono usare canali dedicati ma i costi sono estremamente elevati
- Si possono usare reti a pacchetto pubbliche (es. *Frame Relay*), con costi altrettanto elevati
- Si può usare internet implementando una Virtual Private Network (VPN)

IP Tunnelling: si incapsulano trame IP in altre trame IP. Il payload che viaggia nel segmento pubblico può essere criptato, e gli indirizzi possono essere privati.

Accesso su linea dedicata punto-punto (dial-up domestico): occorre creare un collegamento di livello 2 su cui appoggiare IP. *Subscriber Loop*, connette l'utente alla centrale, accesso analogico (0-4 KHz) commutato. *Digital Subscriber Loop (DSL)*, sul doppino analogico si invia il segnale digitale ISDN.

Accesso con rete telefonica. Accesso dial-up o circuito permanente con ISP. IP è assegnato dal gestore



In caso di SOHO (Small Office – Home) occorre un router che si connette in presenza di traffico, incapsuli in PPP e faccia NAT-PAT

8.2.1 HDLC (High-Level Data Link Control)

Orientato al bit, opera in molti modi con vari meccanismi di controllo errore e flusso (half-duplex, full-duplex, master-slave, peer-to-peer).

Normal Response Mode (NRM): una stazione primaria è collegata a più stazioni secondarie (half-duplex). Solo la primaria puo inviare comandi, le secondarie trasmettono solo dopo un permesso esplicito inviato dalla primaria.

Asynchronous Response Mode (ARM): colloquio sbilanciato come in NRM, ma la stazione secondaria può iniziare trasmissione senza il permesso esplicito (full-duplex). Poco usata.

Asynchronous Balanced Mode (ABM): modalità di funzionamento bilanciato su configurazioni punto-punto. Ho stazioni indipendenti che comunicano full-duplex in modo asincrono (conforme allo stack OSI)

Trama HDLC: [F] [ADDRESS] [CONTROL] [INFO] [FCS] [F]

- Flag (F) pari a 01111110. Si utilizza il bit stuffing. Si può inviare continuamente il flag in caso di mancanza di informazioni.
- Address (A) 8 bit, può essere esteso a n byte. Ultimo bit di ogni byte indica se segue un ulteriore byte di A (0) oppure no (1). In modalità ABM è indirizzo destinazione, nelle altre è quello della stazione secondaria.
- Control (C) differisce a seconda del tipo di pacchetto (Informazione, Supervisione, Non

numerata). Campo lungo 2 byte.

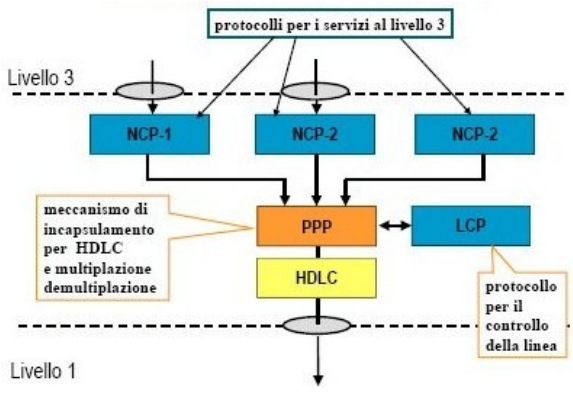
1. *Informazione*: trame numerate per la trasmissione dell'info utente ivi contenuta. Consentono riscontro tramite piggybacking, il polling (P a 1) e l'acknowledgement (F a 1)
2. *Supervisione*: trame numerate per il controllo dell'invio di flusso. (Comandi: RR, RNR, REJ, SRJ). **RR** è usato come ACK, in RN ho la prossima trama attesa, riscontro fino a RN-1. **RNR** blocca l'invio di ulteriori trame, riscontra fino a RN-1. **REJ** richiede la ritrasmissione delle trame da RN in avanti, riscontro fino a RN-1. **SREJ** richiede la ritrasmissione della trama RN.
3. *Non numerate*: informazioni di controllo relative all'instaurazione/chiusura della connessione oppure a invio di informazioni in modalità connectionless.

- Informazione (INFO): contiene informazione utente dei livelli superiori, opzionale, è di lunghezza variabile.
- Frame Check Sequence (FCS): codice di rivelazione d'errore

Sulle slides (08.49 – 08.53) si trovano esempi di comunicazioni nelle varie modalità

LAPM (Link Access Procedure for Modems): Derivato da HDLC e usato nei modem (V.32) per permettere la correzione d'errore. Connessione gestita in ABM – ABME.

PPP (Point to Point Protocol): nato in ambiente IETF per connessioni punto-punto su collegamenti senza errori e per consentire procedure d'ingresso in internet. E'un insieme di protoclli diversi a supporto dei protocolli di livello 3 (negoziazione indirizzi IP e autenticazione). Uso di incapsulamento *HDLC*. Aggiunta di header per multiplazione flussi superiori. HDLC è usato in modalità connectionless.



Generalmente la richiesta di connessione causa l'utilizzo di LCP, PAP e IPCP mediati da PPP e trasferiti su mezzo tramite HDLC. Alla fine, quando la connessione è stata instaurata, IP viene mediato da PPP e trasmesso tramite HDLC.

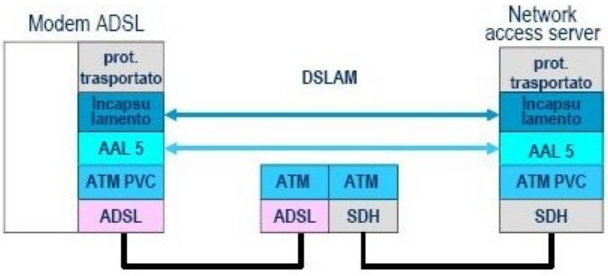
Dopo la connessione si hanno le seguenti possibilità: *c023* (Password Authentication Protocol); *c025* (Link Quality Report); *c223* (Challenge Handshake Authentication Protocol), per poi passare a NCP per protocolli di livello 3.

NCP - IP Control Protocol (IPCP): E'il protocollo NCP che gestisce trasporto e controllo di IP. Si stabilisce una connessione in cui si decide l'assegnazione dinamica di un indirizzo IP ed il tipo di compressione (0021 IP non compresso, 002D TCP/IP compresso, 002F TCP non compresso)

8.2.2 ADSL (Asymmetric Digital Subscriber Loop)

Tecnologia a banda larga per connessioni residenziali alla rete pubblica. I dati e la fonia sono multiplati a divisione di frequenza. Duplexing a divisione di frequenza asimmetrico sullo stesso doppino (Voce 0-4 Khz; Uplink 1 Mb/s 25-200 Khz, Downlink 8 Mb/s 240Khz – 2 Mhz).

Accesso su circuiti virtuali permanenti ATM tra l'utente (ATU-R: ADSL Termination Unit Remote) e l'ISP (DSLAM)



Il protocollo trasposrtato dipende dalla modalità di connessione. PPP è comodo per accedere all'ISP: ognuno può accedere indipendentemente ed ha IP dinamico assegnato dal provider.

Interfaccia modem di tipo LAN Ethernet 802.3 per consentire piccole LAN locali.

PPP è un protocollo lvl2 P2P, come lo convoglio su ethernet fino al modem? Lo incapsulo in un tunnel IP locale tra PC e modem, tramite un protocollo di incapsulamento. PPP-ADSL, PPP-Proxy: il modem ADSL assume funzionalità di router, utilizza PPP over ATM per connettersi al provider e per multiplare traffico IP. E'necessaria la funzione NAT (NAPT) e non ho limitazioni sull'uso della LAN locale.