LIVELLO APPLICAZIONE (LIV. 5)

Architettura Client-Server

Server: ha un indirizzo IP sempre attivo e fisso
Client: comunica con il server, e può contattarlo in qualsiasi momento; può avere indirizzi IP dinamici e non comunica direttamente con altri client

Architettura P2P (*pura*)

Non esiste un server sempre attivo, ma ci sono coppie di host (*peer*) che comunicano tra di loro. È un'architettura facile da scalare, ma difficile da gestire.

Architettura Ibrida

Skype è un'applicazione P2P di VOIP. Ha un server centralizzato e la connessione avviene da client-client. La chat è di tipo P2P.

Servizi Richiesti Dalle Applicazioni

Le applicazioni richiedono dal liv. di trasporto:

- **Perdita dei dati:** Alcune applicazioni possono tollerare la perdita, altre no.
- **Throughput:** Alcune applicazioni richiedono una banda minima, altre no.
- **Sicurezza:** Cifratura dei dati, integrità ecc..
- **Temporizzazzione:** Richiedono piccoli ritardi per rendere internet interattivo

HTTP (*Hyper Text Transfer Protocol*)

Si basa sul modello **client/server** : il client è un broser, il server è un *server web* che invia oggetti alla richiesta.

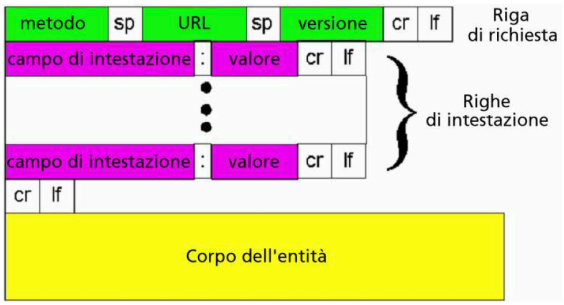
- ➔ Usa il protocollo **TCP**
- ➔ Lavora sulla **Porta 80**
- ➔ HTTP è un protocollo **stateless**

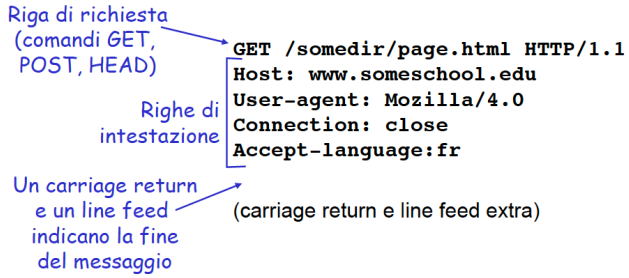
Connessione **Persistente**:
Il processo di richiesta dei file viene attraverso una singola connessione TCP (*maggiore RTT*)

Connessione **NON Persistente**:
Il processo di richiesta dei file viene attraverso una connessione TCP per ogni file (*minore RTT*)

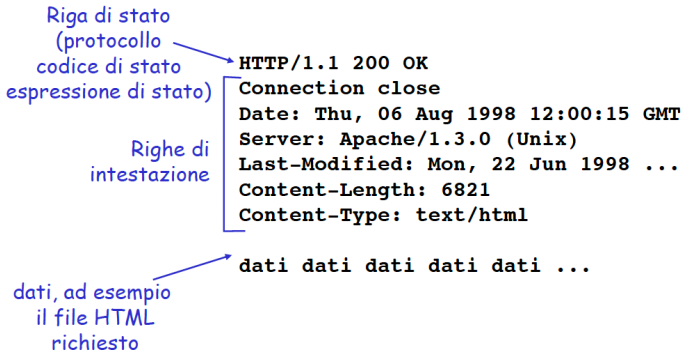
Formato Richiesta HTTP

HTTP/1.1 Permette di effetturare: **GET, POST, HEAD, PUT e DELETE**. Il formato di richiesta http è il seguente:





Il formato di risposta http è il seguente:



Cookie

I cookie sono composti da quattro componenti:

- ➔ Una riga di intestazione nel messaggio di **risposta (*COOKIE: 292)***
- ➔ Una riga di intestazione nel messaggio di **richiesta (*SET-COOKIE: 292)***
- ➔ Un **file** mantenuto **sul client**
- ➔ Un **database** sul server (*web server*)

I cookie possono contenere autorizzazioni, sessioni dell'utente, carte per acquisti ecc..

FTP (*File Transfer Protocol*)

FTP è un protocollo usato per il **trasferimento** dei file ha un host remoto. Si basa sul modello **client/server**

- ➔ Usa la **Porta 21** (*Connessione di controllo)*
- ➔ Usa la **Porta 20** (*Connessione Dati TCP*)
- ➔ Usa **TCP**

Posta Elettronica

La posta elettronica è composta da tre componenti:

- ➔ **Agente Utente:** detto anche *email reader* è il client di posta (i.e. gmail)
- ➔ **Server di Posta:** contiene la coda dei messaggi da trasmettere e utilizza il protocollo **SMTP** tra server per inviare i messaggi.
- ➔ **SMTP**

SMTP

SMTP è un protocollo per scambiare i messaggi di posta elettronica.

- ➔ Usa **TCP**
- ➔ Usa la **Porta 25**
- ➔ Trasferimento diretto (handshake, trasferimento, chiusura)
- ➔ I messaggi devono essere ASCII 7bit
- ➔ Usa Connessioni Persistenti

Protocolli di accesso alla posta (POP/IMAP)

SMTP si occupa di memorizzare e consegnare sul server del destinatario, ma per accedere e ottenere i messaggi dal server ci vuole un protocollo:

- **POP** *Post Office Protocol* : autorizza agente-server e si occupa del download
- **IMAP** *Internet Mail Access Protocol*: Manipola i messaggi sul server

DNS (*Domain Name System*)

E un database distribuito implementato in una gerarchia di server DNS. E' un protocollo a liv. applicazione che consente agli host, router e server di comunicare per risolvere i nomi (aka tradurre ip-nome)
SERVIZI:

- Traduzione host – ip
- Host aliasing (un host più avere più nomi)
- Mail server aliasing
- Distrubuzione locale

Il server **DNS Radice** viene contattato quando il DNS locale non trova il nome; il DNS radice ottiene la mappatura e la restituisce.

Il server **DNS Locale** non appartiene alla gerarchia, e ciascun ISP ha un suo DNS locale. Quando un host effettua una richiesta DNS viene prima inoltrata al DNS locale.

Record DNS

DNS: Database distribuito che memorizza i record di risorsa (*RR*)

RR : (name, value, type, ttl)

Type [A]
- Name = nome host
- Value = ind. IP

Type [NS]
- Name = dominio (i.e. foo.com)
- Value = nome host server competenza del dominio

Type [CNAME]
- Name = nome alias di qualche nome canonico (i.e. ibm.com è servercast.backup2.ibm.com)
- Value = nome canonico

Type [MX]
- Name = X
- Value = nome del server associate a name

P2P Puro - BitTorrent

Torrent: gruppo di peer che si scambiano file
Tracker: tiene traccia dei peer che partecipano

→ Il file viene diviso in parti, quando un peer entra nella rete, non lo possiede, ma lo accumula nel tempo.
→ Mentre effettua il download il peer carica le sue parti su altri peer

LIVELLO DI TRASPORTO (LIV. 4)

Un protocollo a livello di trasporto mette a disposizione una **comunicazione logica** tra processi applicativi di host differenti.
Per **comunicazione logica** si intende, dal punto di vista dell'applicazione, che tutto proceda come se gli host che eseguono i *processi* fossero direttamente connessi.

N.B: I protocolli di trasporto sono implementati nei sistemi periferici e non nei nodi della rete (i.e. router)

Internet mette a disposizione due tipi di protocolli diversi tra di loro: **TCP** e **UDP**.
Il passaggio host-to-host viene detto **multiplexing e demultiplexing**.

Il livello di trasporto mette a disposizione funzioni di creazione, gestione e rilascio delle connessioni, recupero degli errori, consegna ordinata dei segmenti controllo di flusso e congestione.

TCP

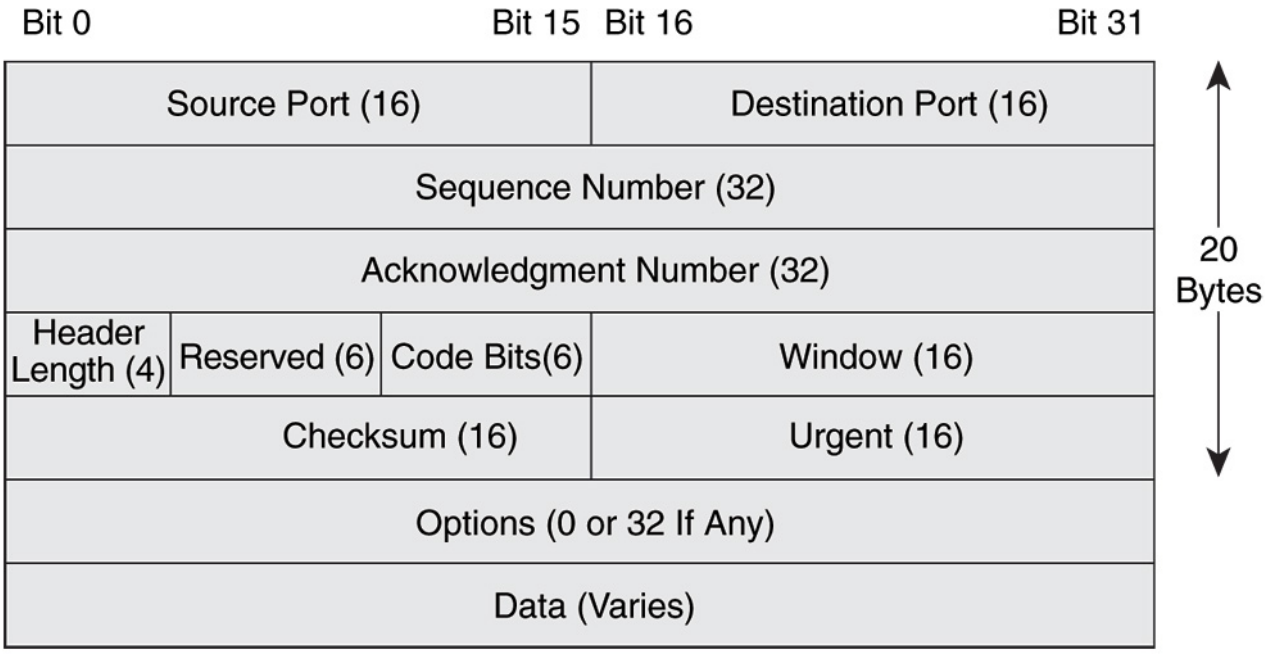
-TCP è **orientato alla connessione** in quanto, prima di effettuare lo scambio dei dati, i processi devono effettuare un handshake, ossia scambiarsi i parametri per il trasferimento.

-TCP offre un servizio **full-duplex**: su una connessione TCP tra il processo A e il processo B su un altro host, i dati possono fluire in entrambe le direzioni.

-Una connessione TCP è **punto-punto** ossia ha luogo con un solo mittente e un solo destinatario. Il multicast con TCP **non** è possibile

- Il TCP è un protocollo di trasporto connection oriented affidabile e svolge le seguenti funzioni:
  - indirizzamento a livello applicativo / multiplazione / demultiplazione
    - utilizza il concetto di porta per indirizzare le diverse applicazioni;
  - instaurazione, gestione e rilascio delle connessioni
    - utilizza il concetto di socket per identificare una connessione;
    - si preoccupa di gestire la connessione, ovvero lo scambio di informazioni necessarie per concordare l'attivazione di un canale di comunicazione (INS, MSS, Window, ...)
  - recupero degli errori
    - gestisce il recupero dei segmenti errati o persi utilizzando un timeout di ritrasmissione; RTT e RTO vengono calcolati dinamicamente
    - una volta recuperati gli errori, il TCP può effettuare la consegna ordinata dei segmenti all'applicazione
  - Controllo di flusso
    - implementa un controllo di flusso a finestra scorrevole (con dimensione della finestra variabile) come controllo del tasso di immissione di segmenti in rete
  - Controllo della congestione
    - reagisce alla congestione (scadere del RTO) diminuendo l'ampiezza della finestra di trasmissione secondo algoritmi noti

Header TCP



- **Sequence number**: il campo identifica il numero di sequenza del primo byte del payload
- **ACK number:** il campo identifica il numero di sequenza del prossimo byte che si intende ricevere (ha validità se il segmento è un ACK)
- **Offset:** lunghezza header TCP
- **Reserved:** riservato per usi futuri
- **Window:** ampiezza della finestra di ricezione
- **Urgent pointer:** il campo identifica che il ricevente deve iniziare a leggere il campo dati a partire dal byte specificato. (usato per eventi asincroni "urgenti")
- **Options:** riempimento e campi opzionali come setup per comunicare il MSS

Instaurazione connessione TCP

TCP è **connection-oriented**, l'instaurazione della connessione avviene secondo la procedura ***three-way-handshake*** :

1) TCPc invia un segmento al TCPs, questo segmento non contiene dati a livello applicativo, ma con il bit **SYN = 1**.
   [SYN INIZIALE]

2) Quando il TCP SYN arriva all'host server, esso, alloca le variabili TCP e invia un segmento di connessione approvata al TCPc con il bit **SYN = 1**, e il campo **ACK = client_isn + 1**.
   [SYNACK]

3) Alla ricezione del SYNACK, anche il client alloca il buffer e le variabili di connessione. Dopodichè il TCPc invia al server un altro segmento di risposta di connessione approvata, con **ACK = server_isn +1**. Il bit di SYN in questo caso è posto a **0** perché la connessione è stata stabilita. Il payload può essere pieno.
   [OKSYN]

Terminazione connessione TCP

1) Il TCPc invia un segmento con il bit **FIN = 1**.
2) Quando il TCPs riceve questo messaggio risponde con un ACK.
3) Il server spedisce quindi il proprio segmento di shutdown con il bit FIN = 1.
4) Infine il client invia un ACK al server e deallocano le risorse.

Affidabilità TCP

TCP garantisce la corretta e ordinata consegna dei sementi. TCP è in grado di gestire le situazioni di errore (tramite il campo checksum) e la perdita dei segmenti. In tal caso si occupa di ritrasmettere i segmenti, con la tecnica *"positive ack with retransmission"*: nella versione base (stop&wait) la sorgente non trasmette un nuovo segmento fino a quando non viene riscontrato.

Gestione Timeout TCP

L'**RTO** indica il tempo entro il quale la sorgente si aspetta di ricevere l'ACK. L'RTO non è un valore statico, perché dipende dalla distanza, dalla condizione della rete: viene calcolato ogni volta. Il calcolo dell'RTO si basa sulla misura dell'RTT.
L'**RTT** è l'intervallo di tempo tra l'invio di un segmento e la ricezione del riscontro di quel segmento. Esso include i ritardi di propagazione, accodamento e di elaborazione. Il suo valore (chiamato SRTT) è dato da:

$$\textbf{\textit{SRTTattuale}} = (a * SRTTprecedente) * ((1-a)*RTT\ istantaneo))$$

Se il valore a è vicino a 1 vuol dire che è stabile, se è vicino a 0 è dipende fortemente alla misura degli RTT istantanei.

Il valore dell'RTO è quindi dato da:

$$\textbf{\textit{RTO}} = b * SRTT$$

Il valore b è il ritardo, solitamente impostato a 2. La sorgente dunque attende fino a due volte il SRTT proima di considerare il segmento perso e ritrasmetterlo.

Controllo di flusso TCP

Serve a prevenire la congestione del buffer a lato ricevente. E' possibile facendo mantenere al mittente una variabile chiamata finestra di ricezione, che fornisce al mittente una indicazione dello spazio disponibile nel buffer del destinatario.
Per incrementare l'efficienza è possibile trasmettere + segmenti consecutivamente senza attendere ogni ack. Tale tecnica è chiamata *finestra scorrevole*

- ➔ Se è troppo piccola viene sottoutilizzata la banda
- ➔ Se è troppo grande gli ack potrebbero arrivare prima della trasmissione della finestra

Se il segmento perso appartiene a una finestra di trasmissione è possibile far tornare indietro la finestra (G-B-N) o ritrasmettere solo il segmento (S-R)

Controllo di congestione TCP

In caso di congestione della rete, alcuni segmenti possono essere persi. La perdita dei segmenti e lo scadere del RTO è considerato un sintomo di congestione (TCP rileva solo da questo).

In caso di congestione, il controllo di flusso a finestra, protegge anche la rete:

- ➔ Se è congestionata arrivano meno ack e quindi il tasso di immissione nel buffer è minore
- ➔ L'attesa dello scadere del timeout lascia libera la banda

La dimensione della finestra varia dinamicamente, e viene chiamata **Congestion Window CWND**. Essa è regolata da due algoritmi: Slow Start e Congestion Avoidance. Altri agoritmi sono stati usati per aumentare l'efficienza di TCP: Fast Retransmit e Recovery

Slow Start e Congestion Avoidance TCP

	Aumento	Andamento
CONGESTION AVOIDANCE	Aumenta di 1 ad ogni RTT	Andamento lineare
SLOW START	Aumenta del doppio di quelli trasmessi.	Andamento esponenziale

Tecnica GBN CWND TCP

Inizio Trasmissione:

- CWND = 1 (ovvero numero di byte pari a MSS)
- SSHTHRESH = RCVWND oppure SSTHRESH = RCWIND / S

Nella Trasmissione:

- La CWND evolve secondo l'algoritmo di slow start fino al raggiungimento della SSTHRESH, dopodichè è regolata dall'algoritmo Congestion Avoidance
- La finestra cresce fino al raggiungimento di RCWND

Errori CWND TCP

In caso di perdita o errore (***singolo-ERR-Trasmissione***):

- La trasmissione si interrompe
- Si attende lo scadere dell'RTO
- Al suo scadere si pone:
  SSTHRESH = CWND / s
  CWND = 1
- Si riprende a trasmettere con la tecnica GBN

In caso di perdita o errore (***consecutiva-ERR-Congestione***):

- Al primo errore, quando l'RTO scade si ritrasmette il primo segmento non riscontrato, il timeout per quel segmento viene raddoppiato.
  RTOnew = 2* RTO
- La CWND rimane a 1, mentre SSTHRESH = 2

Fast Retransmit & Fast Recovery TCP

Esistono due motivi che causano la perdita dei segmenti

1) **Errori di trasmissione**: influisce generalmente su un singolo segmento
2) **Errori di congestione:** provoca la perdita di più segmenti consecutivi

Fast retransmit: il segmento viene subito ritrasmetto
Fast Recovery: la cwnd non viene chiusa eccessivamente

LIVELLO DI RETE (LIV. 3)

Il ruolo del livello di rete è quindi semplice: trasferire pacchetti da un host all'altro. Per fare questo è possibile identificare due importanti funzioni:

• **INOLTRO:** (*forwarding*) Quando un router riceve un pacchetto lo deve spedire sull'appropriato collegamento di uscita (un router ha più interfacce). Un pacchetto può essere eliminato, duplicato o inviato su più interfacce.

➔ L'inoltro viene su una scala temporale molto piccola, ed è usualmente implementato in hardware

• **INSTRADAMENTO:** *(routing)* Il livello di rete deve determinare il percorso che i pacchetti devono seguire tramite gli **algoritmi di instradamento** (algoritmi di routing)

➔ Avviene su scale temporali più grandi, ed è usualmente implementato in software

Servizi Offerti dal LIV. RETE

Il livello di rete potrebbe offrire:

➔ **Consegna Garantita:** Questo servizio assicura che il pacchetto giunga, prima o poi, alla propria destinazione
➔ **Consegna Garantita con Ritardo Limitato:** Garantisce la consegna del pacchetto, ma anche il rispetto di un limite di ritardo (i.e. 100_{ms})
➔ **Consegna Ordinata:** Garantisce che i pacchetti giungano alla destinazione nell'ordine in cui sono stati inviati
➔ **Banda Minima Garantita:** Emula il comportamento di un collegamento trasmissivo con bit rate specifico tra host di invio e destinazione, anche se l'effettivo percorso end-to-end può attraversare diversi collegamenti fisici. Finchè l'host di invio trasmette a un tasso inferiore al bit rate specificato non si verifica perdita di pacchetti
➔ **Servizi di Sicurezza:** Il livello di rete dell'host sorgente può cifrare tutti i datagrammi inviati; il livello destinazione avrà il compito di decifrarli: in questo modo la riservatezza viene fornita a tutti i segmenti del liv. di trasporto.

Il **livello di rete** di Internet **offre** un solo servizio noto come servizio **best-effort**, ossia "col massimo impegno possibile". Questo modello, anche se non offre niente, combinato con un'adeguata larghezza di banda si è dimostrato abbastanza buono da supportare una grande quantità di datagrammi (i.e. streaming).

Interno del ROUTER

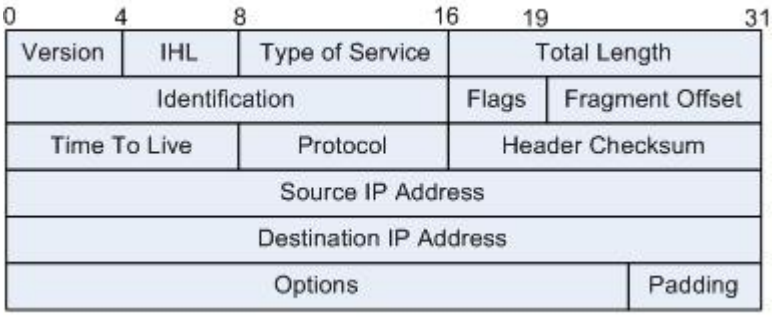
In una visione ad alto livello si possono identificare quattro componenti:

➔ **Porte di Ingresso:** Svolgono le funzioni a livello fisico di terminazione di un collegamento in ingresso al router; Svolgono anche funzioni a livello di collegamento necessarie per inter-operare con lle analoghe funzioni dall'altro capo del collegamento di ingresso; Svolgono anche la funzione di ricerca
➔ **Struttura di Commutazione:** Essa connette fisicamente le porte di ingresso a quelle di uscita: è contenuta all'interno del router.

➔ **Porte di Uscita:**
// todo, da finire

IPv4

Il pacchetto di rete è noto come *datagramma*. Il formato del datagramma IP è il seguente:



-Version [4]: Indicano la versione del protocollo IP nel datagramma, per permettere al router di interpretarlo al meglio.
-Header Lenght (IHL) [4]: Dato che un datagramma può contenere un numero variabile di opzioni, questi 4 bit indicano dove inizia effettivamente il payload. La maggior parte non contiene opzioni, pertanto il tipico datagramma IP ha un'intestazione di *20byte*.
-Type of Service [8]: I bit relativi al tipo di servizio servono per distinguere i diversi tipi di datagramma. Speso è utile distinguerli in tempo reale da altro traffico, usati per esempio per richiedere affidabilità, basso ritardpo, ecc..)
-Total Lenght [16]: Rappresenta la lunghezza totale (header+payload). La dimensione massima è di *65.535byte* anche se raramente superano i **1500** in modo da non superare la lunghezza massima dei *frame ethernet*.
-Identification [16]: Numero sequenziale assegnato al datagramma nel caso in cui fosse frammentato per ricomporlo.
-Flags [4]: Usato per indicare se il datagramma è stato frammentato e per indentificare l'ultimo frammento
-Frag. Offset [4]: Specifica l'offset del frammento rispetto al datagramma originale
-TTL [4]: E' stato inserito per assicurare che i datagrammi non restino in circolazione per sempre nella rete; ad ogni hop viene decrementato di 1 unità, quando raggiunge lo 0 viene scartato.
-Procol [4]: E' usato quando il datagramma raggiunge la destinazione finale. Specifica il protocollo a livello di trasporto al quale vengono passati i dati del datagramma. (Es. val 6 = TCP, val 17 UDP)
-Header Checksum [16]: Consente di rilevare gli errori nell'header del datagramma;
-Source / Destination IP [32|32]: Quando un host crea un datagramma inserisce il proprio IP nel campo origine e destinazione.
-Options / Padding [x]: Campi opzionali con informazioni addizionali. Se il campo opzioni è presente la sua dimensione non è un multiplo di 32bit, vengono messi degli 0 per raggiungere un multiplo di 32bit.

Indirizzamento IPv4

Gli indirizzi IP sono lunghi *32bit* e tali indirizzi sono scritti nella forma dotted-decimal notation (i.e. 192.168.1.1). Ogni interfaccia host e router di internet ha un indirizzo IP globalmente **univoco**, tuttavia i tali indirizzi non possono essere scelti in modo arbitrario: una parte dell'indirizzo server per indicare la sotterete a cui è collegato.

Gli indirizzi IP possono essere divisi concettualmente in due parti:

- **Prefisso:** identifica la rete fisica in cui l'host è connesso
- **Suffisso:** identifica un host specifico all'interno della rete

Lo schema degli indirizzi IP garantisce due proprietà

- Ogni host ha un indirizzo IP univoco
- L'assegnazione dei numeri di rete (prefisso) viene coordinata globalmente

CIDR / Subnetting in IPv4

La soluzione principale era quella di dividere gli indirizzi in classi (A,B,C,D,E), ma questo schema di tipo **classeful** venne sostituito dal tipo **classless**

La maschera indica quanti bit associati all'indirizzo appartengono al prefisso, e da dove parte poi l'indirizzo dell'host della sottorete. La notazione CIDR semplifica la gestione dalla parte degli utenti, indicata come ddd.ddd.ddd.ddd/m, dove **m** sono i bit del prefisso.

Indirizzi Speciali in IPv4

Il protocollo IP definisce un insieme di indirizzi speciali che sono riservati. Questi indirizzi non possono essere assegnati agli host:

- **Indirizzi di rete:** l'indirizzo con host address a 0 viene riservato (identifica la rete)
- **Indirizzi Direct Broadcast:** serve per semplificare il boradcasting (invio a tutti); esso è costituito mettendo tutti i bit a 1 dell'host address (il pacchetto viene consegnato a tutti gli host della rete specificata)
- **Indirizzi Limited Boradcast:** Boradcast limitato verso gli host connessi direttamente alla rete a cui è connesso l'host che invia il pacchetto; esso è costituito da tutti i 32 bit a 1
- **Indirizzi Localhost:** identifica lo stesso host; è composto da 32 bit a 0
- **Indirizzo di loopback:** usato per testare le applicazioni di rete; è un indirizzo che è valido solo sullo stesso host (i.e. 127.0.0.0)

DHCP

Gli indirizzi degli host possono essere configurati manualmente, ma più spesso questo compito è svolto usando il **dynamic host configuration protocol** (DHCP). Esso consente di ottenere un indirizzo ip in modo automatico, soi come di apprendere info aggiuntive come la maschera di sottorete, indirizzo del router per uscire dalla sottorete (def. Gateway) e l'indirizzo dns locale. *L'indirizzo IP assegnato può essere persistente oppure temporaneo.*

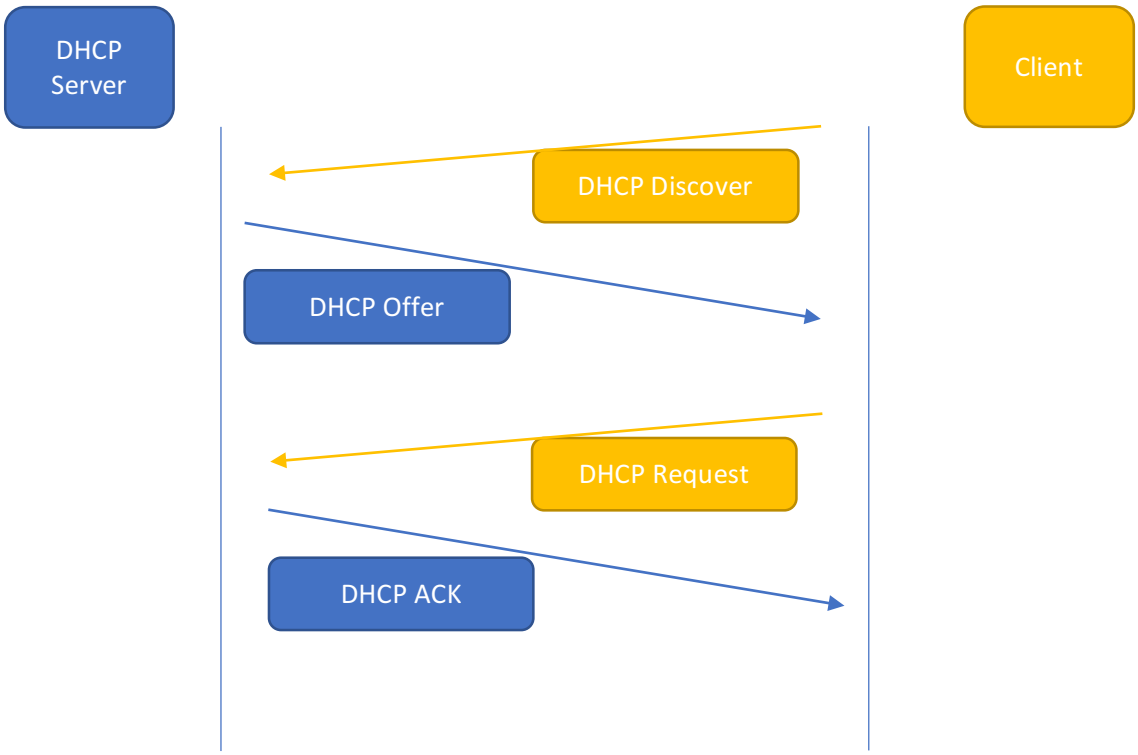
DHCP viene spesso detto protocollo **plug-and-play o zero-conf** per la sua capacità di automatizzare la connessione degli host con la rete.

DHCP è un protocollo **client-server**: ogni host è un client, che ha un DHCP per sottorete. Nel caso non ci sia ci sarà un DHCP Relay che conosce l'indirizzo del DHCP.

Per i nuovi host connessi il protocollo si articola in quattro punti:

- **Individuazione del server DHCP:** il primo compito di un host è identificare il server DHCP. Questa operazione viene effettuata usando un messaggio **DHCP Discover** che un client invia in un pacchetto UDP dalla porta 67. Mette come ind.IP boradcast (255.255.255.255) e come sorgente (0.0.0.0) cioè "questo host".
- **Offerta del server DHCP:** il server che riceve il messaggio risponde con un **DHCP Offer** che viene inviato in broadcast a tutti i nodi della sottorete usando (255.255.255.255)
- **Richiesta DHCP:** il client appena collegato sceglierà l'offerta migliore e risponderà con un messaggio **DHCP request** che riporterà i parametri di configurazione.
- **Conferma DHCP:** il server DHCP risponde al messaggio di richiesta DHCP con un messggio di **DHCP ACK** che conferma i parametri.

Il DHCP consente al client di rinnovare il suo indirizzo IP prima dello scadere.



NAT

Per risolvere il problema della scarsità degli indirizzi pubblici, alcuni spazi di indirizzamento vengono riservati a reti private (es. 10.0.0.0/8). Esistono migliaia di reti private e come viene gestito l'indirizzamento? Con il NAT: In sostanza il NAT si occupa di tradurre l'indirizzo privato in quello pubblico, e viceversa.

ICMP

Il protocollo ICMP è usato per inviare messaggi di errore alla sorgente in caso di problemi. IP e ICMP dipendono uno dall'altro: IP dipende da ICMP per segnalare errori, e ICMP usa IP per trasportare i messaggi di errore. ICMP contiene due tipi di messaggi:

1) Messaggi di **errori** (es. destinazione irrangiungibile, TTL scaduto)
2) Messaggi per **ottenere info** (es echo request)

I messaggi di echo sono usati dall'applicazione *ping*: qundo un host riceve un "*echo request*" ICMP invia un messaggio di "*echo reply*".

NB: ICMP non ha priorità, **ma** se ICMP causa un errore non viene inviato nessun messaggio di errore. (causa valanga)

Consegna Diretta e Indiretta

Diretta: Quando un host invia un messaggio a un altro host della stessa rete (l'indirizzo fisico viene ottenuto tramite ARP)
Indiretta: Quando un host invia un messaggio a un altro host di un'altra rete; Il messaggio passa dal router (gateway predefinito) che si fa carico della consegna. I passaggi intermedi vengono fatti grazie agli algoritmi di roputing.

Algoritmi di Routing

Lo scopo dell'algoritmo di routing è determinare il percorso migliore tra sorgente e destinazione attraverso la rete dei router in base a delle metriche.
Ogni protocollo di routing restituisce una tabella di routing che indicherà quale sarà l'output su cui inviare il pacchetto all'interno del router. I nodi fanno scelte locali basandosi sulla topologia globale (nb. Non tutti hanno una visione globale della rete)
Per far si che tutti i router mantengono le info corrette si scambiano le informazioni periodicamente.
E' possibile rappresentare il problema tramite dei grafi: i nodi rappresentano i router, e gli archi il collegamento fisico temporaneo; un percorso è la sequenza di nodi, e ogni cammino ha il suo costo.

Routing Centralizzato

Calcola il costo minimo tra sorgente e destinazione avendo una conoscenza globale della rete. Questi sono detti *link-state*, dato che l'algoritmo deve conoscere qualunque collegamento della rete.

Routing Decentralizzato

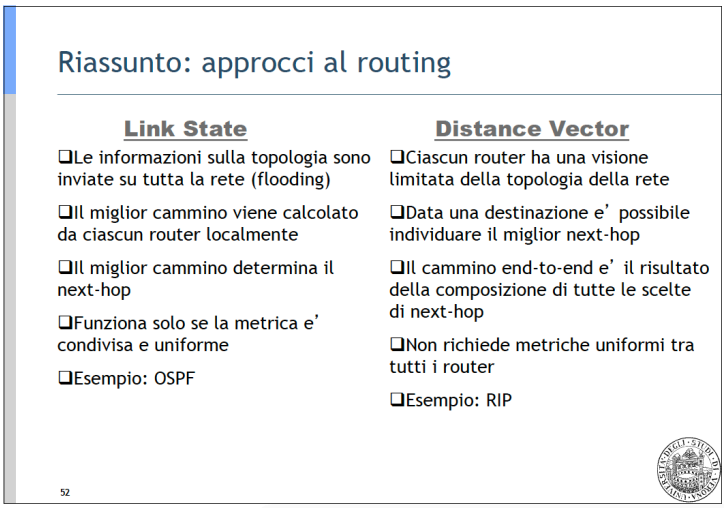
Calcola il costo minimo tra sorgente e destinazione in modo distruibuito e iterattivo: nessun nodo ha conoscenze globali della rete ma solo attraverso uno scambio di informazioni possono calcolare il costo minimo e il percorso migliore attraverso la rete. Questi sono detti *distance-vector*.

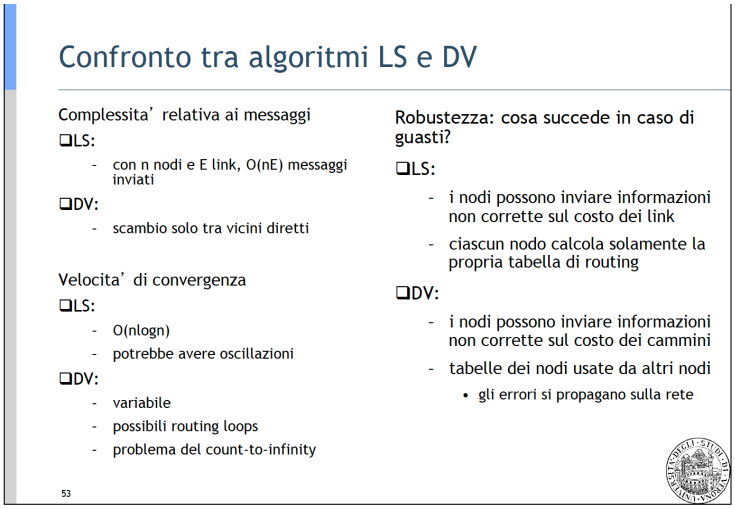
Instradamento LS

La topologia della rete e tutti i costi dei cammini sono noti, ossia disponibili in input all'algoritmo. Si può ottenere in modo che ogni nodo invia il proprio stato a tutti gli altri nodi fino a convergere. L'algoritmo è noto come Dijkstra che calcola il percorso a costo minimo da un nodo a tutti gli altri nodi della rete.
[Algoritmo su slide]

Instradamento DV

Esso è locale: ciascun nodo prende informazioni dai nodi direttamente connessi, a cui resituisce il risultato dopo aver fatto i calcoli;
Iterattivo: il processo DV si ripete fino a quando non c'è stato uno scambio ottimale con i vicini
Asincrono: non tutti i nodi operano con gli altrii. L'algoritmo associato a DV è noto come algoritmo di bellman ford.
[Algoritmo su slide]





Routing Gerarchico

I router visti fino ad ora hanno una visione idealizzata : sono tutti identici e la rete è “piatta”. In realtà è diverso:

1) non si possono memorizzare tutte le destinazioni nelle tabelle di routing, e le tabelle cosi grandi saturerebbero i collegamenti
2) Ciascun amministratore deve avere il controllo della sua rete

Viene introdotto il concetto di **routing gerarchico**:

- ➔ Vengono aggregati i router per le regioni : AUTONOMOUS SYSTEM (**AS**)
  - I router usano lo stesso protocollo di routing all’interno di un AS
- ➔ Nascono i ROUTER DI BORDO (*gateway*): rouer che fanno di collegamento tra due AS

Il routing intra-AS è noto come **IGP.** *(RIP, OSPF ec..)*
Il routing inter-AS è noto come **BGP.** *(permette alle reti di farsi conoscere all’interno della rete)*

Si fa differenza per ridurre la dimensione delle tabelle, per la policy e le prestazioni.