Fondamenti di Reti di Telecomunicazioni

Carmelo Gugliotta 213477

Anno 2021/2022

Contents

1 Lezione n1 28/09/2021 **6**
1.1 Network Core . . . 6

2 Lezione N2 02/10/2021 **7**
2.1 Network Core: Commutazione di Circuito . . . 7
2.2 Network Core: Commutazione di Pacchetto . . . 7
2.3 Reti di accesso: mezzi trasmissivi . . . 8
2.4 Configurazione dei canali e Modi trasmissivi . . . 8
2.5 Topologia di rete . . . 9
2.5.1 Topologia a maglia completamente connessa . . . 10
2.5.2 Topologia a maglia non completamente connessa . . . 11
2.5.3 Topologia ad albero . . . 11
2.5.4 Topologia a Stella . . . 11
2.5.5 Topologia ad anello . . . 12
2.5.6 Topologia a bus . . . 12
2.5.7 Topologie Addizionali . . . 13
2.6 Internet . . . 13
2.7 Funzioni di rete . . . 13

3 Lezione N3 05 10 2021 **15**
3.1 Approfondimento sulle tecniche di commutazione in reti numeriche . . . 15
3.1.1 Commutazione di circuito . . . 15
3.1.2 Commutazione di Pacchetto . . . 16

4 Lezione n4 06 10 2021 **19**
4.1 Protocolli . . . 19
4.2 Architettura . . . 19
4.2.1 Stratificazione . . . 20
4.2.2 Tipi di informazione . . . 21
4.3 Architettura OSI e TCP/IP . . . 21
4.3.1 Architettura OSI . . . 21
4.3.2 Architettura TCP/IP . . . 23
4.3.3 Incapsulamento . . . 24
4.3.4 Critiche al Modello OSI e TCP/IP . . . 24
4.4 Livello Fisico delle Reti: Alcune definizioni . . . 25

5 Lezione n5 12 10 2021 **27**
5.1 Approfondimento Livello di Collegamento . . . 27
5.1.1 Protocolli e le loro Funzioni elementari nel Livello di Link . . . 27
5.1.2 Ricorda... . . . 29

6 Lezione n6 13 10 2021 **30**
6.1 Protocolli ARQ . . . 30
6.2 Diagramma di Timing . . . 30
6.3 Protocollo Stop&Wait . . . 31
6.3.1 Efficienza Stop&Wait . . . 32
6.4 Protocolli Continuous ARQ . . . 32
6.4.1 Go-Back-n . . . 32

6.4.2 Efficienza Go-Back-n . . . 33
6.4.3 Selective repeat . . . 34

7 Lezione N7 19 10 2021 **35**
7.1 Approfondimenti Codice d'Errore . . . 35
7.1.1 Distanza di Hamming . . . 35
7.1.2 Codifica a blocco per controllo d'errore . . . 35
7.1.3 Codici a Ridondanza Ciclica . . . 35
7.2 Approfondimenti ARQ . . . 35
7.2.1 Piggybacking e Pipeline . . . 36
7.3 Protocollo HDLC . . . 36
7.3.1 Struttura del Frame . . . 37
7.3.2 Instaurazione della connessione . . . 39
7.4 Reti in area locale (LAN) . . . 39
7.4.1 Topologia a bus . . . 40
7.4.2 Topologia a stella . . . 41
7.4.3 Topologia ad anello . . . 42

8 Lezione n8 26 10 2021 **43**
8.1 Tecniche di accesso multiplo . . . 43
8.1.1 Allocazione statica (TDMA,FDMA,CDMA) . . . 43
8.1.2 Allocazione Dinamica . . . 44
8.1.3 Polling e Token Ring . . . 45
8.1.4 Riassumendo... . . . 46
8.2 Il protocollo ALOHA . . . 46
8.2.1 Prestazioni . . . 47
8.3 Slotted ALOHA . . . 48

9 Lezione n9 28 10 2021 **50**
9.1 Protocollo CSMA . . . 50
9.1.1 Efficienza . . . 51
9.2 CSMA/CD . . . 52
9.2.1 Efficienza . . . 53

10 Lezione n10 03 11 2021 **54**
10.1 Il protocollo IPv4 . . . 54
10.1.1 Formato dell'unità dati . . . 54
10.1.2 Frammentazione e Aggregazione . . . 57
10.1.3 Indirizzamento . . . 60

11 Lezione N11 09 11 2021 **62**
11.1 Subnetting . . . 62
11.1.1 Subnet Mask . . . 62
11.2 Indirizzi Fisici . . . 63
11.3 Protocollo ICMP . . . 63
11.4 Protocollo ARP . . . 64
11.4.1 Problema della risoluzione dell'indirizzo . . . 64
11.4.2 Funzionamento Protocollo Address Resolution Protocol . . . 65
11.5 Protocollo RARP . . . 66
11.5.1 Funzionamento RARP . . . 66

11.5.2 Altre Soluzioni . . . 66

12 Lezione n12 17 11 2021 **67**
12.1 Livello rete (liv 3): Instradamento dei dati . . . 67
12.1.1 Instradamento Diretto . . . 68
12.1.2 Instradamento Indiretto . . . 69
12.1.3 Tabelle di Routing . . . 70
12.1.4 Sistemi Autonomi . . . 72
12.2 Teoria dei Grafi . . . 73
12.2.1 Proprietà dei grafi . . . 73
12.2.2 Teoria dei Grafi applicata in Reti . . . 75

13 Lezione n13 24 11 2021 **76**
13.1 Algoritmi d'instradamento . . . 76
13.2 Algoritmo di Bellman-Ford . . . 77
13.3 Bellman-Ford vs Dijkstra . . . 79
13.4 Interior Gateway Protocols . . . 79
13.5 Distance Vector . . . 80
13.5.1 Struttura Distance Vector . . . 81
13.6 Link-State . . . 82
13.6.1 Struttura LSP . . . 82
13.7 Riepilogo . . . 83

14 Lezione n14 01 12 2021 **84**
14.1 Lo strato di trasporto . . . 84
14.1.1 Porte TCP e UDP . . . 85
14.2 Protocollo UPD . . . 86
14.2.1 Pacchetto UDP . . . 87
14.2.2 Controllo di errore . . . 88
14.2.3 Congestione . . . 88
14.3 Transimission Control Protocol (TCP) . . . 89
14.3.1 Indirizzamento TCP . . . 89
14.3.2 Formato dell'unità dati TCP . . . 90
14.3.3 Trasferimento dati TCP . . . 92
14.3.4 Instaurazione di una connessione . . . 92
14.3.5 Enumerazione Segmenti . . . 92
14.3.6 Instaurazione di una connessione . . . 93
14.3.7 Rilascio della connessione . . . 94
14.3.8 Trasferimento Dati parte 2 . . . 95
14.3.9 Controllo di Flusso . . . 95
14.3.10 Sindrome della silly window . . . 97

15 Lezione n15 7 12 2021 **98**
15.1 Continuazione Descrizione TCP . . . 98
15.1.1 Controllo di flusso: throughput . . . 98
15.1.2 Controllo di flusso: Window size . . . 99
15.1.3 Controllo e recupero di errore . . . 99
15.2 Lo strato di Trasporto: Controllo Della Congestione . . . 101
15.2.1 Algoritmo di Karn e Jacobson . . . 101

15.2.2 Algoritmo di Jacobson 101

1 Lezione n1 28/09/2021

1.1 Network Core

Come vengono trasferiti i dati attraverso la rete?

1. **Circuit switching** un circuito dedicato per chiamata: Rete Telefonica.
2. **Packet-Switching** i dati sono inviati in rete a pezzi.
    - Le porzioni di informazioni, con cui si lavora,variano al variare del livello di lavoro (Pacchetti,Frame,Segmenti) .

La possibile *degradazione del segnale* è provocata da rumori di vario genere e dalla distanza. Inoltre vi si devono considerare anche le condizioni al contorno, come la possibile eterogeneità dei dispositivi in utilizzo.

Domanda Semplice - Soluzione molto complicata

2 Lezione N2 02/10/2021

2.1 Network Core: Commutazione di Circuito

La **Commutazione di Circuito** utilizza il protocollo end-to-end , richiede risorse utilizzabili solamente in mutua esclusione (No condivisione, per i dispositivi terminali e per i nodi intermedi.

Qualità garantita nella comunicazione vocale ma dispendio di risorse.

Le risorse di rete (Es. Ampiezza di banda) sono dive in pezzi, che possono essere **allocate** alle chiamate o **non utilizzati** (idle) se non occupati dalla chiamata assegnata.

La divisione della banda del link in "pezzi" può avvenire :

- **Frequency Division**
  I Canali sono divisi in sottocanali frequenziali. Tale meccanismo permette l'incremento della quantità di dati in rete, associando un certo numero di flussi a una determinata porzione di sottocanale frequenziale,una migliore gestione delle utenze e vari vantaggi economici a discapito della qualità.

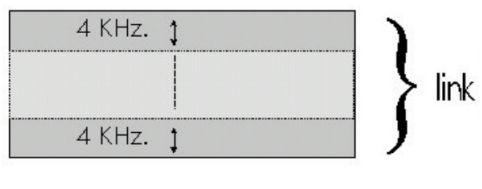


Figure 1: Frequency Divison

- **Time Division**
  Ogni slot corrisponde a una specifica porzione di tempo, e ciò permette di garantire l'elaborazione di una certa quantità di dati, in un intervallo temporale.

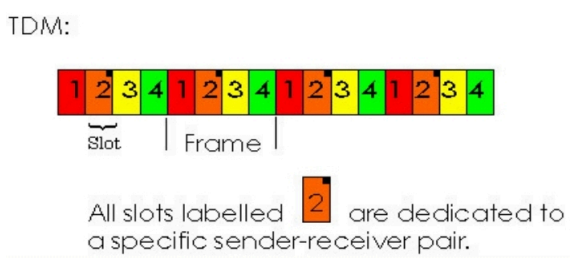


Figure 2: Time Division

2.2 Network Core: Commutazione di Pacchetto

La **Commutazione di Pacchetto** utilizza il protocollo end-to-end,e i flussi di dati (di vario genere) sono divisi in **sotto-unità** di informazioni chiamati **pacchetti**.
I pacchetti associati a flussi di dati differenti **condividono** le risorse di rete, infatti ogni pacchetto usa tutta la banda sul link, però per un intervallo temporale limitato (poiché è una sotto parte dell'informazione). Questo approccio crea una sorta di parallelismo della comunicazione.
La condivisione determina una contesa delle risorse e permette quindi una richiesta delle stesse,

che può superare l'ammontare disponibile. Il costo per questo tipo di approccio è la **congestione** . Durante una fase di congestione vi sono molti pacchetti accodati, in attesa di usare il link.
Si ha quindi uno **store and forward** ovvero, i pacchetti si muovono un *hop* alla volta e la trasmissione avviene sul link e può accadere che si abbia un attesa del turno per il link successivo.

Riassumendo si ha che il **Packet switching** permette alla rete di offrire servizi a un maggior numero di utenti, rispetto al **Circuit switching**.

2.3 Reti di accesso: mezzi trasmissivi

L'utente finale non sempre è collegato in maniera diretta alla core-network e il collegamento può essere di varia natura tramite rete di accesso. In particolare si ha che gli End-System sono connessi agli edge router tramite

- **Reti di accesso residenziali**
  Basato su un accesso punto-punto che può avvenire tramite:
  - **Via modem**: accesso diretto al router fino a 56Kbps.
  - **ISDN** : acronimo di integrated services digital network e basato su connessione al router a 128Kbps.
  - **ADSL**: che sta per asymmetric digital subscriber line con connessione fino a 1 Mbsp home-to-router e fino a 8 Mpbs router-to-home.

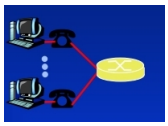


Figure 3: Rete di accesso residenziale

- **Reti di accesso istituzionali**
  Basato su una local area networks (LAN) che permette la connessione del'end-system all'edge router tramite l'utilizzo dell'**Ethernet**, un cavo condiviso o dedicato. (10Mpbs,100Mbsp,Gigabit)

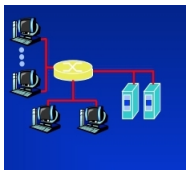


Figure 4: Rete di accesso istituzionale

- **Reti di accesso Wireless**
  Rete di accesso condivisa che connette l'end system al ruoter tramite l'utilizzo di uno spettro radio, wireless LAN.

2.4 Configurazione dei canali e Modi trasmissivi

Dalle informazioni precedenti è facile carpire che, una delle risorse fondamentali delle reti è il canale, ovvero una certa banda di trasmissione caratterizzata da una banda frequenziale e da una certa

velocità di trasferimento dei dati. Inoltre le comunicazioni che possono avvenire sul canale sono di diverso tipo:

- **Punto-punto**
  Due soli nodi collegati agli estremi del canale che viene utilizzato in modo paritetico.

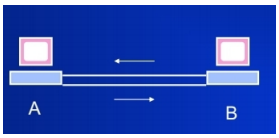


Figure 5: Punto-Punto

- **Punto-multipunto**
  Più nodi collegati ad un unico canale: un nodo master e numerosi slave.

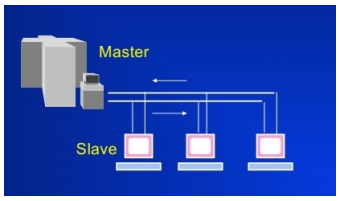


Figure 6: Punto-MultiPunto

- **Canale Broadcast**
  Un unico canale di comunicazione, condiviso da tutti i nodi, l'informazione inviata da un punto è ricevuto da tutti gli altri.
  Chiaramente nei casi in cui si vuole sfruttare un canale Broadcast, ma non si vuole che tutti i nodi processino i dati che arrivano, si utilizzano gli indirizzi. In particolare i dati trasmessi contengono l'indirizzo fisico/logico del nodo di destinazione.

I **Modi trasmissivi** che caratterizzano un canale sono di tre tipi:

- **Simplex**
  tipicamente è una modalità **one-way** ovvero unidirezionale.
- **Half-Duplex** Il canale si presta a far viaggiare i dati in ambedue le direzioni, ma non contemporaneamente.
- **Full-Duplex** Caratterizzato da più linee di trasmissione presenti all'interno di un unico canale fisico che permette la trasmissione dei dati contemporanee in un senso e nell'altro. (Costo maggiore rispetto ai precedenti)

2.5 Topologia di rete

Per **Rete di telecomunicazione** si intende un insieme di **nodi** (Dispositivo terminale/intermedio) necessariamente collegati da canali (con caratteristiche fisiche/chimiche ma anche di comunicazione variabili) che permette la telecomunicazione tra due o più punti.
Nodo: si chiama **nodo** un punto in cui avviene la commutazione.
Canale: si chiama **canale** un mezzo di trasmissione.
La disposizione di nodi e canali definisce la **topologia della rete di telecomunicazione**, bisogna però distinguere:

- **Topologia Logica**
  Definisce l'interconnessione tra nodi mediante canali.

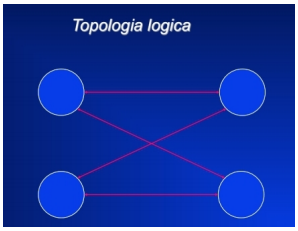


Figure 7: Topologia logica

- **Topologia Fisica**
  Tiene conto del percorso dei mezzi trasmissivi.

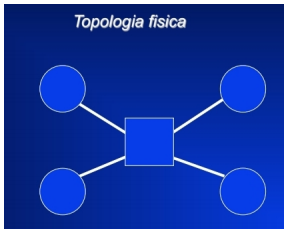


Figure 8: Topologia Fisica

2.5.1 Topologia a maglia completamente connessa

La somma dei collegamenti presenta in una tipologia a maglia (equivale a una serie nota) converge a:

$$numeroCanali = (N * (N - 1))/2$$

Ove N indica il numero di nodi.
Il vantaggio di utilizzare questa topologia è che essendo caratterizzata da molti percorsi tra due nodi, è tollerante ai guasti, il problema è che necessita di un elevato numero di canali.
Infatti è usata solo quando i nodi sono pochi.

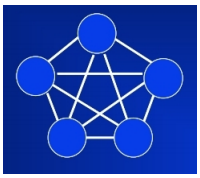


Figure 9: Completamente Connesso

2.5.2 Topologia a maglia non completamente connessa

Il numero di collegamenti che si vanno a coinvolgere oscilla tra un estremo maggiore e un estremo inferiore:

$$N - 1 < numeroCanali < N(N-1)/2$$

È la topologia di rete più utilizzata, pur essendo caratterizzata da una commutazione più complessa per il calcolo dell'**instradamento** e da una topologia irregolare. D'altra parte predispone una tolleranza ai guasti e un numero di canali, selezionabili, così da avere cammini alternativi tra due nodi.

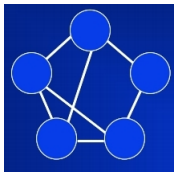


Figure 10: Non Completamente Connesso

2.5.3 Topologia ad albero

Topologia di rete che presenta un numero di collegamenti pari a :

$$numeroCanali = N - 1$$

Tale rete, ha il vantaggio di essere caratterizzata da un basso numero di canali e questo permette di ridurre i costi e semplificare la stesura dei canali, ma proprio per tale caratteristica presenta una elevata vulnerabilità ai guasti (svantaggio).

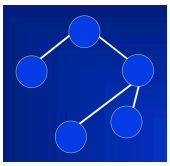


Figure 11: Albero

2.5.4 Topologia a Stella

La comunicazione avviene tramite un nodo centrale ed è caratterizzata da un numero di collegamenti pari al numero di nodi:

$$numeroCanali = N$$

È utilizzata per ridurre i costi e semplificare la stesura dei canali, poiché presenta un numero basso di canali e una commutazione semplice dal punto di vista dei nodi. Ma proprio perché tutto il traffico va al centro stella presenta una elevata vulnerabilità ai guasti del nodo centrale (attivo o passivo).

Figure 12: Topologia a Stella

2.5.5 Topologia ad anello

La comunicazione può essere **unidirezionale** o **bidirezionale** e presenta un numero di collegamenti pari al numero di nodi.

$$numeroCanali = N$$

Nel caso unidirezionale presenta un alta vulnerabilità ai guasti, poiché il malfunzionamento causerebbe l'interruzione della rete, nel caso bidirezionale invece, in caso di guasto, è assicurata la sopravvivenza della rete,anche se a capacità dimezzata, tramite un istradamento alternativo. In ogni caso presenta una commutazione semplice ed è utilizzata molto in reti locali e metropolitane.

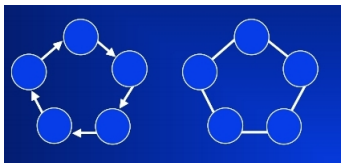


Figure 13: Topologia ad Anello

2.5.6 Topologia a bus

Il bus (broadcast) la comunicazione può essere **attiva** o **passiva**.
Attiva : Ogni nodo, quando riceve un informazione, a prescindere se è esso è interessato o meno, pulisce il segnale lo rigenera e lo inserisce nel canale successivo.
Caratterizzato da un numero di canali pari a:

$$numeroCanali = N - 1$$

Passiva : Serie di nodi collegati ad un unico mezzo trasmissivo tramite adattatori. L'informazione viaggia sul tutto il mezzo e i nodi collegati al medesimo mezzo, potenzialmente possono ricevere il dato.
Caratterizzato da un numero di canali pari a:

$$numeroCanali = 1$$

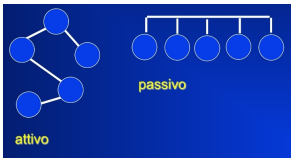


Figure 14: Bus Attivo e Passivo

2.5.7 Topologie Addizionali

Topologia Ibrida
Questo tipo di topologia non può essere associata a un qualcosa di specifico e ben definito, poiché a differenza delle Macro categorie precedentemente analizzate, quest'ultima è caratterizzata da una combinazione di topologie diverse.

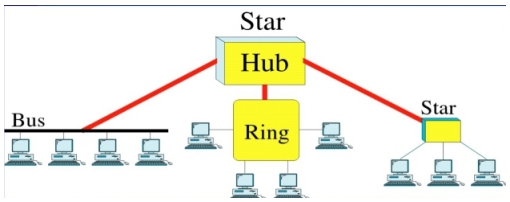


Figure 15: Ibrida

Topologia Gerarchica:
Caratterizzata da un approccio gerarchico non solo a livello organizzativo, ma anche a livello di domini di appartenenza e di gestione.

Esempio struttura rete Telecom Italia su pdf Lez2-intro-parte-II

2.6 Internet

Internet è sostanzialmente una rete di reti, nata spontaneamente interconnettendo sistemi eterogenei ,che permette la comunicazione di milioni di dispositivi (host e end-system) che eseguono le applicazioni di reti, tramite:

- **Link di comunicazione**
  Quali fibra, rame, radio, satellite.
- **Router**
  Che si occupa di inoltrare pacchetti attraverso la rete.

2.7 Funzioni di rete

Le **funzioni** in una rete di telecomunicazioni sono operazioni svolte all'interno della rete al fine di offrire un servizio. Le funzioni principali sono le seguenti:

- **Commutazione e Instradamento**
  La rete individua le risorse necessarie per collegare i due utenti e stabilisce un circuito. In particolare è il processo di interconnessione di unità funzionali, canali di trasmissione o circuiti di telecomunicazione per il tempo necessario per il trasferimento di un flusso informativo verso uno o più canali di uscita.
  Tale funzione va distinta dalla funzione di **Instradamento** (o routing) ovvero, la decisione del percorso che deve essere seguito da un flusso informativo attraverso i nodi intermedi per giungere a destinazione.
  Si hanno tre tipi di **Switching** (Commutazione):
  - Circuit Switching

  - Packet Switching
  - Message Switching

- **Segnalazione**
  Lo scambio di informazioni che riguardano l'apertura e il controllo di connessioni e la gestione di una rete di telecomunicazione. In particolare parleremo di:
  Segnalazione D'utente per il trasferimento delle informazioni di controllo tra utente e rete. Più propriamente tra utente e nodo.
  Segnalazione di Rete trasferimento di informazioni di controllo internamente alla rete che può avvenire ad esempio durante la costruzione di un circuito.
  Segnalazione Internodale per lo scambio di informazioni tra nodi.

- **Trasmissione**
  Il trasferimento di segnali da un punto a uno o più punti.
  Permette la comunicazione tra due utenti.

- **Gestione**
  Importante poiché una rete di telecomunicazione cambia continuamente (allacciamento nuovi utenti, evoluzione tecnologica, riconfigurazione per i guasti).
  Il **Network Management** normalmente consiste in:

  - Gestione della configurazione
  - Gestione delle prestazioni
  - Gestione dei guasti.
  - Gestione della sicurezza.
  - Gestione della tariffazione

3 Lezione N3 05 10 2021

3.1 Approfondimento sulle tecniche di commutazione in reti numeriche

3.1.1 Commutazione di circuito

Come già descritto nella Lezione N1, la rete costituita da tale tecnica, usa le risorse disponibili per allocare un circuito, in uso esclusivo per tutta la durata della comunicazione, a ogni richiesta di servizio. La terminazione della comunicazione, avviene su indicazione degli utenti , e permette il rilascio delle risorse.

Vi sono 3 principali fasi:

1. **Impegno**
2. **Trasferimento dati**
3. **Svincolo**

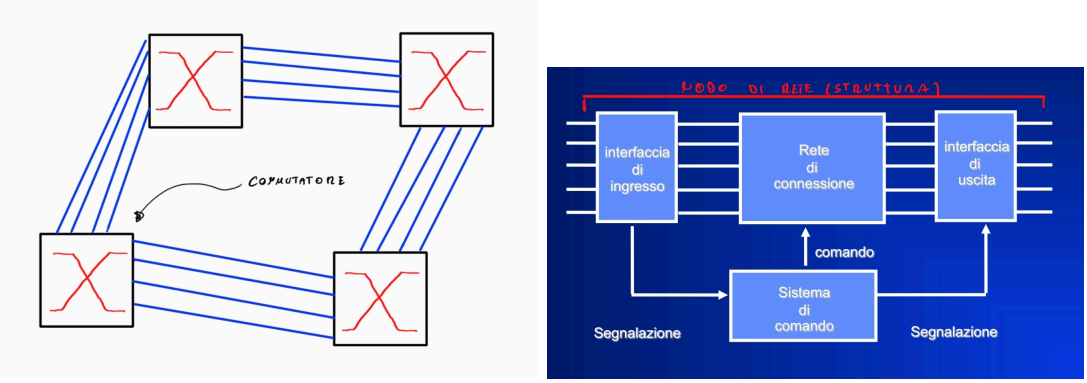


Figure 16: Commutazione di Circuito

I vantaggi della **Commutazione di Circuito**

- **Ritardi di trasferimento costanti**
  Viene garantito poichè calcolato a priori.
- **Trasparenza del circuito**
  I nodi intermedi svolgono tutti la stessa funzione.
- **Bassi ritardi nell'attraversamento dei nodi**

D'altra parte si hanno degli svantaggi non trascurabili. Infatti data la **muta esclusione** sul canale, le risorse sono dedicate a una comunicazione per volta, e questo determina una **buona efficienza** solo in caso di sorgenti **non intermittenti** . Si noti che non vi è nessuna la possibilità di effettuare la conversione di formati,velocità o protocolli e la **tariffazione** si basa sul tempo di esistenza del circuito (Ciò determina maggiori costi).

Il calcolo dell' **Efficienza** η avviene attraverso la seguente formula:

$$\eta = D/(C + D + R)$$

Ove:

- **C**
  Tempo necessario per costruire il circuito. (circa [1,5] s).
- **D**
  Tempo impiegato per trasferire i dati.
- **R**
  Tempo necessario per il rilascio del canale. (circa 1s)

-D Parametro intrinsicamente legato alla tecnologia utilizzata.-
-C e R legati al modo in cui viene utilizzata la comunicazione-

Si noti che il valore di C ed R non sono mai pari a 0, poiché è fisicamente impossibile,ma in condizioni normali, la loro somma vale circa 2 secondi. Utilizzando la formula precedentemente citata, potremo dedurre in maniera abbastanza semplice, che fissati C ed R, per avere un efficienza elevata, il circuito deve essere caratterizzato da una velocità non pienamente soddisfacente.(10kbit/s).

Nelle reti ad alta velocità la commutazione di circuito è sconsigliata per via della bassa efficienza.

3.1.2 Commutazione di Pacchetto

Tecnica studiata espressamente per sorgenti intermittenti (bursty). Il suo funzionamento è analogo al sistema postale: si utilizzano diversi mezzi per fare arrivare a destinazione una lettera.
In particolare l'informazione da trasferire è organizzata in **unità dati** che comprendono l'informazione di utente e di controllo, rispettivamente chiamati **SDU** (service data unit) e **PCI** protocol control information.

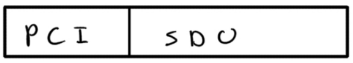


Figure 17: SDU E PCI

SDU: contiene il contenuto informativo vero e proprio
PCI: contiene tutta l'informazione necessaria affinché il pacchetto possa essere identificato e inoltrato fino alla sua destinazione finale.

Tale informazione di utente può dover essere frazionata in molti pacchetti,a cui deve essere associata la rispettiva informazione di controllo.

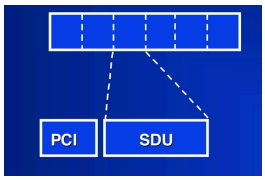


Figure 18: Frazionamento pacchetti

L'informazione viaggia attraverso i nodi intermedi detti **commutatori di pacchetto**, che ricevono un pacchetto, e tramite l'utilizzo dell'informazione di controllo associata, decidono qual è il percorso migliore, che il pacchetto può prendere per raggiungere la sua destinazione. Tale strada dipende prevalentemente dalle condizioni della rete.

Il funzionamento dei nodi intermedi può basarsi su due possibili meccanismi diversi:

- **Store-and-forward**
  Ogni nodo memorizza il pacchetto, lo elabora e determina il canale su cui inoltrarlo. Successivamente mette il pacchetto in coda per la trasmissione sul canale.
  Tale approccio necessità di disporre dell'informazione di controllo prima di poter effettuare l'instradamento (decisione non istantanea che richiede tempo di elaborazione),ed è caratterizzato anche da forme di protezione per evitare violazioni di dati.

- **Non Store-and-forward**
  Ogni nodo memorizza i primi byte del pacchetto,determina il canale su cui inoltrarlo e inizia immediatamente la ritrasmissione del pacchetto senza attendere la completa ricezione. Quindi uno stesso pacchetto può impegnare simultaneamente più nodi e più canali. Da ciò ne consegue una maggiore velocità(minore latenza) dei commutatori (switch) e un minore dispendio di risorse, ma anche un peggioramento della qualità dell'informazione. Si hanno limiti anche riguardanti l'impossibilità di effettuare conversioni di formato e di velocità, o di effettuare possibili controlli d'errore.

  Si noti che se il canale di uscita è occupato, il nuovo pacchetto in arrivo viene memorizzato nel nodo e avviene un meccanismo chiamato **CUT-ThROUGH**, che necessità di molta memoria nei nodi,ed effettua uan riduzione delle latenze solo con basso traffico. (Molto Simile a Store-and-forward). Presenta difficoltà nel gestire situazioni di congestione con conseguente perdite di pacchetti.

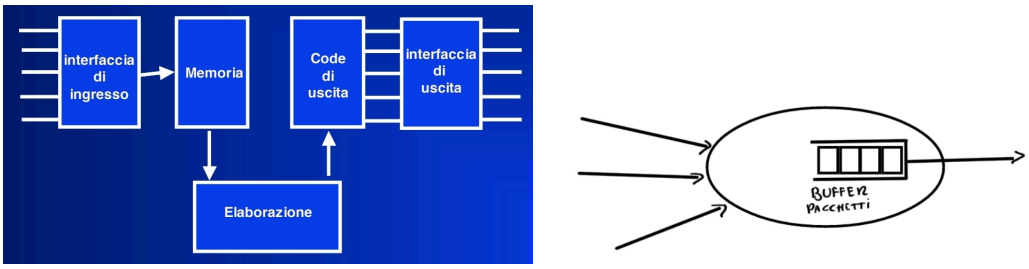


Figure 19: Commutazione di Pacchetto e Buffer interno al nodo

Precedentemente si è detto che l'informazione di utente può dover essere frazionata in molti pacchetti. La lunghezza dei pacchetti è variabile e determinano:

- **Parallelizzazione e Ritardo di pacchettizzazione**
  Pacchetti brevi favoriscono la trasmissione in parallelo e il ritardo di pacchettizzazione. Attenzione si potrebbe avere una maggiore efficienza a discapito di un ritardo complessivo maggiore.

- **Percentuale di informazione di controllo**
  Se il pacchetto fosse troppo piccolo le informazioni di controllo sarebbero proponderanti rispetto all'informazione vera e propria.

  INFO DI CONTROLLO >> INFO UTILE DA TRASMETTERE

  In particolare se la PCI è rappresentata da p bit e la SDU da s bit si ha che la **frazione di informazione di controllo** è pari a:

  $$p/(s+p)$$

- **Probabilità di errore**
  Pacchetti corti riducono la probabilità di errore.
  In particolare data la probabilità di errore p di un bit, dato un canale con errori indipendenti (eventi indipendenti) e dato un pacchetto di n bit, si ha che la probabilità che una pacchetto **sia corretto** è pari a:

  $$(1-p)^n$$

  Infatti se i pacchetti sono costituiti da un numero elevato di bit, la probabilità di errore aumenta.

  per $n->\infty$ questa probabilità tende a zero, qualsiasi sia il valore di p

  Piccola digressione su quello che verrà affrontato nella prossima Lezione

4 Lezione n4 06 10 2021

4.1 Protocolli

Le reti sono molto complesse, e questo rende difficile una loro possibile organizzazione della struttura della rete. Infatti la descrizione e l'analisi delle reti viene semplificata dalla formalizzazione in modelli, che permettono la **Comunicazione** (trasferimento di informazioni) secondo convenzioni prestabilite.

In maniera generica possiamo definire un protocollo come un insieme di regole che governano il trasferimento dei dati tra **entità** che risiedono in diversi **sistemi**.
Formalmente invece si ha che un protocollo è la descrizione formale delle procedure adottate per assicurare la comunicazione tra due o più funzioni dello stesso livello gerarchico.
In particolare si ha che definiscono in una comunicazione:

- **Semantica**:
  Insieme di comandi e risposte, riguarda il significato di ogni sequenza di bit (Come interpretarla).
- **Sintassi**:
  Struttura di comandi e risposte, definisce il formato dei dati, più precisamente l'ordine con cui vengono presentati.
- **Sincronizzazione** Sequenze temporali di comandi e risposte, definisce quindi il quando e a quale velocità i dati vanno inviati.

4.2 Architettura

In generale gli elementi che costituiscono un'architettura di comunicazione sono i seguenti:

- **Sistemi**
  Effettuano il trattamento e/o trasferimento di informazione in vista di specifiche applicazioni
- **Processi Applicativi**
  Coinvolti da esigenze di interazione con altri processi.(Generano e gestiscono l'informazione).
- **Mezzi trasmissivi**
  La struttura fisica di interconnessione tra sistemi.

L'architettura di una rete, è inoltre organizzata in **Strati**(livelli) e **Sottosistemi**. Infatti ogni sistema è logicamente composto da una successione ordinata di **sottosistemi**,ad ognuno dei quali è associato un sottoinsieme funzionale, che raggruppa per logica e per tecnologia realizzativa funzioni simili. Si noti che le **Funzioni** possono essere orientate al **trasferimento dell'informazione** (Connessione fisica,Trasmissione,Controllo di flusso ecc) o **all'utilizzazione dell'informazione** (Gestione del dialogo e adattamento sintattico).

Invece uno **strato** è l'unione di tutti i sottosistemi omologhi(di uguale rango) appartenenti a sistemi interconnessi.Un altra precisione da fare è la seguente, ovvero, ogni strato è **distribuito** (può coinvolgere più sistemi, non solo un sistema), **implementa** alcune funzioni in ogni nodo e **scambia messaggi** tra altri strati omologhi su nodi diversi.

4.2.1 Stratificazione

Si noti che la comunicazione dal punto di vista **Logico** avviene in tal modo:

1. Vengono prelevati I dati dal livello applicazione.
2. Viene elaborato un **Datagramma** tramite l'aggiunta di indirizzamento e info per controllo affidabilità.
3. Viene inviato il Datagramma all'entità pari.
4. aspetta che l'entità pari riscontri la ricezione

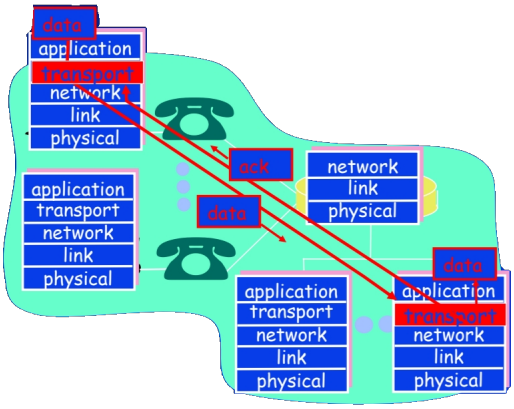


Figure 20: Stratificazione: Modello Logico

Dal punto di vista **Fisco** (Reale) l'informazione attraversa i livelli sottostanti, tramite l'utilizzo di interfacce comuni e funzioni di rete specifiche, per poter essere passata al canale, e tramite il canale giunge al destinatario ove risalirà i vari livelli sino a quello applicativo.

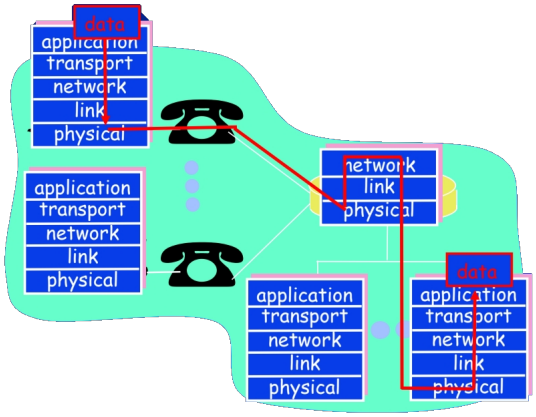


Figure 21: Stratificazione : Modello Fisico

4.2.2 Tipi di informazione

Le informazioni scambiate in un processo di comunicazione sono strutturate in unità informative specifiche di ogni strato dette **Unità di dati** e distinguiamo due tipi di informazioni:

- **Informazioni di utente**
  Sono l'oggetto primario dello scambio per le finalità del processo di comunicazione. (Senze di esse non avrebbero senso le reti)

- **Informazioni di controllo**
  Hanno lo scopo di coordinare le azioni da svolgere in modo cooperativo da parte delle entità di strato.

4.3 Architettura OSI e TCP/IP

4.3.1 Architettura OSI

l'architettura OSI, Open Systems Interconnection, è caratterizzata da 7 livelli diversi:

Figure 22: Architettura OSI

1. **LIVELLO - FISICO**
   Fornisce i mezzi meccanici, fisici, funzionali e procedurali per attivare,mantenere e disattivare le connessioni fisiche. Ha lo scopo di fornire un mezzo di trasferimento da un nodo all'altro dell'unità informativa minima, il bit. Le procedure utilizzate per la trasformazione del flusso informativo di bit in segnali da inviare al mezzo sono, **modulazione/demodulazione**, **codifica/decodifica**, **multiplazione fisica e accesso multiplo fisico**.

Figure 23: Livello-Fisico

2. **LIVELLO - COLLEGAMENTO o LINEA**(Data Link)
   Fornisce i mezzi funzionali e procedurali per il trasferimento delle unità dati tra entità di livello rete e per fronteggiare malfunzionamenti del livello fisico. Ha lo scopo di controllo di rilevare e recuperare gli errori di trasmissione e della gestione del flusso.
   Il trasferimento di unità logiche di bit,dette **trame** su un collegamento avviene tramite **divisione delle trame**, **Controllo d'errore e protocolli di ritrasmissione** e **Controllo di flusso**.

Si noti che nel nodo esiste un'entità di linea per ogni SAP(interfaccia) fisico.
Inoltre per reti broadcast il livello di linea è diviso in due sottolivelli:

(a) **MAC**: Medium Access Control che permette l'accesso multiplo.

(b) **LLC**: Si occupa della gestione delle ritrasmissioni e il controllo del flusso

3. **Livello - Rete**
   Fornisce i mezzi funzionali e procedurali per lo cambio di informazioni tra entità di livello trasporto. Mette a disposizione i mezzi per instaurare, mantenere e abbattere le connessioni di rete tra entità di livello trasporto. Presenta 3 macro funzioni **instradamento**, **indirizzamento**, **controllo di connessione** ma permette anche (solo nell'architettura OSI) di effettuare il **controllo di congestione**, lo **scheduling** e il **controllo della connettività** di rete.
   Si noti che esiste un'unica entità di rete per l'instradamento. L'instradamento che abbiamo

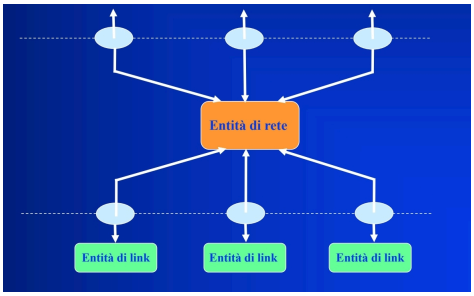


Figure 24: Livello - Rete

citato precedentemente avviene tramite le **tabelle di instradamento**, che consentono di capire ,sulla base dell'informazione presente nei canali di ingresso, verso quali canali di uscita mandare l'informazione considerando che si vuole arrivare verso una certa destinazione.

4. **Livello - TRASPORTO** Fornisce alle entità di livello sessione le connessioni di livello trasporto, colma le deficienze della qualità di servizio delle connessioni di livello rete. Presenta frammentazione di messaggi in segmenti(Unità informativa dette segmento di informazione), e multiplazione (più segmenti in un unico canale logico), suddivisione di connessioni.
   Le funzioni fondamentali svolte sono **connessione**, **controllo di errore e di flusso**.
   Viene implementato nei nodi terminali e permette il trasporto dei dati per il colloquio tra applicazioni residenti in sistemi remoti. Inoltre le entità di trasporto possono essere molteplici in funzione di diversi servizi.

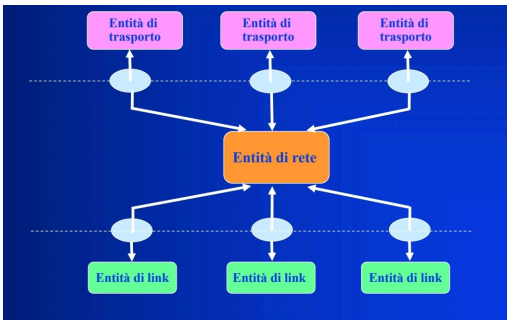


Figure 25: Livello - Trasporto

5. **Livello - SESSIONE**
   Gestisce l'organizzazione del dialogo fra due programmi applicativi di sistemi diversi, in particolare struttura e sincronizza lo scambio di dati in modo da poterlo sospendere, riprendere e terminare ordinatamente, e permette anche di mascherare le interruzioni del servizio trasporto.
   Le funzioni fondamentali difatti sono: **Gestione del dialogo e sincronizzazione tra eventi**

6. **Livello - PRESENTAZIONE**
   Risolve i problemi di compatibilità riguardante la rappresentazione dei dati da trasferire,e i problemi relativi alla trasformazione della sintassi dei dati oltre predisporre di servizi di cifratura delle informazioni.

7. **Livello - APPLICAZIONE**
   Fornisce ai processi applicativi i mezzi per accedere all'ambiente OSI

4.3.2 Architettura TCP/IP

L'architettura TCP/IP è si suddivide in soli 4 livelli, meno modularità rispetto all'architettura ISO/OSI.

Figure 26: Architettura TCP/IP

1. **Livello - ACCESSO ALLA RETE**
   Include le funzioni che nel modello OSI sono comprese negli strati fisico, di link e di rete (Struttura del modello logico ben diversa) quindi comprende le funzionalità per il trasferimento dei dati tra due sistemi terminali connessi alla stessa rete.

2. **Livello - INTERNET**
   Consente l'interfunzionamento delle varie reti con funzionalità, che nel modello OSI ,sono collocate in un livello di rete più alto. In particolare **Indirizzamento** e **Instradamento** senza però un controllo di connessione, quindi ogni informazione viene inviata indipendemente (Generazione di datagrammi in maniera del tutto indipendente) . Di eventuali errori si accorgerà poi il livello TRASPORTO.
   Il protocollo cardine è il protocollo **IP** Internet Protocol.

3. **Livello - TRASPORTO o HOST-TO-HOST**
   Corrisponde allo strato di trasporto e di sessione del modello OSI e i protocolli cardine sono i seguenti:

- **TCP** : Transimission Control Protocol, nell'ambito del quale vi è il controllo di connessione.
- **UDP** : User Datagram Protocol, protocollo alternativo al TCP , che opera senza controllo di connessione, applicando il **Multiplexing** (flussi di applicazioni diverse, in un medesimo canale) non garantendo però sicurezza e affidabilità.

4. **Livello - APPLICAZIONE**
   Presenta dei processi applicativi che consentono la gestione di alcuni tipi di dati, garantendo la loro integrità.
   Corrisponde agli strati di sessione,a presentazione e di applicazione del modello OSI.

4.3.3 Incapsulamento

L'**Incapsulamento** dell'informazioni, all'interno dell'architettura **TCP/IP** e OSI avviene in tal modo:
Ogni livello riceve dati dal livello superiore, tramite un interfaccia che garantisce l'aggiunta , all'informazione utile, di un **Header** senza effettuare violazioni sui dati. Al livello fisico apparirà come un unico flusso di bit e verrà spedita al destinatario ove viene utilizzato il medesimo standard di comunicazione per prelevare l'informazione utile.

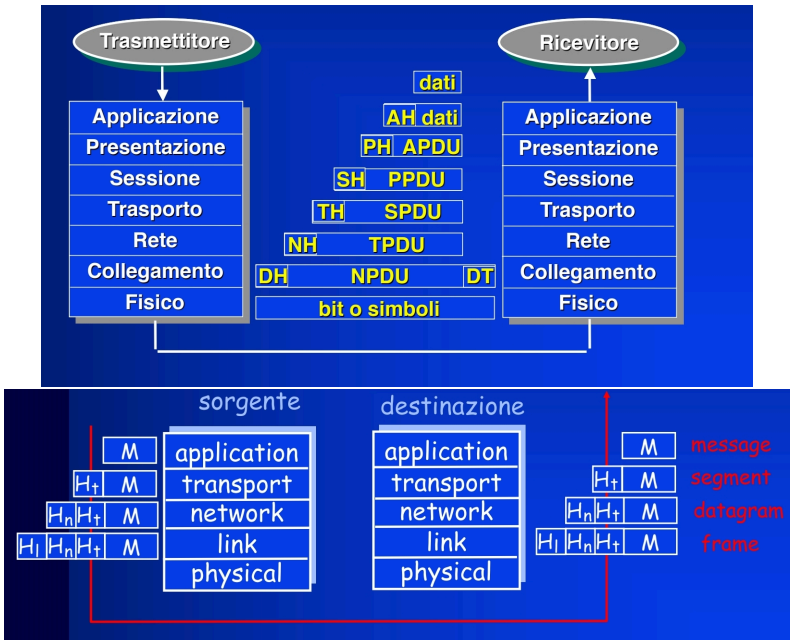


Figure 27: Incapsulamento in OSI e in TCP/IP

Attenzione: la comunicazione orizzontale è puramente LOGICA.

4.3.4 Critiche al Modello OSI e TCP/IP

Il **modello OSI** pur presentandosi con un modello abbastanza generale tale da garantire, a livello di rete servizi con e senza connessione e a livello di trasporto servizi con connessione, i livelli di cui è composto non sono tutti necessari(ripetizione di alcune funzioni a vari livelli : contollo di flusso) e alcuni sono sovraccarichi di funzioni (2,3). Inoltre dalla complessità del modello ne conseguì un implementazione molto pesante. Proprio per questo molti produttori decisero di investire su

TCP/IP, che pur presentandosi senza una chiara distinzione tra servizi, interfacce e protocolli, l'impossibilità di applicabilità ad altri tipi di reti e senza una distinzione tra livello fisico e di data link, garantiva un adattamento perfetto dei protocolli al modello.

4.4 Livello Fisico delle Reti: Alcune definizioni

Abbiamo due tipi di sistemi di comunicazione:

- **Sistema di comunicazione digitale**
  È un sistema che trasferisce l'informazione digitale da una sorgente (digitale) ad un sink (nodi specializzati).

- **Sistema di comunicazione analogico**
  Trasferice l'informazione analogica da una sorgente analogica ad un sink.

Ma cosa intendiamo per informazione **analogica/digitale**? L'informazione viene trasmessa in forma **D'onda**.

- **Analogica**
  Caratterizzata da un dominio continuo e da un codominio anche esso continuo, rappresentata quindi da una funzione f(t) con eventuali caratteristiche di periodicità,frequenza ecc.

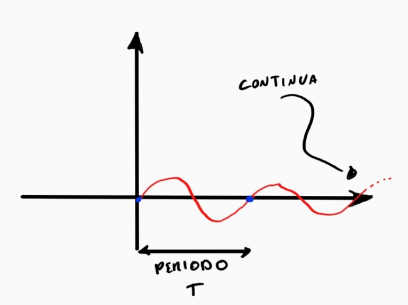


Figure 28: Forma D'onda Analogica

- **Digitale**
  Non mantiene la continuità nel tempo, e il codominio assume dei valori che sono discreti, ovvero non appartenenti a un dominio continuo.

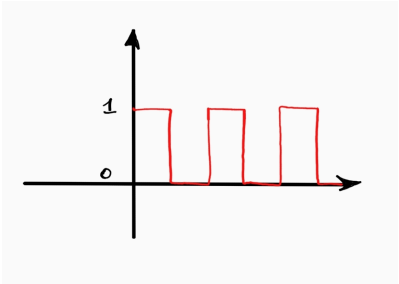


Figure 29: Forma D'onda Digitale

Differenze tra **Sistema elettrico** e **Sistema di comunicazione**:
Nel **sistema elettrico**, in genere le forma d'onda sono sconosciute a priori e la progettazione del sistema ha come obbiettivo ridurre la perdità di energia. Invece **Nel sistema di comunicazione** la forma d'onda che arriva ad un sink è sconosciuta e il sistema deve garantire sicurezza e affidabilità anche in caso di alterazioni introdotte dal rumore (Disturbo).

Vantaggi dell'introduzione della comunicazione digitale:

1. Il circuito digitale è relativamente poco costoso.
2. Possono essere introdotte funzioni come l'encryption per salvaguardare la Pryvacy.
3. I dati provenienti da voce,video,testo possono essere uniformati e trasmessi tramite il medesimo sistema digitale (Uso del multiplaxing), da cui ne consegue un maggiore sharing delle risorse (Nello specifico della natura digitale).
4. Per sistemi di comunicazione a lunga distanza, il rumore non si accumula,tramite specifiche tecniche, da ripetitore a ripetitore (Dispositivi intermedi).
5. Gli errori sui dati rilevati possono essere resi molto piccoli in presenza di "grande rumore" sul segnale ricevuto.(Si riesce a descriminare o valorizzare il segnale utile).
6. Gli errori possono essere corretti attraverso l'uso della codifica.

Ovviamente tali vantaggi hanno un costo, ovvero una richiesta di più banda in trasmissione (più risorse) rispetto ai sistemi analogici e la necessità di sincronizzazione degli eventi.

5 Lezione n5 12 10 2021

5.1 Approfondimento Livello di Collegamento

Il livello **data-link** ha la responsabilità di trasferire i datagrammi da un nodo al nodo adiacente sul link. Fornisce servizi al livello rete e realizza due funzioni principali:

- **Livello Data Link**:
  Si occupa di **Farming**, **Controllo Errori** e **Controllo di Flusso**.
- **Livello Medium Access Control**:
  Nel caso di mezzo condiviso fornisce i mezzi per condividere in maniera ottimale le risorse.

Il livello di collegamento quindi, si occupa del trasferimento dei dati su un link **seriale**, e si hanno due modalità di **trasmissione asincrona** o **sincrona** con protocollo, orientato al **carattere** (byte) o al **bit**, che determina il tipo di comunicazione consentita (**half-duplex** o **full-duplex**) e l'organizzazione della comunicazione sul canale (**master-slave** o **peer-to-peer**.

Si noti che con Protocolli **Orientati al Carattere** (ormai in disuso) si intende che l'informazione numerica è interpretata a gruppi di 8 bit sulla base dei caratteri di un alfabeto(ASCII) e **Orientati al Bit** invece indica che l'informazione è senza interpretazione diretta e vi è un trasferimento "trasparente" dei bit.
Normalmente, su linee a basso bit rate si preferisce un protocollo orientato al carattere, invece su linee lunghe e ad alta velocità si preferiscono protocolli orientati al bit.

5.1.1 Protocolli e le loro Funzioni elementari nel Livello di Link

il protocollo del livello di link è gran parte implementato in un **Adattatore**, anche conosciuti col nome di schede di interfaccia di rete. Tipicamente contengono: Ram,chip DSP, interfaccia per il bus dell'host e un'interfaccia del link. Distinguiamo due lati dell'adattatore **Semi-autonomo**:

- **Lato trasmittente**:
  Incapsula datagrammi in un frame e aggiunge bits di controllo di errore e di controllo di flusso.
- **Lato ricevente**: Controllo di errori e di flusso e si occupa di estrarre datagrammi e inviarli al nodo ricevente.

FRAMING:
Tra le funzioni elementari di un protocollo del sottolivello data-link vi è il **framing**, ovvero la delimitazione e identificazione delle trame.
Infatti nella trasmissione dati i bit trasmessi a livello fisico sono organizzati in gruppi logici: **trame**.
Le quali sono separate con metodi opportuni e identificate tramite un **header** dal livello di linea.
Questa suddivisione del flusso permette di limitare l'inefficienza con un migliore controllo degli errori riscontrati.
I **Delimitatori di trama** ,che consento al ricevitore di riconoscere l'inizio e la fine dell'informazione senza ambiguità, possono essere:

- sequenze di bit(flag) o caratteri speciali inseriti a inizio e fine trama.
- violazione del codice di linea usato a livello fisico.
- temporizzazioni ricevute su canali di servizio.

- conteggio dei caratteri o dei bit.

In particolare si ha che nei protocolli orientati al carattere viene utilizzato il **BSC Binary Synchronous Communications**, che utilizza caratteri speciali per delimitare trame (SYN) e all'interno header(SOH) e dati(STX,ETX).
Invece nei protocolli orientati al bit, viene utilizzato il carattere **DLE** (Data Link Escape) prima di ogni carattere di controllo. Alcuni implementano anche la tecnica del **character stuffing** ovvero che i bit del flusso possono codificare casualmente anche il DLE.
Un esempio di utilizzo di flag è l'**HDLC** ovvero una sequenza di 8 bit all'inizio e alla fine di una trama.

$$0-1-1-1-1-1-1-0$$

Il problema però è che vi è la possibilità che tale sequenza sia presente all'interno del pattern dei dati utili.
Una tecnica utilizzata per evitare tale problema è il **Bit Stuffing**. Tale tecnica si basa sul dare la possibilità al trasmettitore di inserire, quando vi è una specifica sequenza di bit, dei bit aggiuntivi,che permettono di evitare la problematica precedentemente descritta. Il ricevitore, una volta acquisita l'informazione, elimina i bit aggiuntivi.

RILEVAZIONE DI ERRORI:
Altra funzionalità molto importante è la **Rilevazione di errori**. Difatti in ricezione è possibile che venga riconosciuta una sequenza di bit diversa da quella trasmessa (Alcuni bit errati), a causa di **rumore termico**, interferenze provenienti da altre trasmissioni sullo stesso mezzo, disturbi elettromagnetici o perdite di sincronismo.
Tra le tecniche per la protezione dagli errori di trasmissione vi sono:

- **FEC**:
  Forward Error Correction, si basa sul trasmettere un bit e, ripetere la comunicazione di quest'ultimo per N volte, così ché sia possibile effettuare al più, la correzione, di $N/2-1$ errori.
  È osservabile dunque che tramite l'aggiunta della ridondanza in trasmissione si cerchi di rimediare ai bit errati, anche se in numero contenuto.

- **ARQ**:
  Automatic Repeat reQuest, si basa sull'analizzare l'informazione per capire se si ha la necessità di ritrasmissione per eventuali alterazioni.
  È dunque osservabile che l'aggiunta della ridondanza in ricezione per ogni unità informativa è utilizzata per **rilevare** la presenza di errori e non per correggerli.
  Normalmente la ridondanza richiesta per rivelare gli errori è molto più contenuta di quella richiesta per la correzione (16-32 bit sufficiente) e vengono utilizzati messaggi di servizio per la richiesta di ritrasmissioni o la conferma di corretta ricezione.
  Il più semplice codice a rivelazione di errore è costituito dal **Bit di parità**.

Attraverso la tecnica del **Bit di parità**, si permette la rilevazione di errori singoli sul bit, tramite l'aggiunta, alla fine del pacchetto, di un bit pari:

- **1**:
  Se il numero di 1 nel pacchetto è dispari.

- **0**:
  Se il numero di 1 nel pacchetto è pari.

Si noti però che Il livello di collegamento non è l'unico livello, attraverso la ritrasmissione, ad introdurre dei rimedi agli errori, ma si ha anche il livello di Trasporto (Ritrasmissione anche se con approccio diverso) e livello Fisico (Correzione) che collaborano ad offrire un servizio con basso tasso di errore.

5.1.2 Ricorda...

Si ricordi che host e routers,insieme a bridges e switches, sono **nodi**, che i canali di comunicazione che connettono nodi adiacenti lungo un percorso sono **links** e che **PDU** è un frame, incapsula datagrammi.
Inoltre si hanno 3 collegamenti diversi **Point to Point**, **Broadcast** e **Switched**.

ESEMPIO CODICE A RIPETIZIONE SU SAMSUNG NOTES.

6 Lezione n6 13 10 2021

6.1 Protocolli ARQ

Come già descritto in precedenza, sono protocolli, che si basano sulla ritrasmissione automatica. Difatti ipotizzando di avere una stazione **TX** che invia un informazione su un canale e un ricevitore **RX**.

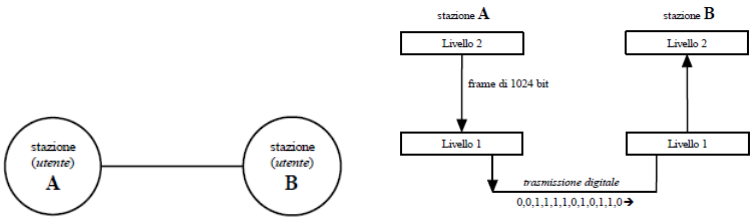


Figure 30: Collegamento

Possiamo distinguere diverse caratteristiche:

- **Velocità di trasmissione -** V_t : Indica la velocità con la quale immettiamo le informazioni sul canale, detta anche **Data Rate**. Dipende dalle caratteristiche del mezzo e dalle tecniche di codifica utilizzate e viene misurata in bit/s
- **Velocità di propagazione -** V_p : Viene misurata in m/s e dipende dalle caratteristiche Chimico/Fisiche del canale.
- **Tempo di propagazione -** τ_p : Viene misurato in s ed è dato dal rapporto tra Distanza m e Velocità di propagazione m/s
- **Tempo di trasmissione -** τ_T : Vine misurato in s ed è dato dal rapporto tra la Dimensione del frame bit e la Velocità di trasmissione bit/s.
- T_T : Indica il tempo necessario a inviare un frame dalla sorgente al destinatario ed è dato dalla somma del **Tempo di trasmissione** e del **Tempo di propagazione**.

6.2 Diagramma di Timing

Il Diagramma di Timing (Diagramma Temporale) , è uno strumento impiegato per spiegare la logica di funzionamento di un protocollo.
Tipicamente strutturato su una linea temporale,associata a una specifica entità (Nodo, Sotto-Sistemi ecc) a cui viene assegnata una direzione.

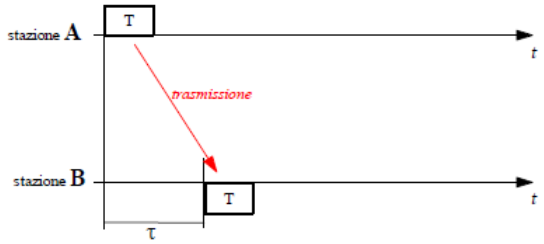


Figure 31: Esempio

Attraverso tale rappresentazione, stiamo descrivendo come due stazioni diverse, utilizzando uno stesso protocollo, interagiscono tra di loro per lo scambio di informazioni (Frame).
Ovviamente il Diagramma, varia al variare del livello di dettaglio che si vuole utilizzare..

6.3 Protocollo Stop&Wait

Lo scambio di informazioni,tramite Protocollo o Tecnica di **Stop&Wait**, si basa sull'invio di un frame da parte di una stazione sorgente (A) a una stazione destinataria (B). Una volta che la stazione destinataria (B) ha ricevuto correttamente il frame trasmesso, quest'ultima si occuperà di inviare, entro un tempo τ_{out}, un riscontro esplicito del corretto ricevimento dell'informazione, verificato tramite un codice di rilevazione. Solo dopo che la stazione trasmittente (A) ha ricevuto un feedback positivo, sarà permesso l'invio di un nuovo frame tramite il canale.
Si noti che inviare e generare un frame, sono due fasi distinte, quindi può essere effettuata la bufferizzazione di più frame, che dovranno essere inviati uno per volta, ottenendo di volta in volto il rispettivo riscontro.
Inoltre la stazione trasmittente non essendo a conoscenza della condizione della stazione ricevente, utilizza un timer, associato ad ogni frame, τ_{out}, che indica il tempo di attesa massimo per la ricezione del riscontro. Scaduto tale timer, la stazione trasmittente assumerà che il frame inviato precedentemente non è stato ricevuto correttamente dalla stazione destinataria. Il τ_{out} deve necessariamente , per non incorrere in problemi di progettazione, maggiore di uno specifico LowerBound ovvero:

$$\tau_{out} \geq 2\tau + 2T_E + T_R$$

Si osservino, i seguenti schemi:

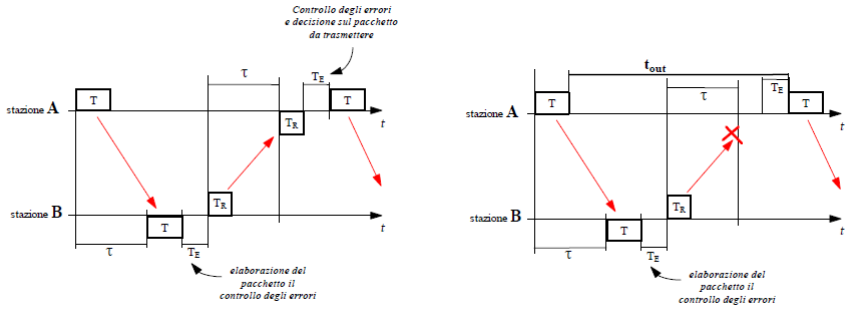


Figure 32: Esempio Senza Errore e Con Errore

Si noti che per discriminare tra frame già ricevuti e frame ancora non ricevuti è necessario e sufficiente un solo bit (0 e 1).

6.3.1 Efficienza Stop&Wait

L'**Efficienza**, nel caso in ci vengano trascurati eventuali errori sul canale (Caso Migliore) è data dal seguente rapporto:

$$E = \tau_T / T_{tot}$$

Ove il T_{tot} rappresenta il tempo in cui il canale risulta occupato e sarà dato quindi da:

$$\tau + T_F + T_E + \tau + T_R + T_E = 2\tau + T_F + 2T_E + T_R$$

Se invece non trascuriamo la possibilità di errore sul canale, allora avremo che l'**Efficienza** sarà espressa come:

$$\tau_T / (E[n_t] * T_{ToT})$$

Ove n_t indica il **Numero di Ritrasmissioni** e quindi si ha che $E[n_t]$ è pari a $1/(1-p)$ ove p indica la probabilità che un generico frame, arrivi a destinazione con errori.

Si noti che anche in presenza di un Canale Ideale(p tendente a 0) e assumendo che il Tempo di Elaborazione e Il Tempo di Riscontro siano trascurabili, l'efficienza sarà sempre un valore minore di uno.

Applicazione Stop&Wait su Samsung Notes

6.4 Protocolli Continuous ARQ

Protocollo che effettua l'invio di un numero finito di frame sul canale, senza la necessità di un riscontro esplicito positivo. Il numero di frame Inviabili è espresso dalla **Finestra in trasmissione** o anche detta Transmission-Window. Ogni volta che viene ottenuto un riscontro, la finestra può traslare e permettere quindi l'invio di ulteriori nuovi frame.

$$X - Riscontri \iff X - Traslazioni$$

Si noti che a differenza di Stop&Wait (che è supportato in presenza di un Half-Dupelx), tale protocollo necessità la presenza del **Full-Duplex**, poiché si effettuano contemporaneamente scambi di informazioni bidirezionali.

6.4.1 Go-Back-n

Go-Back-N è un istanza specifica del protocollo Continuous ARQ, in cui la dimensione della Finestra in trasmissione è pari ad N, ma il ricevitore dispone di una sola posizione di memoria, in cui trattiene il frame ricevuto, per il tempo necessario al controllo di errore. È facile desumere che il **lato ricevitore** diviene **più semplice** da implementare, perché deve tener conto solamente dell'ultimo frame, che rappresenta l'estremo inferiore della finestra, ricevuto correttamente (Se il frame, non rappresenta l'estremo inferiore della finestra, anche se ricevuto correttamente, sarà scartato e sicuramente ritrasmesso successivamente per lo scadere del τ_{out}). Invece il **lato trasmettitore** è **più complesso**, poiché deve permettere la bufferizzazione di più frame, ovvero di quelli non ancora inviati e di quelli non ricevuti correttamente.

Si noti che per discriminare i frame, non è sufficiente un solo bit. In particolare per la corretta

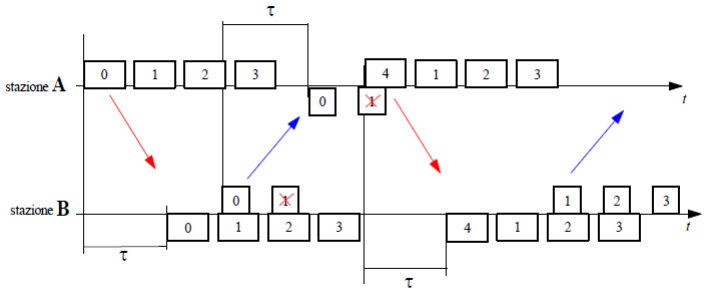


Figure 33: Go-Back-N

gestione dei **Numeri Di Sequenza** (Numerazione dei Frame), sono necessari un numero di Bit pari ad M tale per cui è soddisfatta la seguente condizione:

$$2^M > n$$

Ove n rappresenta la dimensione della Finestra di Trasmissione.

RICORDA: Dalle affermazioni precedentemente effettuate si carpisce che, la trasmissione dei frame deve avvenire in sequenza (in ordine), altrimenti anche se quest'ultimo è ricevuto correttamente viene scartato.

6.4.2 Efficienza Go-Back-n

Elenchiamo alcune assunzioni da effettuare per la costruzione di un modello di analisi di efficienze:

1. Si trascuri il tempo che intercorre tra l'immissione in linea di un frame e del successivo.
2. Si scelga una lunghezza della finestra in trasmissione, tale che appena la stazione di trasmissione tenti di inviare il frame, che rappresenta l'Upper Bound della finestra, arrivi il riscontro del frame, che rappresenta il Lower Bound della stessa. (Se non vi sono errori, si hanno sempre trasmissioni)
3. Si utilizzi un tempo di τ_{out}, minimo, entro la quale può essere effettuato l'invio e l'ottenimento del riscontro del frame.

Per il calcolo dell'**Efficienza**, dalla teoria dei segnali, utilizziamo:

$$E = \frac{\tau_T}{E[T_{tot}]} = \frac{\tau_T}{E[\tau_T + n_r(\tau_T + \tau_{out})]} = \frac{\tau_T}{\tau_T + E[n_r](\tau_T + \tau_{out})}$$

Mettendo in evidenza e semplificando alcuni termini avremo che la precedente equazione diventa:

$$E = \frac{1-p}{1+p\frac{\tau_{out}}{\tau_T}}$$

Ipotizzando di fissare il τ_{out} sull'lower bound e di trascurare T_E (Tempo di elaborazione) e T_R (Tempo di trasmissione del riscontro):

$$E = \frac{1-p}{1+2\frac{\tau}{\tau_{out}}P}$$

Si noti che in condizioni ideali, o meglio, con p che tende a 0, l'efficienza del protocollo è pari a 1.

6.4.3 Selective repeat

Tale tecnica differisce dal **Go-Back-N** per il semplice fatto che, in presenza di un riscontro negativo, la sorgente ritrasmette solo il pacchetto che è stato indicato come errato. Questo è possibile,grazie alla numerazione die pacchetti e dei corrispondenti riscontri.
Si raffigura una possibile situazione:

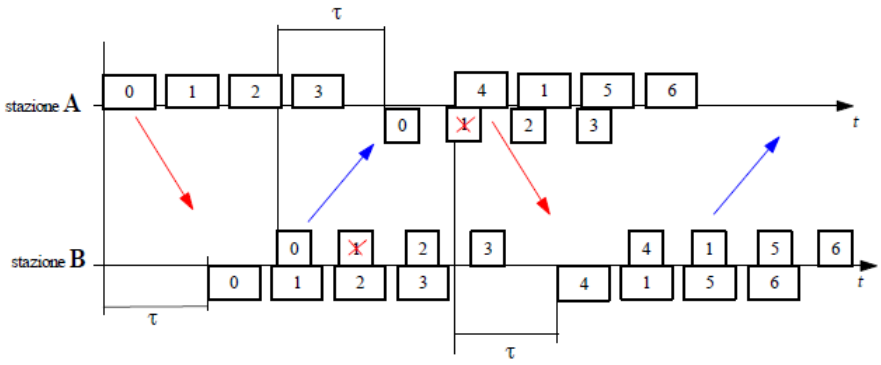


Figure 34: Selective Repeat

Si verifica inoltre, un efficienza praticamente unitaria sempre, per il basso tasso di errore. D'altra parte, l'implementazione di un simile protocollo è più complicata e quindi più costosa,in quanto sia in trasmissione sia in ricezione è necessario un accurato controllo della numerazione dei pacchetti e dei riscontri; in particolare, in ricezione, dati io possibili errori, i pacchetti non saranno mai ordinati correttamente, per cui bisognerà poterli ordinare.

7 Lezione N7 19 10 2021

7.1 Approfondimenti Codice d'Errore

7.1.1 Distanza di Hamming

Si discute quindi, ulteriormente sui codice d'errore; in particolare, la **Distanza di Hamming**, specifica da non sottovalutare, esplica , date due stringhe di egual lunghezza, il numero minimo di errori che possono aver portato alla trasformazione di una stringa nell'altra; In parole semplici la distanza è data dal numero di "caratteri" (bit nel nostro caso) diversi.

Quindi è una specifica che, utile per rilevare o correggere possibili errori nelle sequenze di bit.

7.1.2 Codifica a blocco per controllo d'errore

È utile, per utilizzare nella maniera più corretta i codice di rilevazione o di correzione degli errori, introdurre **bit di controllo** alla sequenza di bit utili o detti anche **bit utente** come nel seguente caso:

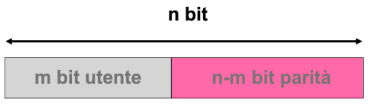


Figure 35: Codifica a blocco

Tali bit di parità sono necessari, per determinare se durante la manipolazione dei dati, vi sono state alterazioni.

7.1.3 Codici a Ridondanza Ciclica

Codice adatto a riscontrare errori di tipo "burst" o a raffica, ovvero errori che compromettono più bit consecutivi. Questo tipo di codice viene anche chiamato codice polinomiale in quanto i bit di dati da controllare possono essere considerati come coefficienti (di valore 0 o 1) di un polinomio che chiameremo M(x). Per calcolare il CRC, oltre al polinomio che rappresenta l'informazione da trasmettere, abbiamo bisogno di un polinomio detto generatore che chiameremo G(x): G(x) deve essere noto sia al mittente che al destinatario, i bit associati ai termini con grado più alto e più basso devono essere uguali ad 1 e il suo grado deve essere inferiore a quello di M(x). Quando il ricevitore riceve il frame, il polinomio M(x) associato a quest'ultimo, viene diviso per G(x). Se c'è resto allora significa che c'è stato errore.

7.2 Approfondimenti ARQ

I protocolli Automatic Repeat reQuest **ARQ**, da come abbiamo visto precedentemente effetuano il controllo congiunto di errore, flusso e sequenza durante una connessione; in particolare con il controllo di flusso, indichiamo la regolazione della **Velocità di invio delle unità informative** in modo tale che quest'ultima non sia superiore a quella con la quale le unità informative vengono smaltite a destinazione. Questo tipo di controllo, ricordiamo, avviene in due livelli : **Livello di Linea (2)** e **Livello di Trasporto (4)**.

Figure 36: Schema Sorgente-Destinazione

7.2.1 Piggybacking e Pipeline

Per aumentare l'efficienza delle tecniche ARQ viste precedentemente, **Go-Back-N** e **Selective Repeat**, in determinati casi viene utilizzato il **Piggybacking**. Tale tecnica,che necessità la presenza di un flusso dati **bidirezionale**, si basa sull'includere nell'intestazione della PDU dati un campo con l'informazione di riscontro **(ACK)** per il flusso di dati che sta fluendo in direzione opposta. In parole più semplici se si hanno due Stazioni A e B che hanno la necessità di scambiare informazioni tra di loro, per aumentare l'efficienza della comunicazione, si può effettuare l'agganciamento dell'**ACK** (associato ai frame che viaggiano nell'altro senso) a un pacchetto contenente informazioni utili, così con un unica trasmissione si esprimono due messaggi diversi (Informazioni d'utente e riscontro positivo a frame ricevuti precedentemente).

In genere tale tecnica è agganciata a tipi di protocolli **Pipeline**, ovvero quei protocolli, che in presenza di buffering al sender e/o al receiver, consentono al sender l'invio di pacchetti multipli senza l'obbligo di riscontri positivi, per supportare meglio la trasmissione continua dei frame.

7.3 Protocollo HDLC

Il protocollo **High-Level Data Link Communications** deriva dal protocollo proprietario **SDLC** Synchronous Data Link Controlo di IBM, e opera in modi differenti, con diversi meccanismi quali **half-duplex** o **full-duplex** e **master-slave** (Conf. sbilanciata) o **peer-to-peer** (Conf. bilanciata).
Presenta tre diverse modalità di funzionamento,a ognuna delle quali è associata una specifica modalità di trasferimento dei frame:

1. **Normal Response Mode (NRM)**:
   Una **stazione primaria** è **collegata** a una o **più stazioni secondarie** tipicamente in modalità **half-duplex**. Solo la stazione primaria può inviare i comandi e le stazioni secondarie trasmettono a seguito di un **polling** (permesso) esplicito inviato dalla stazione primaria. (Conf.sbilanciata)

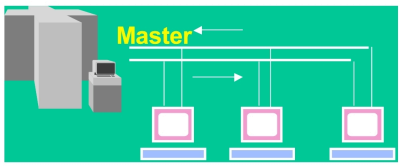


Figure 37: NRM

2. **Asynchronous Response Mode (ARM)**
   Anche in questo caso come nel NRM il colloquio avviene attraverso una **Configurazione Sbilanciata**, ma la stazione secondaria ha la possibilità di iniziare una trasmissione senza il permesso esplicito della stazione primaria iniziando così un colloquio **Full-Duplex**

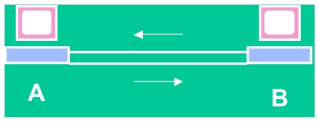


Figure 38: ARM

3. **Asynchronous Balanced Mode (ABM)**
   Fornisce una modalità di funzionamento bilanciato su **Configurazione Bilanciata**, connessione punto-punto tra stazione "combinate" (Sia Primarie che Secondarie) che possono, in modalità **Full-Duplex**, inviare informazioni in modo indipendente ed asincrono. Più adatto con il modello ISO/OSI.

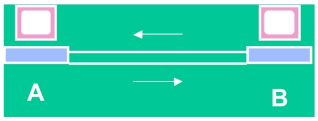


Figure 39: ABM

7.3.1 Struttura del Frame

La struttura del Frame è composta in tal modo:

Flag 01111110	Address	Control	Information	FCS	Flag 01111110
8 bit	8 × *n* bit	8/16 bit	Variable (8 × *m* bit)	16/32 bit	8 bit

Figure 40: Struttura del frame

- **Flag**:
  Utilizzata come **Delimitatore** frame, implementata tramite l'utilizzo del bit stuffing.

- **Address**:
  Normalmente composto da 8 bit, ma può essere di n byte, esplica l'indirizzo della sorgente o del destinatario; In particolare nelle modalità sbilanciate (NRM,ARM) contiene sempre quello della stazione secondaria (ricevitore), invece nella modalità ABM contiene è quello della stazione destinataria.
  L'ultimo bit di ogni byte è usato per indicare se segue un ulteriore sequenza di byte (0 si,1 no).

- **Control**:
  Il campo **Control** contiene le informazioni riguardante **tipo di frame** (Esplicato dai primi bit), **numerazione**, **riscontro**. Vi sono tre tipi di frame possibili:

  1. **Information** (I):
     Sono trame numerate per la trasmissione di informazione d'utente, contenute nel campo **Information**. È possibile trovare l'informazione riguardante il numero del frame corrente e del prossimo frame atteso.

2. **Supervisory** (S):
   Sono trame numerate per il controllo dell'invio del flusso di informazione e difatti il campo **information** è vuoto.

Bit S S	Command	Response	Frame Designation	Descrizione
0 0	RR	RR	Receive Ready	Stazione pronta a ricevere il frame N(R)
0 1	RNR	RNR	Receive Not Ready	Staz. Temporaneamente fuori servizio. N(R) il prossimo frame atteso
1 0	REJ	REJ	Reject	ACK negativo (Go-back-n)
1 1	SREJ	SREJ	Selective Reject	ACK negativo (Selective Repeat)

Figure 41: Tabella Comandi Type Trame S

3. **Unnumbered** (U):
   Sono trame non numerate usate per l'invio di informazione di controllo o per l'invio di informazione in modalità senza connessione e senza riscontro. Effettuano quindi la gestione del collegamento. Caratterizzato da un Campo **M** di 4 bit divisi in due sottocampi **M(1) M(2)**, esplica fino a 32 comandi diversi.

Si osservi nella figura 42 anche la presenza del bit Poll/Final che in **modalità sbilanciata** la stazione primaria effettua un Poll (Bit a 1)(invito a trasmettere verso una secondaria)e la secondaria risponde con una serie di frame in cui P/F=0, tranne l'ultimo che presenta il bit a 1 per inficare un Final; In **Modalità bilanciata** P/F a 1 indica la richiesta di ACK in un comando e la presenza dell'ACK nella risposta.

trame I	**0**	**SN**		**P/F**	**RN**	
trame S	**1**	**0**	**Type**	**P/F**	**RN**	
trame U	**1**	**1**	**M (1)**	**P/F**	**M (2)**	**X**

Figure 42: Trame S I U

- **Information** Tale campo contiene l'informazione d'utente(dei livelli superiori),caratterizzata da una lunghezza variabile, ed è presente sono nelle trame I e trame U, usate per il trasferimento di informazione in modalità connectionless.
- **Frame Check Sequence** (FCS) Contiene il codice rivelatore d'errore usato per riconoscere le trame errate, CRC a 16 bit o a 32 bit.

7.3.2 Instaurazione della connessione

La fase di instaurazione della connessione avviene mediante lo scambio di messaggi che consentono di definire il modo di trasferimento (SNRM, SARM, SABM ove S: Start). Per concludere invece la fase di trasferimento dati la connessione viene chiusa mediante il comando **DISC** che sta per Disconnection.

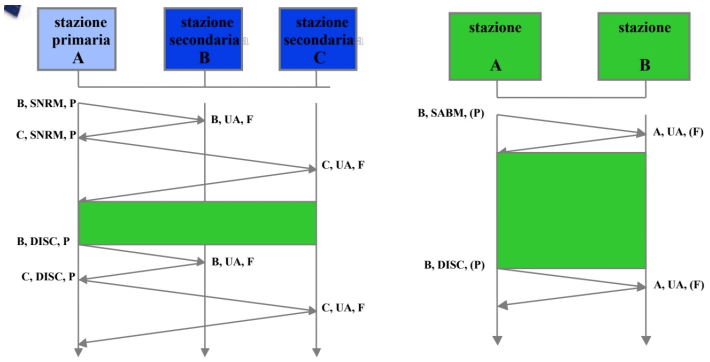


Figure 43: Instaurazione della connessione NRM e ABM

Tenendo conto che, la simbologia utilizzata è la seguente (Address, command, P/F bit) si osserva che come già detto prima, l'**Instaurazione** della connessione avviene in maniera esplicita,da parte della stazione primaria,con bit P/F settato a significato di polling per sollecitare la risposta da parte della stazione secondaria. Nella risposta della secondaria il bit è settato con significato di final per indicare che il controllo ripassa alla stazione primaria. il **Rilascio** della connessione avviene anche esso in maniera esplicita,tramite l'invio, da parte della stazione primaria, dell'apposito comando

che nella prima immagine, la connessione è **Instaurata** dalla stazione primaria, con bit P/F settato con significato di polling per sollecitare la risposta da parte della stazione secondaria. Nella risposta della secondaria il bit è settato con significato di final ad indicare che il controllo ripassa alla stazione primaria. il **Rilascio della connessione** avviene quando la stazione primaria effettua l'invio del comando **DISC**

7.4 Reti in area locale (LAN)

Prima di addentrarci nei protocolli di accesso al mezzo, è opportuno panoramica delle Reti in area locale, anche dette **LAN**, che permettono l'interconnessione di apparecchiature di calcolo in area geografica limitata; Gli **Elementi di Progetto** di una rete LAN, sono fondamentalmente 3 : Il **Sistema di comunicazione**, i **Protocolli di accesso al mezzo** e la **Topologia**, e sono gli elementi attraverso i quali avviene la discriminazione delle reti LAN. (Attraverso la quale si differenziano)

Figure 44: Elementi di una Rete Lan

In particolare, la **Topologia** della rete, che determina dimensioni e forma e influenza il numero massimo di stazioni collegabili, il numero di linee di interconnessione, i costi e la complessità della rete, indica il modo in cui le stazioni in rete sono interconnesse.
Ci sono essenzialmente 3 tipi di tipologie di rete: **BUS**, **STELLA**, **ANELLO**.

7.4.1 Topologia a bus

Consideriamo per prima la topologia a bus:

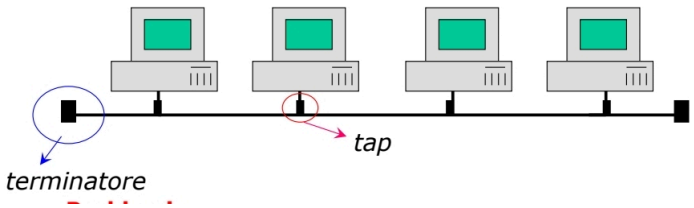


Figure 45: Topologia a BUS

Tutte le stazioni condividono **fisicamente** un unico Mezzo Trasmissivo, intrinsecamente bidirezionale e le connessioni sono di tipo multipunto. La trasmissione del segnale, difatti, si propaga per tutta la lunghezza del mezzo e viene ricevuta da tutte le stazioni.Il segnale alla fine del bus, da entrambi i lati, viene assorbito da opportuni **Terminatori**.

Ma come si garantisce che l'informazione venga memorizzata esclusivamente dal destinatario? Tramte l'**indirizzamento**.
Tramite l'implementazione dell'indirizzamento, ciascuna stazione è caratterizzata da un indirizzo univoco detto **MAC address**, assegnatogli dal costruttore della scheda di rete. Il MAC address della **stazione destinazione** viene agganciato come **header** all'informazione e quando una stazione riceve una **trama** verifica tale campo. Se corrisponde ad un suo indirizzo, inoltra la **trama** ricevuto ai livelli superiori, altrimenti scarta l'informazione.

Tale tipo di topologia, non è adatta per mezzi trasmissivi in fibra ottica, ed è impiegata, infatti, tramite un **Cavo coassiale** al quale le stazioni sono collegati con agganci passivi detti **tap**.
Tipicamente il **Cavo coassiale** in **Banda Base** (baseband), è caratterizzato da una trasmissione bidirezionale e utilizza una codifica **Manchester** sui segnali digitali, ovvero **Basso-Alto** per trasmettere 1 e **Alto-Basso** per trasmettere uno zero.
Per quanto riguarda il **Cavo coassiale** a **Banda Larga** (Broadband), permette la trasmissione di segnali analogici ed è Unidirezionale, e proprio per tale motivo permette due configurazioni diverse:

- **Configurazione split**: La risorsa del mezzo trasmissivo viene suddivisa in due parti, una in cui trasmettiamo con una frequenza f1 e la trasmissione con senso opposto con una frequenza f2.

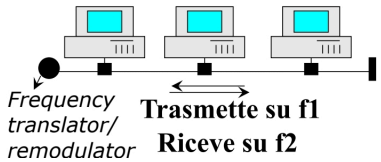


Figure 46: Configurazione Split

- **Configurazione duale**: Basato su un doppio collegamento alle stazioni,uno per effettuare le Trasmissioni e il secondo per le Ricezioni.

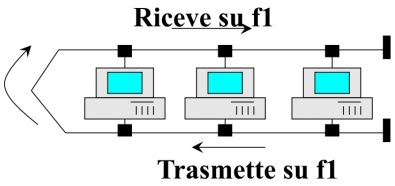


Figure 47: Configurazione Duale

7.4.2 Topologia a stella

Consideriamo la topologia a stella:

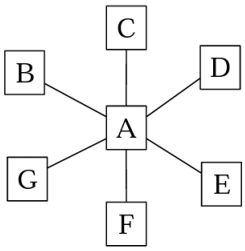


Figure 48: Topologia di rete a stella

Tutte le stazioni sono collegate ad una stazione centrale **HUB** (centro-stella) e tali connessioni sono di tipo **punto-punto**. Si adotta una scelta di questo tipo, tipicamente, quando si vuole mantenere un **controllo centrale** di tutte le connessioni tra coppie di interlocutori. Il controllo centralizzato è implementato tramite l'utilizzo di un **HUB**, che svolge la funzione di ripetitore; In particolare,nelle reti a stella **diffusiva**, quest'ultimo può ricevere un flusso di bit su un determinato canale, e successivamente rinviare il segnale ricevuto verso tutti i nodi; dal punto di vista logico, si comporta quindi come un BUS, ma dal punto di vista reale, l'informazione è ricevuta da una sola stazione grazie all'**Indirizzamento**.
Questo non accade però nelle reti con topologia a stella **non diffusiva**, realizzate tramite uno Switch (Più costose), in cui l'HUB rinvia il segnale ricevuto solo al destinatario, aumentando così la "banda" disponibile.
Se da una parte, abbiamo **Alte prestazioni, Facilità di controllo** e bassa vulnerabilità al guasto (Ovviamente non deve guastarsi il centro stella), in caso di **intenso traffico**, il nodo centrale può risultare sovraccaricato e questo potrebbe portare al blocco delle richieste di connessione.

7.4.3 Topologia ad anello

Nella topologia ad anello, tutte le stazioni sono collegate in una caratteristica **configurazione circolare**, chiusa su se stessa, nella quale le stazioni sono tra loro collegate tramite linee punto-a-punto.

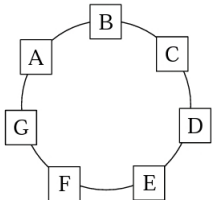


Figure 49: Topologia di rete ad anello

La trasmissione avviene in un unico senso prefissato e i nodi sono **logicamente** condivisi. Infatti tutte le stazioni prendono parte alla trasmissione : quando una stazione invia sulla linea il proprio pacchetto, questo percorre l'intero anello, in quanto ciascuna stazione riceve il pacchetto, lo memorizza, lo rigenera e lo ritrasmette sulla linea successiva; se il pacchetto era diretto ad esso inoltra il pacchetto anche agli starti superiori. Quando il mittente riceve il pacchetto, inviato in precedenza, lo rimuove dall'anello.
Tale tipo di approccio, permette un elevatissima estendibilità della rete, ottima quindi per le fibre ottiche, ma anche un elevatissima probabilità di guasto; difatti il malfunzionamento di una singola stazione provoca la caduta dell'intera rete e anche l'inserimento di un eventuale nuova stazione, rende necessario, interrompere il funzionamento dell'intera struttura.
Per eliminare i problemi appena descritti,

8 Lezione n8 26 10 2021

8.1 Tecniche di accesso multiplo

Quando l'accesso ad una risorsa può avvenire da parte di più utenti indipendenti,si parla di **risorsa condivisa** ed è necessaria l'implementazione di particolari protocolli di **accesso multiplo** che permettono il multi-accesso, al canale di comunicazione che è di tipo broadcast, da parte di varie stazioni (Sorgenti e/o ricevitori di informazione).

Ricorda: Funzione svolta dal sottolivello MAC del livello Link.

Il **problema** dell'accesso multiplo è che se due o più trasmissioni, avvengono contemporaneamente, si verifica una **collisione** ovvero i segnali si sovrappongono in **totalità** o solo parzialmente, non permettendo di ricevere correttamente il segnale.

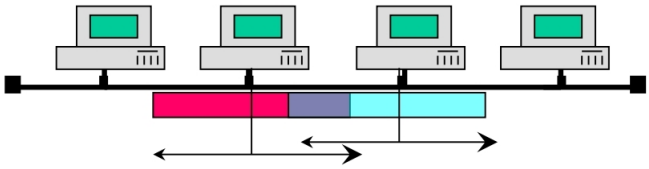


Figure 50: Esempio Collisione

I protocolli hanno, difatti, il compito di regolare la trasmissione delle stazioni in modo tale che non si sovrappongono sul mezzo condiviso e se avviene una collisione, quest'ultima deve essere individuata, in modo da poter ritrasmettere le trame corrotte.

Le tecniche di Accesso multiplo si suddividono in due Macro categorie **Allocazione statica** e **Allocazione dinamica**.

8.1.1 Allocazione statica (TDMA,FDMA,CDMA)

Le tecniche di Accesso multiplo con allocazione statica sono anche dette tecniche di **Channel Partitioning** (Partizionamento del canale), poiché il canale viene diviso in "pezzi" più piccoli , in base a caratteristiche temporali, di frequenza o di codice , ognuno dei quali è assegnato con accesso esclusivo ad una stazione soltanto. Da ciò ne consegue **l'impossibilità di collisione** e **di ritardi deterministici** che caratterizzeranno la comunicazione.

Come protocolli si ha:

- **TDMA**: Il **Time division multiple access** si basa su un accesso al canale a **turno**, ovvero ad ogni stazione viene assegnato uno slot temporale specifico, durante la quale può essere effettuata la trasmissione del pacchetto. (Gli slot non utilizzati restano inutilizzati "idle").

- **FDMA**: Il **frquency division multiple access**, divide lo spettro di frequenza del canale in bande di frequenza, ognuna delle quali viene assegnata a una specifica stazione. (Le bande di frequenza inutilizzate restano "idle")

- **CDMA** Il **Code Division Multiple Access**, utilizzato soprattutto in canali wireless broadcast, assegna un unico codice (sequenza di chip specifica), ad ogni utente, con la quale vengono codificate le informazioni trasmesse.

$$segnaleCodificato = datiOriginali \times sequenzaChip$$

$$decodifica = segnaleCodificato \times sequenzaChip$$

L'operazione sopra riportata, non è una moltiplicazione, ma un operazione di **convoluzione**, difatti, tutti gli utenti condividono la stessa frequenza e quindi vi è trasmissione simultanea ma l'interferenza, grazie ai codici e le operazione di decodifica e codifica, è resa minima. (Per essere minima i codici devono essere "ortogonali", ovvero il codice in convoluzione al segnale codificato associato, restituisce il segnale originale, e in convoluzione al segnale codificato non associato si annulla)

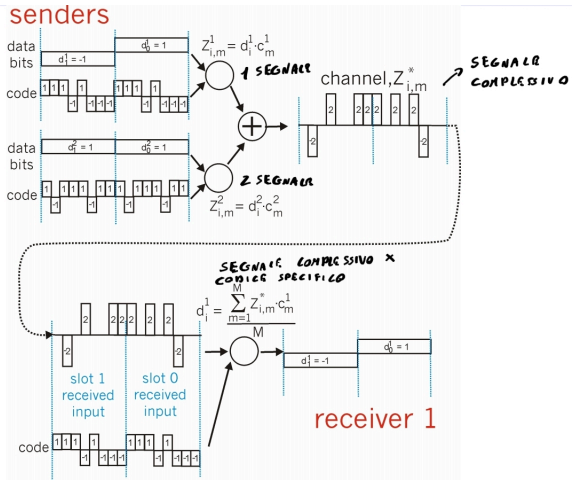


Figure 51: Esempio Interferenza di due Utenti

Fondamentalmente però, come si è potuto notare dalla descrizione dei protocolli, con l'allocazione statica, l'accesso multiplo, nelle reti locali, non viene completamente risolto a causa della natura del traffico su tali reti; In particolare, la rigidità con la quale vengono suddivise le risorse tra le stazioni, impedisce che durante i periodi di silenzio le risorse inutilizzate siano assegnate a stazioni che hanno necessità di smaltire più velocemente eventuali pacchetti e da ciò ne consegue un **throughput basso** e **ritardi elevati**.

Si noti che il traffico sulle LAN è molto variabile ed è detto bursty, poichè caratterizzato da periodi di attività nei quali si effettua la trasmissione di pacchetti ad un ritmo sostenuto e periodi di silenzio

8.1.2 Allocazione Dinamica

Le tecniche di accesso multiplo con allocazione dinamica, effettua l'assegnazione delle risorse su domanda e la gestione dell'accesso al mezzo e sono detti **Protocolli MAC**, (Medium Access Control); si distinguono in base alla modalità di risoluzione delle contese di utilizzazione, in **Protocolli MAC con collisione** e **Protocolli MAC senza collisione**, ma in tutte e due le casistiche si hanno dei **Ritardi non sempre controllabili**; in particolare si ha che:

- **Allocazione Dinamica Casuale**:
  La stazione tenta di inviare un pacchetto, alla massima velocità permessa dal canale, e se il mezzo è **libero** la emissione ha **successo**,altrimenti occorre riprovare successivamente. Vi è collisione, poiché non vi è **coordinazione** a priori tra i nodi, e nel caso due o più nodi trasmettono informazioni (collisione), i pacchetti vanno tutti distrutti.

All'aumentare del traffico generato dalle stazioni, aumenta anche il numero di collisioni e ciò può limitare fortemente il traffico globale.

Inoltre viene specificato come **rilevare le collisioni** (Mancata ricezione corretta dell'informazione) e come **recuperare le informazioni non trasmesse correttamente**. Per rilevare l'avvenuta della corretta ricezione, si utilizza il riscontro (**ACK**), che può essere fornito da **osservazione diretta** (canale broadcast) o **indiretta** (è il ricevente che segnala la corretta ricezione). Altre informazioni sono fornite dal **feedback di canale**, che una stazione può ottenere da un'osservazione del canale. (Esempi **ALOHA**, **slotted ALOHA**, **CSMA** e **CSMA/CD**.

- **Allocazione Dinamica Controllata**:
  Si distinguono i casi di Controllo Centralizzato e Controllo Distribuito; nel **Controllo Centralizzato** si ha che una delle stazioni (primaria) provvede ad abilitare ognuna delle altre (Secondarie) ad trasmettere.(Esempio **Polling**).
  Nel **Controllo Distribuito** il controllo passa ordinatamente da stazione a stazione, e le stazioni non attive non assorbono risorse; ne consegue un'efficiente ripartizione della capacità di trasferimento del mezzo fra le sole stazioni attive.(Esempio **Token Ring**).

8.1.3 Polling e Token Ring

Il **Polling** è un protocollo con allocazione dinamica controllata, ove ciascuna stazione, a turno, ha l'opportunità di trasmettere. Se non ha trame in coda declina l'opportunità di trasmettere, se invece ha trame in coda, trasmette fino ad un numero massimo di trame che dipende dal **tempo di servizio**; successivamente il diritto di trasmettere passa alla stazione successiva.
il **Tempo di servizio**, ossia il tempo per cui una stazione può trasmettere una volta ottenuto il permesso può essere **Illimitato** e quindi si ha un servizio esaustivo o **Limitato** e quindi si ha un servizio limitato, o meglio, k-limitato in cui si consente la trasmissione di k pacchetti consecutivi prima di rilasciare il controllo alla prossima stazione.
Il **Polling** consente si un **perfetto coordinamento tra le stazione** per evitare la sovrapposizione nelle trasmissioni (PRO) ma è caratterizzato da una bassa efficienza nel caso di reti con molte stazioni che generano poco traffico in modo fortemente intermittente, o meglio, si hanno **tempi di attesa molto elevati** (latenza) da parte delle poche stazioni che hanno necessità di trasmettere informazioni. Inoltre i messaggi di polling ricevuti dalle stazioni e i successivi messaggi di rilascio del canale, inviati alla fine della trasmissione da ogni stazione, usano le risorse del mezzo trasmissivo e quindi **riducono la banda disponibile** per le informazioni d'utente. (polling overhead)

Il **Token Ring** si basa sul passaggio di un **token message** da un nodo all'altro in modo sequenziale; Il nodo mittente attende la ricezione del Token per modificarlo **on the fly** e accodargli la trama informativa. Successivamente, il nodo destinatario riconosce l'indirizzo e legge la trama (per inoltrarla agli strati superiori). Quando la trama giunge di nuovo al nodo mittente, questi la rimuove ed emette un nuovo Token che puo essere catturato dai nodi a valle. La criticità di tale

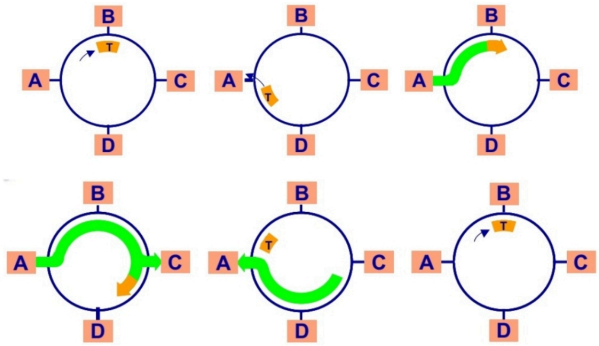


Figure 52: Esempio Token Ring

protocollo è l'overhead del token (utilizzo di banda), la latenza ed è caratterizzato da un singolo punto di fallimento (token).

8.1.4 Riassumendo...

- **Protocolli ad allocazione statica**:
  Presentano una condivisione efficiente del canale a carico alto, ma inefficienti a basso carico, poiché alloca ad ogni stazione 1/N di banda allocata anche se c'è solo 1 solo nodo attivo.

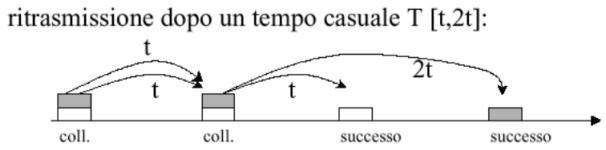
- **Protocolli ad accesso random**:
  Efficienti a basso carico, poiché un solo nodo può sfruttare tutto il canale, per gestire situazioni di alto carico ha la necessità di introdurre un overhead (Minore banda disponibile per informazioni di utente).

- **Protocolli con allocazione dinamica controllata**:
  In caso di controllo centralizzato presenta bassa efficienza nel caso di reti con molte stazioni che generano poco traffico.
  In caso di controllo distribuito il sistema è più robusto a guasti o malfunzionamento rispetto a quelli a controllo centralizzato, ed essendo il coordinamento più schematico presenta un efficienza più alta anche se rinuncia alla conoscenza completa dello stato del sistema e quindi accettano gestioni che possono non essere le migliori.

8.2 Il protocollo ALOHA

È il protocollo più semplice e fa a meno di ogni forma di coordinamento fra le stazioni. Infatti ogni stazione inizia la trasmissione non appena ha dei pacchetti da trasmettere, senza considerare in nessun modo le condizioni di occupato/libero del canale e tale pacchetto, se la trasmissione va a buon fine, viene ricevuto indistintamente da tutte le stazioni. Ciascuna stazione, una volta ricevuto il generico pacchetto, esamina l'indirizzo del destinatario: se si riconosce in tale indirizzo, compie sul pacchetto le necessarie elaborazioni, altrimenti lo scarta visto che non le interessa.
Dal momento che ogni stazione ignora completamente tutte le altre, un pacchetto subirà una collisione se la sua trasmissione si sovrapporrà tutta o in parte a quella di un altro pacchetto, e verrà ritrasmesso dopo un tempo **X**, variabile casuale di opportune caratteristiche (leggi deterministiche di ritrasmissione).
La sola informazione di servizio necessaria è il **Riscontro** dell'avvenuta corretta trasmissione, ottenuta dalla stazione ricevente.
Si noti che la ritrasmissione di un pacchetto non avviene dopo un tempo fisso t ma dopo un tempo

casuale $T \in [t, 2t]$ in modo da ridurre la probabilità che due stazioni che hanno colliso collidano di nuovo.



Inoltre, ipotizzando un tempo di trasmissione di un pacchetto pari ad 1, se una stazione genera un pacchetto nell'istante t_0, tale pacchetto collide se almeno un'altra stazione genera un pacchetto tra $t_0 - 1$ e $t_0 + 1$ Questo intervallo di tempo, durante il quale i pacchetti possono subire collisioni con i pacchetti di altri utenti, viene definito **periodo di vulnerabilità** V ed è pari a $2T$ ove T è la durata del pacchetto.

8.2.1 Prestazioni

Si effettuino le seguenti assunzioni:

- I pacchetti hanno lunghezza costante (durata T).
- Il data rate del canale è fissato.
- Gli utenti generano nuovi pacchetti a istanti casuali.
- Indichiamo con $\lambda \frac{pkt}{s}$ l'intensità del traffico totale generato dalle stazioni (sia pacchetti trasmessi per la prima volta e sia pacchetti che avendo subito collisioni, vengono ritrasmessi). E per semplicità è di tipo poissoniano.

Avendo effettuato le assunzioni precedenti, possiamo introdurre G, ovvero il numero medio di pacchetti complessivi, inviati sul canale, nell'unità di tempo T che è pari:

$$G = \lambda T$$

Nello specifico G è il **traffico normalizzato sul canale** (in Erlang) dovuto ai pacchetti arrivati e bufferizzarti e quindi è una misura relativa all'utilizzazione del canale: Proprio per questo il valore di G deve necessariamente essere compreso tra 0 ed 1 (Se G fosse maggiore di 1 i pacchetti generati supererebbero il max rate di trasmissione del canale).

Considerando un intervallo di tempo di durata T: grazie alle caratteristiche del processo di Poisson, si ha che la probabilità che ci siano n arrivi nell'intervallo T è data da:

$$P(n, T) = \frac{(\lambda T)^n}{n!} e^{-\lambda T} = \frac{G^n e^{-G}}{n!}$$

Inoltre supponendo che l'intervallo T considerato sia quello necessario ad una data stazione per trasmettere il proprio pacchetto, affinché non ci sia collisione, è necessario che, durante l'intervallo T, non ci sia alcuna trasmissione, ossia $n = 0$. Ma in realtà c'è un altra condizione per evitare la collisione, ovvero che quando la stazione comincia a trasmettere, è necessario che nessun'altra lo stesse già facendo. Di conseguenza poniamo $n = 0$ ma l'intervallo totale durante il quale si può verificare una collisione non è T, ma $2T$ (periodo di vulnerabilità). Di conseguenza **la probabilità di non avere collisioni**, detta probabilità di successo Ps, con riferimento ad un intervallo $2T$ è pari a:

$$Ps = Pr(0, 2T) = e^{-2G}$$

Dalle ultime relazioni introdotte, possiamo esplicitare un parametro di prestazione molto importante, ovvero, il numero medio di pacchetti trasmessi con successo nell'intervallo di tempo T, il cosidetto **throughput**:

$$S = Ge^{-2G} \text{ varia tra [0,1]}$$

Sostanzialmente, abbiamo esplicitato, che il **througput normalizzato** S è pari al prodotto del carico offerto totale per la probabilità di successo della trasmissione e possiamo vedere tale valore come la frazione di tempo (frazione di Erlang) in cui il canale è utilizzato.

Quanto più S tende ad 1,tanto migliore è l'efficienza del sistema.

Si può diagrammare l'andamento di S in funzione di G, ottenendo una curva del tipo seguente:

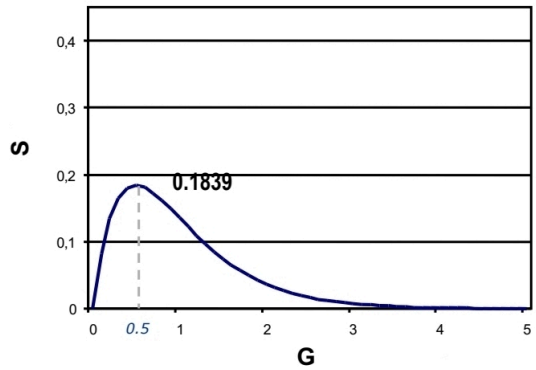


Figure 53: Grafico S unslotted-ALOHA

Come si nota, la curva possiede un valore massimo ottenibile per $G = \frac{1}{2}$ infatti:

$$\frac{dS}{dG} = e^{-2G} - 2Ge^{-2G} = 0$$

$$G_{max} = \frac{1}{2} \text{ e } S_{max} = \frac{1}{2}e = 0,1839$$

8.3 Slotted ALOHA

il protocollo **slotted ALOHA** prevede invece che ogni stazione possa trasmettere appena vuole, ma possa farlo solo entro un intervallo temporale ben definito, detto **slot**; in generale,gli slot sono di durata costante e uguale per tutti (ma si può anche pensare di fornire slot più lunghi a quelle stazioni che mediamente generano più pacchetti). E' evidente che, almeno a livello medio, con questo protocollo l'occupazione del canale da parte di una stazione sia minore rispetto a quella del protocollo un-slotted, nel quale non esiste alcuna limitazione temporale.
È evidente che, dal punto di vista del generico slot temporale messo a disposizione, le situazioni possinili sono solo 3:

- **La prima** è quella per cui nessun utente ha necessità di trasmettere, per cui non ci sono richieste di servizio e quindi lo slot è vuoto;
- **La seconda** situazione è invece quella in cui un solo utente ha fatto richiesta di servizio e quindi ha occupato lo slot per la trasmissione dei propri dati.

- **L'ultima** situazione è infine quella in cui ci sono state 2 o più richieste di accesso allo stesso slot temporale, per cui c'è stata collisione e nessuna trasmissione è stata effettuata,e le steazioni proveranno a effettuare la ritrasmissione dopo un numero di slot scelto uniformemente in un intervallo [0,r].

  Se r=0 la collisione si ripete all'infinito; in particolare va scelto un valore di **r piccolo** in situazioni di **rete scarica** e **r grande** in situazioni di **congestione** per evitare instabilità.

In particolare r sarà un intero X scelto a caso ed in modo uniforme nell'intervallo $[0, 2^{Min(k,max)}]$ ove k è il numero di collisioni già subite dal pacchetto e max è settato a priori per limitare la dimensione massima dell'intervallo.
Assumendo un parziale coordinamento fra le stazioni, consistente nel sincronismo dei possibili istanti di trasmissione, che distano T (slotting del tempo) si ha che le stazioni, hanno un riferimento temporale e possono trasmettere solo iniziando al tempo $T, 2T$... Ciò evita il caso di sovrapposizioni parziali delle trasmissione (Collisione solo tra pacchetti che partono nello stesso istante di tempo).
Da ciò si ha che **il periodo di vulnerabilità** è pari alla durata di un pacchetto T e quindi **la probabilità di successo** PS è:

$$Pr(0, T) = e^{-G}$$

E quindi si ha che il numero medio di pacchetti trasmessi con successo nell'intervallo di tempo T è pari:

$$S = Ge^{-G}$$

Diagrammando S in funzione di G otteniamo:

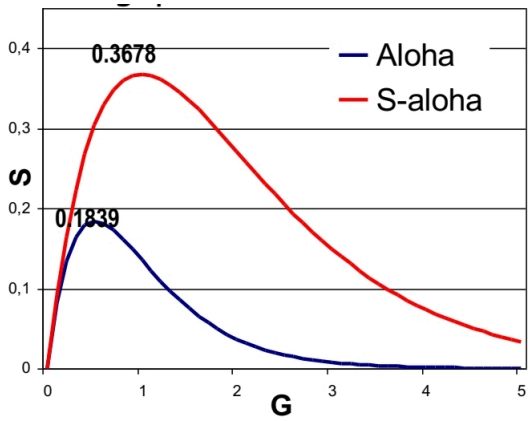


Figure 54: Grafico S slotted-ALOHA

Si osserva che il massimo dell'efficienza si ottiene in corrispondenza di $G = 1$, ossia in presenza di una sola richiesta, mediamente, di accesso al canale in ogni slot di lunghezza T e $S =' \frac{1}{e}$ che corrisponde circa al 36%.

9 Lezione n9 28 10 2021

9.1 Protocollo CSMA

Il protocollo **Carrirer Sense Multiple Access**, che sta per accesso multiplo a rilevazione di portante, deriva direttamente dall'ALOHA, ed è perciò un *protocollo ad allocazione dinamica casuale* (sono possibili le collisioni), con l'aggiunta del feedback che riguarda l'occupazione del canale. Quindi la differenza sostanziale rispetto all'ALOHA è quella per cui una stazione, prima di trasmettere, "ascolta" se il canale è già stato occupato per una trasmissione e trasmette solo in caso negativo. Ciò è possibile grazie al **Carrier Sensing** che è lo strumento che rivela l'occupazione del canale e dà il nome al protocollo. Si può osservare, duenque che la **procedura** è la seguente:

1. Una stazione che desidera emettere ascolta se il canale è occupato da una emissione precedente.
2. Se il canale è libero (idle), la stazione emette.
3. Se il canale è occupato (busy), la stazione ritarda l'emissione.

Ma di quanto sarà ritardata l'emissione dell'informazione? Si hanno due approcci diversi *CSMA persistente* e *CSMA Non-persistente*. Il **CSMA Persistente** riporva immediatamente con probabilità p quando il canale diventa idle, e ciò può causare instabilità. Il **CSMA Non-persistente** riprova dopo un intervallo di tempo random.
Un meccanismo così descritto, apparentemente sembra evitare le collisioni. ma invece a causa dei **ritardi di propagazione** non evita il pericolo delle collisioni.

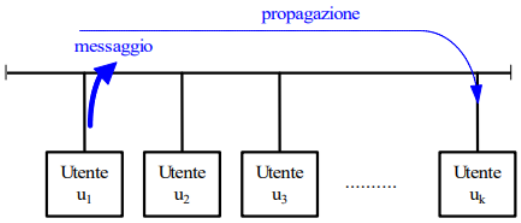


Figure 55: Esempio Collisione CSMA

Supponiamo, come mostrato, che la prima stazione abbia appena rilevato il canale libero ed abbia così deciso di immettere sulla linea il proprio pacchetto ed impiega un tempo T. Il segnale immesso sul mezzo, impiega a sua volta un certo tempo per propagarsi e raggiungere tutte le stazioni. Questo tempo è in funzione della distanza (Inversamente Proporzionale) tra le stazioni e della velocità di propagazione (Direttamente Proporzionale).
La stazione che per ultima riceve il messaggio è quella più lontana dalla stazione sorgente, supponiamo quindi che il la propagazione del segnale impieghi un tempo τ per percorrere tutta la linea. È evidente che, nell'intervallo $[t_0, t_0 + \tau]$, il segnale non ha ancora raggiunto l'ultima stazione, di conseguenza, il canale risulta libero per tale stazione, per cui potrebbe sentirsi autorizzata a trasmettere, elaborando così una collisione.

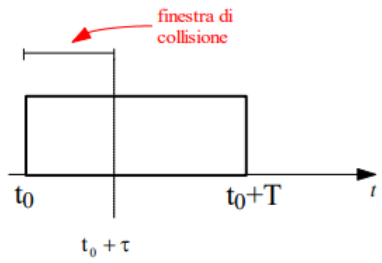


Figure 56: Finestra di collisione

In particolare, possiamo affermare che ipotizzando τ il ritardo di propagazione, se una stazione genera un pacchetto nell'istante t_0, tale pacchetto collide se almeno un'altra stazione genera un pacchetto tra $t_0-\tau$ e $t_0+\tau$. Tale intervallo di tempo prende il nome di **intervallo di vulnerabilità** ed è pari a 2τ.
Come già detto, in precedenza, se le stazioni rilevano uno stato di busy, del mezzo trasmissivo (bus bidirezionale), ritardano la trasmissione; Ma di quanto? Ritardano la trasmissione di una quantità di tempo che varia al variare dell'approccio utilizzato.

- **1-persistente**:
  Il terminale ascolta finché il canale è libero poi trasmette con probabilità 1 (trasmette appena libero).

- **non-persistente o 0 persistente**:
  Il terminale aspetta un tempo random prima di ritrasmettere (come se avesse colliso)

- **p-persistente**:
  il terminale aspetta finché il canale è libero e quindi trasmette con probabilità p, oppure con probabilità 1-p ritarda la trasmissione di un tempo casuale.

9.1.1 Efficienza

/ IL PROFESSORE DURANTE LA LEZIONE HA PARLATO DI PRESTAZIONE DEL CSMA P PERSISTENTE MA HA SCRITTO LE PRESTAZIONI DELLA MODALITÀ NON PERSISTENTE CHE È L'UNICA TRATTABILE PROPRIO PER LA SEMPLICITÀ DELLE ASSUNZIONI CHE POSSONO ESSERE FATTE SUL TRAFFICO DEL CANALE/

Analizziamo le prestazioni, della modalità **non-persistente**, l'unica che consente di trattare facilmente il traffico sul canale, introducendo il valore $a = \frac{\tau}{T}$ ovvero il rapporto tra il **tempo di propagazione** associato al mezzo trasmissivo e **il tempo di trasmissione del pacchetto** dipendente dalla lunghezza del frame e dal data rate che caratterizza il canale in uso. Difatti si può mostrare che il **Throughput** è pari a:

$$S = \frac{Ge^{-aG}}{G(1+2a)+e^{-aG}}$$

La quantità al **numeratore** indica la quantità, di dati utili, trasmessa con successo da parte delle stazioni; Al **denominatore** invece viene espresso il traffico generato totale da parte di tutte le stazioni (dato grezzo ecco perché al denominatore), in particolare $(G + G2a)$ esplicita il traffico emesso dalle varie stazioni (G) e il traffico trasmesso nell'intervallo 2τ, che potenzialmente deve essere ritrasmesso $(2aG)$ invece il termine e^{-aG} indica la trasmissione di un determinato frame avvenuto con successo.

S all'interno del CSMA è data dal rapporto tra il traffico utile e il traffico totale.

Diagrammando S in funzione di G (variando anche il valore di a) otteniamo:

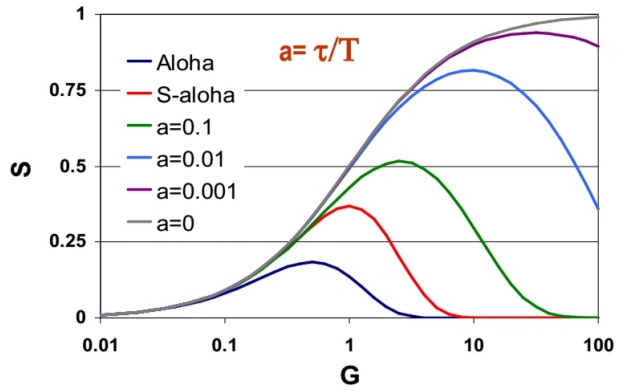


Figure 57: Grafico CSMA

9.2 CSMA/CD

Una variante del CSMA è il CSMA/CD, anche detto **Carrier Sense Multiple Access - Collision Detected**. Essa è stata resa popolare dalle reti locali di tipi **Ethernet** ed è stata standardizzata con il nome di **IEEE 802.3**. Si distingue dal CSMA poiché, ogni stazione non "ascolta" il canale solo prima di trasmettere ma anche mentre sta trasmettendo informazioni utili; in particolare, non appena una stazione si accorge di un eventuale collisione, smette di trasmettere, riducendo così la durata delle collisioni e lo spreco delle risorse di rete, con un netto miglioramento delle prestazioni. Il vantaggio rispetto al CSMA è tanto più elevato, quanto più è piccolo il tempo necessario affinché le varie stazioni coinvolte nella collisione se ne accorgano rispetto al tempo di trasmissione della trama.
In particolare, elaborando un diagramma di timing:

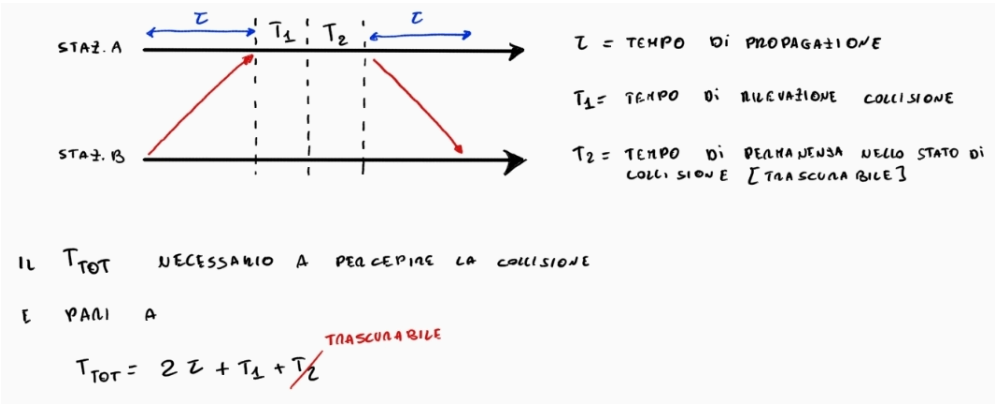


Figure 58: Diagramma Timing CSMA/CD

È osservabile che, si ha la necessità di una **lunghezza minima del frame**, affinché il meccanismo di Collision Detection possa portare a un miglioramento dell'efficienza, che deve essere pari a:

$$L_{min} = (2\tau + T1)R \text{ Ove R rappresenta il Data Rate}$$

9.2.1 Efficienza

Assumendo sempre la modalità non-persistente e indicando con:

- T: tempo di trasmissione del pacchetto:
- τ: tempo di propagazione;
- $a = \frac{\tau}{T}$;
- δ: tempo per accorgersi della collisione e interrompere.

Il throughput sarà pari a:

$$S = \frac{Ge^{-aG}}{G(1+2a) + e^{-aG} - G(1-\delta)(1-e^{-aG})}$$

Ove alcune quantità sono già state descritte nel CSMA, e quindi ci rimane da osservare la quantità $G(1-\delta)(1-e^{-aG})$ che esplica la quantità di dati che verrebbero inviati senza la rilevazione della collisione; in particolare $1-\delta$ è la frazione di tempo rimanente dopo la rilevazione della collisione e $1-e^{-aG}$ è la probabilità di collisione.

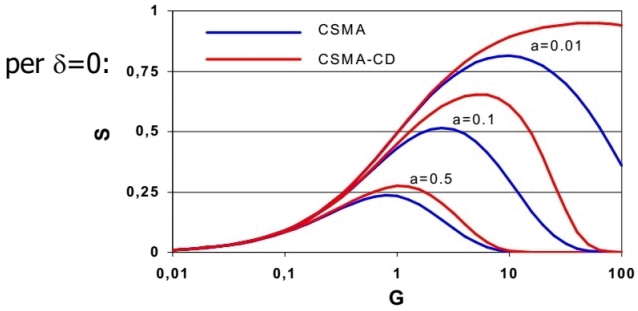


Figure 59: Prestazioni CSMA/CD

10 Lezione n10 03 11 2021

10.1 Il protocollo IPv4

IP è un protocollo di livello network (strato 3) dell'architettura TCP/IP, che fornisce un servizio di trasferimento delle unità informative con modalità a **datagramma**, **senza connessione**, **inaffidabile**.
il termina *senza connessione* indica che ciascuna unità informativa viene gestita indipendentemente dalle altre poiché ognuna può seguire una strada diversa per arrivare a destinazione, e non viene conservata alcuna informazione di stato sulle unità di dati inoltrate.
il termine *inaffidabile* indica che non è garantita la consegna di un'unità informativa a destinazione; infatti un'unità dati può essere persa, duplicata, ritardata, o consegnata fuori sequenza.
Allora chi fornisce la garanzia sulla qualità di servizio? La qualità è demandata agli strati superiori residenti negli host e dipende strettamente dalle caratteristiche delle sotto-reti attraversate.
Tale protocollo opera come segue:

- **In trasmissione**:

  1. Riceve i dati dal livello transport e li incapsula in pacchetti di dimensione massima pari a **64 kbyte** (anche se la dimensione media è in realtà circa 1500 byte).
  2. Instrada i pacchetti sulla subnet, eventualmente frammentandoli lungo il viaggio.

- **In ricezione**:

  1. Riassembla (se necessario) i frammenti in pacchetti.
  2. estrae da questi i dati del livello transport.
  3. consegna al livello transport i dati nell'ordine in cui sono arrivati (che non è necessariamente quello in cui sono partiti).

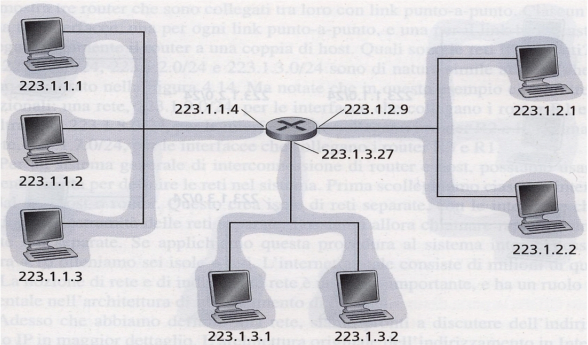
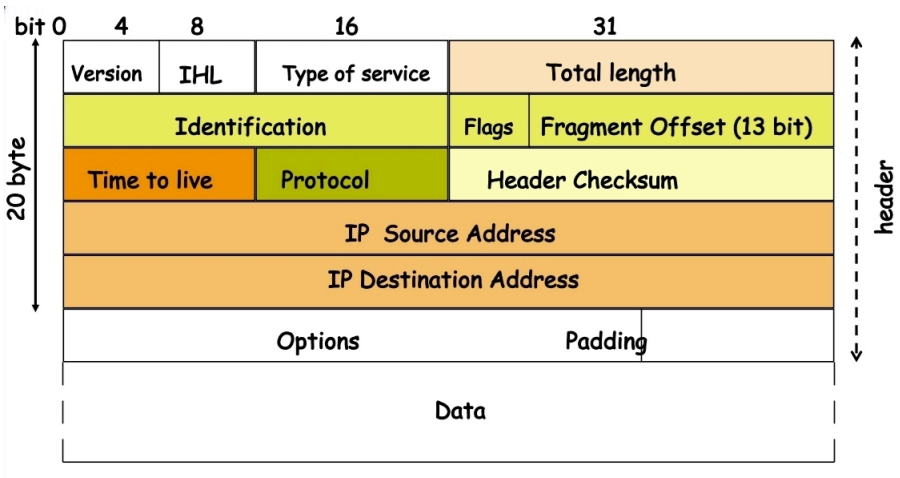


Figure 60: Esempio di comunicazione a livello di rete tramite indirizzamento IP

10.1.1 Formato dell'unità dati

Le unità-dati dello strato IP sono dette **datagrammi**, composto da un **campo informativo** che contiene i dati utente e da un'intesatzione **header**. L'header ha una parte **fissa** di **20 byte** e una parte, **opzionale**, di lunghezza variabile. La figura seguente illustra la struttura dell'**header IP**:
Si analizzano con dettaglio le funzioni dei vari campi dell'header:



- **Versione** (4 bit):
  Descrive la versione del protocollo utilizzato, in modo tale da rendere il passaggio tra diverse versioni di IP.

- **IHL** (4bit):
  Descrive la lunghezza dell'intestazione in parole da 32 bit; il **valore minimo** è 5 ($5 * 32 = 160bit = 20byte$) ed è quello che si ha quando non è presente nessuna *opzione sul pacchetto*; in presenza invece di una o più opzioni, la lunghezza cresce fino ad un **massimo** di 15 (dati i 4 bit: 1111) e si ha una lunghezza complessiva dell'header ($15 * 32 = 480bit = 60byte$) (40 byte dedicati alle opzioni, 20 byte sono fissi.

- **Type of Service** (TOS) (8 bit):
  Specifica parametri della qualità di servizio richiesti dall'utente; ad esempio, per la voce digitalizzata si richiede la massima velocità di trasmissione, mentre invece per il trasferimento file si richiede una trasmissione priva di errori.
  Nella versione originale era formato da **2 sottocampi** : un sub-campo **precedence** di 3 bit per specificare l'importanza del datagramma; e un sub-campo di 3 bit per richiedere particolari prestazioni come:

  - **D-bit** (Delay): Se settato a 1 basso ritardo.
  - **T-bit** (throughput): Se settato a 1 alta portata.
  - **R-bit** (reliability): Se settato a 1 alta affidabilità.

  Si noti che si sono utilizzati solo 6 di 8 bit, infatti 2 sono erano riservati per usi futuri. **Nella versione Aggiornata**, solo **uno** dei due bit è stato **aggiunto** al secondo sub-campo per indicare la richiesta di **minimizzare il costo**.
  Nel caso nessun bit è a 1 allora servizio normale.
  Si noti inoltre che specificare il **TOS** significa effettuare la richiesta di un servizio migliore se è possibile, ma se non è possibile il datagramma non viene scartato.

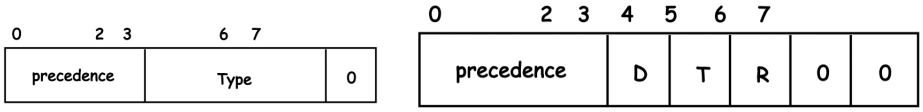


Figure 61: Struttura OLD_TOS and NEW_TOS

- **Total Length** (16 bit:
  Specifica la lunghezza dell'intero pacchetto IP (header+dati); il valore massimo è di 65535 byte;la minima è di 576 byte poiché permette di trasmettere una quantità ragionevole di dati oltre all'header (se fosse più piccola avremmo problemi di efficienza). In generale si ha sempre una lunghezza multipla di 4byte.

- **Identification** (16bit):
  Consente all'host destinazione di determinare a quale pacchetto appartiene un dato frammento, in quanto **tutti i frammenti di uno stesso pacchetto hanno lo stesso valore di questo campo**

- **Flags** (3 bit):
  È un campo di bit di controllo, il primo rimane inutilizzato e posto a zero. Successivamente si hanno due campi:

  - **DF** (Don't Fragment) ordina ai route, quando posto a 1, di non frammentare i pacchetti.
  - **MF** (More Fragments) serve ad indicare se un pacchetto, diviso in frammenti, è terminato (MF=0) oppure non lo è ancora (MF=1); Tutti i frammenti tranne l'ultimo hanno MF=1.

- **Fragment Offset**(13 bit):
  Indica la posizione del frammento attuale, all'interno del datagramma originario; consente infatti di ricostruire il pacchetto originale a partire dai singoli frammenti in cui è stato scomposto, rispettando l'ordine corretto. Tutti i frammenti, tranne l'ultimo, devono essere multipli di 8 byte. Dato che il campo Fragment offset è lungo 13 bit, è possibile avere un massimo di 8192 frammenti per pacchetto. ($2^{13} = 8192$ frammenti ciascuno di 8byte quindi 65536 byte).

- **Time to Live** (8 bit) indica quanto tempo il datagramma può rimanere all'interno della rete; nello specifico è un contatore (inizializzato al valore 255=28) che viene decrementato di uno a ogni *salto* (hop), e quando il contatore arriva a zero, il pacchetto viene scartato. In questo modo, si evita che un pacchetto resti nella rete per sempre.

- **Protocol** (8 bit):
  Indica a quale processo dello del livello di trasporto (superiore deve essere consegnato il contenuto informativo del datagramma. Le varie possibilità sono TCP,UDP e altro.

- **Header Checksum**(16 bit):
  il campo serve a verificare solamente la correttezza dell'header (controllo di errore). Il funzionamento è il seguente:
  Si sommano le parole a 16 bit dell'header e si memorizza il complemento ad uno del risultato nel campo.

  I controlli non vengono eseguiti sul flusso dei dati dell'utente, e questo conesente di utilizzare un algoritmo di checksum semplice ma necessità che un protocollo di livello superiore esegua un controllo degli errori sui dati dell'utente.

- **Source Address e Destination Address**(32 bit):
  Contengono gli indirizzi, che possono essere di diverso formato, dell'host sorgente e dell'host destinazione.

- **Options** (lunghezza variabile a multipli di 8 bit):
  È un campo opzionale e esprime informazioni che sono raramente richieste. Inizia con un codice di 1 byte che la identifica e con un campo di 1 byte che indica la lunghezza dell'opzione stessa; poi c'è il cmapo che contiene l'opzione vera e propria. La fine del campo dè delimitata da un byte di 0.

Type (8)	Length (8)	Pointer (8)	Route data (var)

Figure 62: Option

  - **Record Route Option** (RRO): permette al mittente di creare una lista vuota di indirizzi IP; ogni nodo inserisce il suo indirizzo nella lista permettendo così il tracciamento del percorso.
  - **Source Route Option** (SRO): consente al mittente di specificare i nodi attraverso i quali vuole che transiti il datagramma per evitare determinati router o nodi.

- **Padding** (riempitivo):
  Rende la lunghezza dell'intestazione un multiplo intero di 32 bit mediante introduzione di zeri.

10.1.2 Frammentazione e Aggregazione

È importante segnalare che esistono due limiti che impongono restrizioni sulla dimensione dei segmenti:

- Il primo limite è che un segmento (incluso l'header) deve comunque entrare nel payload di un pacchetto IP, e quest'ultimi come detto precedentemente hanno una lunghezza massima di 65535 byte.

- Il secondo limite è che ogni sotto-rete, in TCP/IP, possiede una **unità massima di trasferimento** (**MTU** , *Maximum Transfer Unit*); quindi è proprio l'MTU a stabilire, il limite superiore della dimensione del segmento. Se un segmento attraversa una serie di reti senza essere frammentato e poi arriva a una rete in cui la MTU è inferiore alla dimensione del segmento, è compito del **router di confine** di tale rete frammentare il segmento in due o più segmenti più piccoli (non necessariamente tutti della stessa dimensione) ; in questo caso, ogni nuovo segmento ottiene il proprio header IP, il che fa si che lo spazio totale "sprecato" aumenti, in quanto ogni segmento aggiunge byte di header extra.
  Come viene generato il nuovo Header Ip? L'informazione presente nell'intestazione del datagramma viene "copiata" nell'intestazione di ogni frammento; tranne il campo Total Length che viene modificato con la lunghezza del frammento in questione. Si noti che i campi Options dell'header possono venir copiati o non, o possono venir copiati solo nel primo frammento; inoltre ogni frammento diventa a sua volta un datagramma e può a sua volta essere ulteriormente frammentato.

I datagrammi IP non segmentabili (bit DF=1) non vengono frammentati; ciò comporta la perdita del datagramma se incompatibile con la capacità di trasporto della rete e verrà generato un messaggoo ICMP.

Si ha però che il protocollo IP impone anche che i router devono accettare datagrammi di dimensione pari a quelli delle MTU delle sottoreti interconnesse, che tutti i sistemi (host e router) devono comunque **accettare datagrammi** di dimensioni **almeno** pari a **576 byte** e devono poter **inoltrare datagrammi** di **68 byte** senza ulteriori frammentazione (in caso contrario si ha spreco massimo 60 byte di header al più + 8 byte di frammento, che è il minimo).

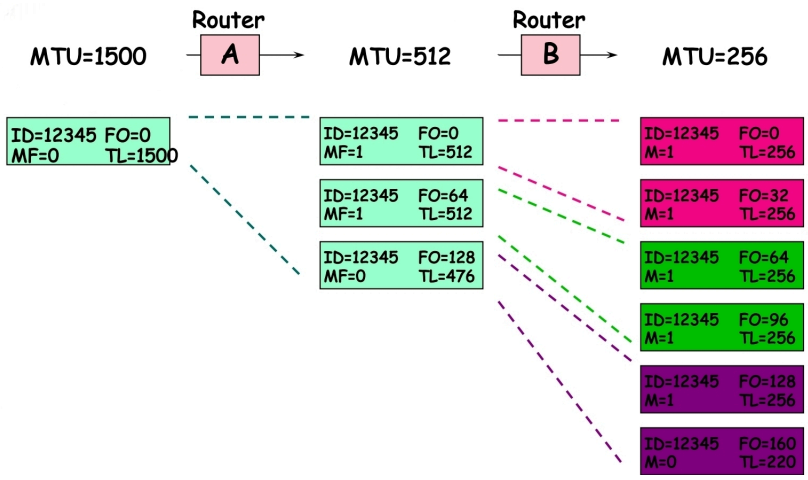


Figure 63: Esempio di frammentazione

Il datagramma originale viene **ricostruito** solo a destinazione che ha il compito di combinare i frammenti vengono con gli stessi valori dei campi *Identification, Source/Destination Address* e *Protocol*; la posizione del frammento all'interno del datagramma è esplicitata dal campo *Fragmente OffSet*. Se per sfortuna uno o più frammenti vengono persi, i restanti che arrivano a destinazione vengono anch'essi scartati (dopo un time-out). (Se si utilizza TCP come protocollo di trasporto all'interno dell'architettura TCP/IP, l'intero datagramma viene ritrasmesso dalla sorgente).

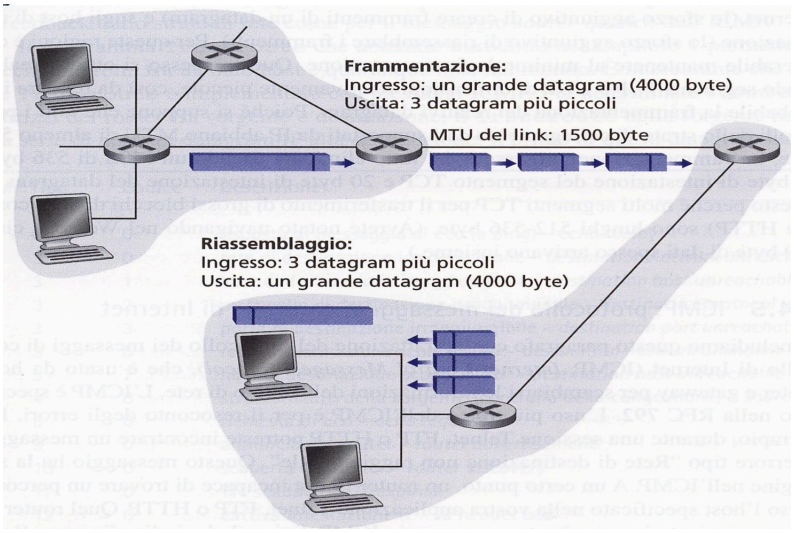


Figure 64: Esempio di frammentazione e aggregazione

Per i più curiosi (Allora con il cazzo che si studia), si osservino i seguenti pseudo-codice della procedura di Aggregazione/Frammentazione:

```
To produce the first fragment:
  (1) Copy the original internet header;
  (2) OldIHL <- IHL; OldTL <- TL; OldFO <- FO;
      OldMF <- MF;
  (3) Number of Fragment Blocks <- (MTU-IHL*4)/8;
  (4) Attach the first NFB*8 data octets;
  (5) Correct the header:
      MF <- 1;  TL <- (IHL*4)+(NFB*8);
      Recompute Checksum;
  (6) Submit this fragment to the next step in datagram
      processing;

To produce the second fragment:
    (7) Selectively copy the internet header (some options are
        not copied);
    (8) Append the remaining data;
    (9) Correct the header:
        IHL <- (((OIHL*4)-(length of options not
                    copied))+3)/4;
        TL <- OTL - NFB*8 - (OIHL-IHL)*4);
        FO <- OFO + NFB;  MF <- OMF;
        Recompute Checksum;
   (10) Submit this fragment to the fragmentation test;

DONE.
```

LEGGENDA:

- FO - Fragment Offset
- IHL - Internet Header Length
- DF - Don't Fragment flag
- MF - More Fragments flag
- TL - Total Length
- OFO - Old Fragment Offset
- OIHL - Old Internet Header Length
- OMF - Old More Fragments flag
- OTL - Old Total Length
- NFB - Number of Fragment Blocks
- MTU - Maximum Transmission Unit

Figure 65: Pseudo-Codice Frammentazione

```
(1) BUFID <- source|destination|protocol|identification;
 (2) IF FO = 0 AND MF = 0
 (3)   THEN IF buffer with BUFID is allocated
 (4)        THEN flush all reassembly for this BUFID;
 (5)      Submit datagram to next step; DONE.
 (6) ELSE IF no buffer with BUFID is allocated
 (7)   THEN allocate reassembly resources with BUFID;
                TIMER <- TLB; TDL <- 0;
 (8)   put data from fragment into data buffer with
       BUFID from octet FO*8 to octet
       (TL-(IHL*4))+FO*8;
 (9)   set RCVBT bits from FO  to FO+((TL-(IHL*4)+7)/8);
 (10)  IF MF = 0 THEN TDL <- TL-(IHL*4)+(FO*8)
 (11)  IF FO = 0 THEN put header in header buffer
 (12)  IF TDL # 0
 (13)     AND all RCVBT bits from 0 to (TDL+7)/8 are set
 (14)  THEN TL <- TDL+(IHL*4)
 (15)      Submit datagram to next step;
 (16)      free all reassembly resources for this BUFID;
       DONE.
 (17)  TIMER <- MAX(TIMER,TTL);
 (18)  give up until next fragment or timer expires;
 (19)  timer expires: flush all reassembly with this BUFID;
       DONE.
```

- FO - Fragment Offset
- IHL - Internet Header Length
- MF - More Fragments flag
- TTL - Time to Live
- NFB - Number of Fragment Blocks
- TL - Total Length
- TDL - Total Data Length
- BUFID - Buffer Identifier
- RCVBT - Fragment Received Bit Table
- TLB - Timer Lower Bound

o (TDL+7)/8 are set

Figure 66: Pseudo-Codice Aggregazione

10.1.3 Indirizzamento

Per consentire a tutti i sistemi connessi a Internet di comunicare è necessario stabilire un metodo globalmente accettato per identificare ed indirizzare in modo univoco tutti i sistemi.
Viene utilizzato un **Indirizzo IP**, unico in tutta la inter-rete, che identifica un sistema (non lo specifico utente ma un host o router utilizzato da più utenti o processi) e lo **schema di indirizzamento in Internet**, definito all'interno degli strati IP e TCP/UDP (Indirizzo Completo caratterizzato da 2 parti, presente in IP e in TCP o UDP); Quest'ultimo è stato progettato per facilitare le operazioni di instradamento, perciò presenta una struttura gerarchica, difatti, un indirizzo IP identifica prima una porzione di inter-rete a cui un sistema è connesso e poi il sistema all'interno di quella porzione.

Si noti che è possibile attribuire indirizzi arbitrari ad una rete TCP/IP solo se questa non è connessa con altre reti o con Internet e inoltre l'identificazione di un utente (in senso OSI) all'interno di un sistema è affidata ai protocolli di strato superiore (TCP o UDP).

Un indirizzo IP è costituito da una sequenza di 32 bit (232 possibili indirizzi), che per semplicità vengono raggruppati in 4 gruppi da 8 bit ciascuno (numero decimale compreso tra 0 e 255 per ogni gruppo).

192.168.150.110

11000000 10101000 10010110 00001010

Ogni indirizzo Ip può essere visto come una coppia di due numeri:

$$IpAddress = NetId.HostId$$

- **Net-id**: identifica una porzione di inter-rete, che può coincidere con una sotto-rete fisica, o comprendere più sotto-reti fisiche o essere un sottoinsieme di una sotto-rete fisica; Indichiamo d'ora in poi la porzione individuata da Net-Id come rete logica.
- **Host-id**: Identifica il sistema (host o router) nella rete logica.

Si noti che la divisione dei bit tra Net-Id e Host-Id non è fissa. Esistoino infatti cinque classi di indirizzi IP:

- **Classe A**:
  Gli indirizzi di questa classe inziano con un bit a 0 (i primi 8 bit hanno un valore decimale compreso tra 1 e 127) e i successivi 7 bit corrispondono al Net-Id, mentre i rimanenti 24 bit corrispondono al numero di host. Allora è possibile avere 128 reti logiche di classe A con 16.777.216 host ciascuna;
- **Classe B**:
  Gli indirizzi di questa classe hanno i primi due bit a 10(i primi 8 bit hanno un valore decimale compreso tra 128 e 191); i successivi 14 bit sono per il Net-Id mentre i rimanenti 16 per l'Host-id. Si hanno cosi 16382 reti di classe B, di 65.534 host ciascuna;
- **Classe C**:
  Questi indirizzi hanno i primi 3 bit posti a 110 (tra 192 e 223 i primi 8 bit), dopodiché i successivi 21 bit sono per il Net-Id e solo gli ultimi 8 bit sono per l'Host-Id. Quindi 2.097.150 reti di classe C, di 254 host ciascuna.

- **Classe D**:
  Questi indirizzi iniziano con quattro bit posti a 1110 (tra 224 e 239 i primi 8 bit) e dopodiché sono caratterizzati da 28 bit. Sono riservati per il cosiddetto multicasting, ossia per fare in modo che un pacchetto IP venga inviato a varie destinazione anziché una sola.

- **Classe E**:
  Primi quattro bit posti a 1111 (primi 8 bit tra 240 e 254 in decimale) e successivamente da 27 bit. Sono riservati per usi futuri e ricerca.

Classe	Bit Iniziali	Net_Id	Host_Id	"reti logiche" disponibili	"indirizzi" disponibili
A	0	7 bit	24 bit	128	16 777 216
B	10	14 bit	16 bit	16 384	65 536
C	110	21 bit	8 bit	2 097 152	256
D	1110	indirizzi multicast: 28 bit indirizzi possibili: 2^32/16=268 435 456			
E	11110	riservati per usi futuri e ricerca: 27 bit indirizzi possibili: 2^32/32=134 217 728			

Figure 67: Tabella Riassuntiva Classi

Esistono altri indirizzi con un significato speciale, di seguito riassunti:

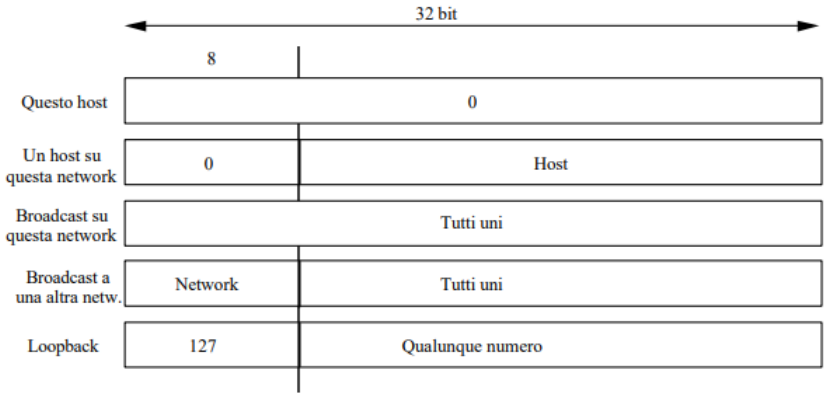


Figure 68: Indirizzi Speciali

Occorre evidenziare che **non sono i nodi** ad avere un indirizzo IP, bensì le interfacce verso le reti. Quindi se un nodo ha N interfacce, esso ha N indirizzi IP. (Ad interfacce di rete distinte appartengono Reti logiche distinte). Inoltre, l'indirizzo IP non identifica l'host in quanto tale, ma la connessione di un host alla relativa rete. Di conseguenza, se un host viene spostato in un'altra rete IP, il suo indirizzo deve essere cambiato.
Al fine di assicurare l'univocità degli indirizzi di Internet è stata costituita un'autorità centrale, l'**InterNIC** che assegna il Net-id; al contrario, l'Host-Id è delegato all'organizzazione richiedente. Cosi un utente che ha bisogno di un indirizzo IP non deve rivolgersi all'INTER-NIC, ma può fare riferimento ad un'autorità locale.

11 Lezione N11 09 11 2021

11.1 Subnetting

La struttura di indirizzamento a due livelli gerarchici era sufficiente nella fase iniziale di Internet (l'interpretazione degli indirizzi è gerarchica, **NON** internet). Ma successivamente alcune organizzazioni, hanno avuto la necessità di gestire più di una sotto-rete, e richiedere un indirizzo diverso, per ogni sotto-rete, avrebbe causato un aumento esponenziale della dimensione delle tabelle di routing. Proprio per tale motivo si decise di rendere più flessibile l'indirizzamento aggiungendo un terzo livello gerarchico: il livello di sotto-rete (Subnet).

Il **Subnetting** permette di sottrarre qualche bit dell'Host-id al fine di ottenere una maggiore flessibilità di configurazione di rete, invisibile però fuori dalla rete stessa. In particolare, il generico indirizzo IP, risulta composto da 3 elementi: un **Net-id**, un **Subnet-id** (che identifica una sottorete nella rete in questione) e un **Host-id**(che identifica un nodo all'interno della sottorete.

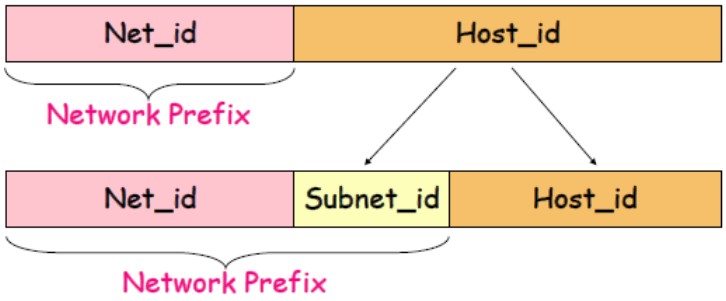


Figure 69: Generico Indirizzo IP

11.1.1 Subnet Mask

Dato un indirizzo IP di un nodo di rete, con tale struttura, esiste un modo immediato per ricavare il numero di rete e il numero della sottorete a cui il nodo appartiene: è sufficiente effettuare un AND, bit a bit, tra l'indirizzo IP e la cosiddetta **subnet mask**.
Una **Subnet Mask** è una parola di 32 bit in cui:

- I bit '1' idnetificano i bit del Net-id e del Subnet-id;
- I bit '0' identificano i bit dell'Host-id;

- *Classe A*: **255.0.0.0**, pari a 11111111.00000000.00000000.00000000
- *Classe B*: **255.255.0.0**, pari a 11111111.11111111.00000000.00000000
- *Classe C*: **255.255.255.0**, pari a 11111111.11111111.11111111.00000000

Figure 70: Valori di Default per le prime tre classi

11.2 Indirizzi Fisici

Ipotizziamo di avere più host, collegati alla medesima rete locale, per comunicare tra di loro hanno la necessità dell'indirizzo IP?
Prima di rispondere a questa domanda, osserviamo che il Protocollo IP entra in gioco quando si ha la necessità di rendere visibili e raggiungibili, i dispositivi all'esterno, poiché permette l'interconnessione tra reti di tipologie diverse.
Quindi la risposta è non necessariamente, perché se appartengono alla stessa rete locale, non si ha la necessità di implementare il Protocollo IP per la comunicazione, ma è sufficiente l'indirizzamento fisico espresso dall'indirizzo MAC. (L'estensione della medesima rete locale LAN può avvenire tramite Hub e Switch). Ricordando si ha:

- **Indirizzi IP a 32-bit**:
  Indirizzi logici di livello rete, che permettono la corretta consegna del pacchetto ad un destinatario collegato alla rete; (Abbiamo già visto come si compongono)

- **Indirizzi LAN o MAC O Fisici**:
  Indirizzi di 48-bit memorizzati nella ROM delle schede di rete, che permettono la trasmissione di un frame da una scheda di rete ad un' altra scheda di rete con cui sussiste un collegamento diretto.
  Si compongono di **due parti** grandi di **3 Byte ciascuna**; **I tre byte più significativi** indicano il lotto di indirizzi acquistato dal costruttore della scheda, detto anche vendor code o OUI (Organization Unique Identifier) assegnato dall'IEEE; **I tre byte meno significativi** sono una numerazione progressiva decisa dal costruttore. Gli indirzzi MAC sono di tre

Figure 71: Struttura Indirizzo Mac

  tipologie: **Single** di una singola stazione, **Multicast** di un gruppo stazioni e **Broadcast**: di tutte le stazioni (ff-ff-ff-ff-ff-ff, f corrisponde ad avere 4 bit a 1 (verificare)).
  Si noti inoltre che ogni scheda, quando riceve un pacchetto, lo passa ai livelli superiori:

  - Broadcast: Sempre.
  - Single: Se il DSAP è uguale a quello hardware della scheda (scritto in ROM) o a quello caricato da software in un apposito buffer.
  - Multicast: se ne è stata abilitata la ricezione via software.

11.3 Protocollo ICMP

Il protocollo **Internet Control Message Protocol**, permette di svolgere funzioni, accessorie e complementari, rispetto a quello che svolge già il Protocollo IP, che risultano estremamente utili. Tra queste vi sono:

1. Verificare lo stato della rete (Echo request, Echo reply).
2. Riportare anomalie (destination unreachable, time exceeded, parameter problem)
3. Identificazione della netmask (mask request, address mask reply)

4. Migliorare il routing (redirect).

Dal punto di vista strutturale, i messaggi, sono individuati da un **Tipo** e da un **Codice** Si osservi

Type	Code	description
0	0	echo reply (ping)

Table 1: Fornitore

che il protocollo ICMP, è utilizzato maggiormente per troubleshooting; difatti con funzioni, come echo reply ed echo request, permette di implementare tool tra i quali troviamo:

- **Ping**:
  Utilizzato per verificare la connettività a livello di rete tra due host, A e B. In particolare l'host A invia un pacchetto "echo request" e alla ricezione di tale messaggio, l'host B risponde con un pacchetto "echo reply".

- **Traceroute**:
  Utilizzato per scoprire il percorso seguito per raggiungere una certa destinazione. In particolare viene inviata una serie di pacchetti con TTL via via crescente,a partire da '1'; il router che, decrementando il TTL, lo azzera invierà indietro un messaggio "time exceeded" e in questo modo si riesce a determinare il percorso fino alla destinazione.

```
C:\Documents and Settings\spromano>tracert 143.225.229.3

Rilevazione instradamento verso grid.grid.unina.it [143.225.229.3]
su un massimo di 30 punti di passaggio:

  1     2 ms     2 ms     2 ms  192.168.2.1
  2     4 ms     8 ms     9 ms  217.9.64.193
  3    11 ms    19 ms    19 ms  192.55.101.129
  4     3 ms     3 ms     3 ms  grid.grid.unina.it [143.225.229.3]

Rilevazione completata.
```

Figure 72: Esempio di Traceroute

11.4 Protocollo ARP

11.4.1 Problema della risoluzione dell'indirizzo

Due host per comunicare direttamente devono necessariamente, essere collegati, alla stessa rete fisica e inoltre per poter scambiare informazioni devono conoscere i rispettivi indirizzi fisici. Invece tramite il Protcollo IP si consente di individuare univocamente un host tramite un indirizzo logico (Indirizzo IP) e questo permette alle applicazioni di usare gli indirizzi logici ed ignorare completamente la rete fisica.
Il problema però è che per inviare un messaggio occorre necessariamente conoscere anche l'indirizzo fisico (Perché? Se conoscessimo solo l'indirizzo fisico, quando la scheda di rete capisce che può processare l'informazione, quindi la decapsula togliendo l'header e passa l'informazione ai livelli superiori, ma il livello superiore, se stiamo utilizzando il protocollo IP, andrà a controllare gli indirizzi logici che non saranno presenti e quindi non saprà come gestire l'informazione "questo esempio fa schifo al cazzo perché serviva sapere perché ho necessità di avere l'indirizzo mac non perché ho necessità di avere l'indirizzo IP anche perché l'indirizzo IP si conosce sicuramente se i due host sono sulla medesima rete locale") Pertanto, serve un meccanismo di corrispondenza tra gli indirizzi logici e indirizzi fisici. Tale meccanismo è offerto dal protocollo ARP.

11.4.2 Funzionamento Protocollo Address Resolution Protocol

Ipotizziamo che un Host A debba spedire un datagramma all' Host B, appartenente alla medesima rete logica (medesima rete IP) e che A conosce solo l'indirizzo IP dell'host B ma non il suo indirizzo fisico.
Allora **l'Host A** manda in **broadcast** a tutti gli host della rete un pacchetto contenente **l'indirizzo di rete di B**, allo scopo di conoscere l'indirizzo fisico di B. L'Host B riconosce il suo indirizzo di rete e risponde all' Host A che finalmente viene a conoscenza dell'indirizzo fisico di B e può spedire il datagramma all'Host B.
Non andiamo nei dettagli del protocollo ma è necessario sapere, che quest'ultimo interagisce direttamente con il livello Data-Link e il pacchetto ARP viene incapsulato in un frame e spedito in broadcast sulla rete e presenta tale struttura:

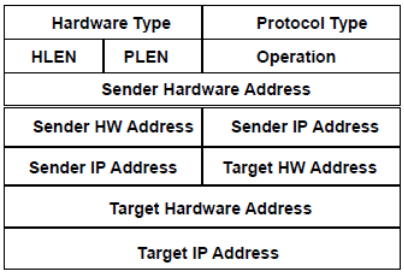


Figure 73: Struttura pacchetto ARP

Esempi del Protocollo ARP su CMD di windows su PDF NOTES.

Ora però quando si effettua l'ARP-request e l'ARP-reply si hanno due casi possibili:

1. Caso: l'host destinazione è sulla stessa LAN(stessa subnet IP)

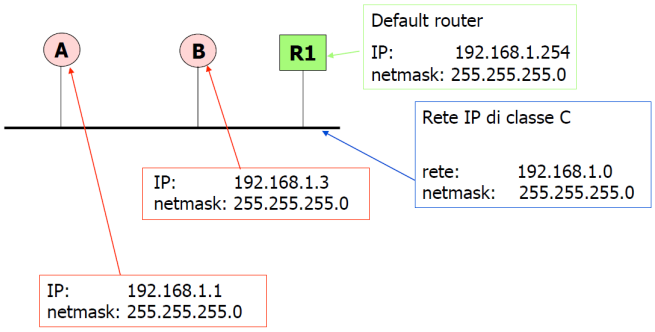


Figure 74: Primo caso

Per verificare se appartengono alla stessa subnet, la sorgente esegue una AND bit a bit, tra l'indirizzo IP destinazione e la propria netmas (tutte informazioni che conosciamo a priori), se appartengono alla stessa subnet siamo nel caso corrente, se no siamo costretti a spostarci nel Secondo caso.

2. Caso: l'host destinazione non è sulla stessa LAN (subnet IP) e quindi si ha la necessità di un nodo intermedio che propaghi ulteriormente la richiesta effettuata.

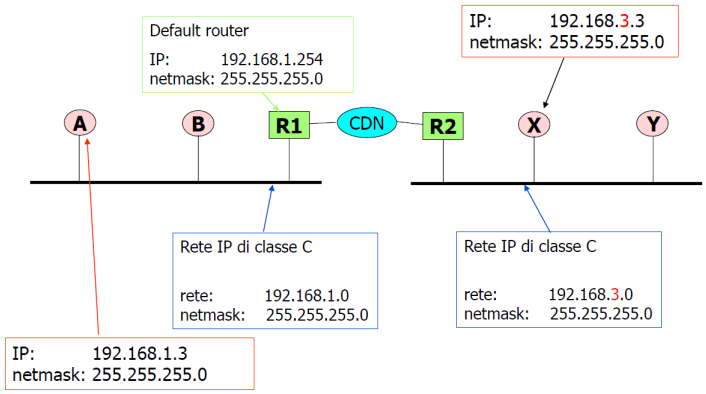


Figure 75: Primo caso

In tal caso è possibile notare, che il problema si scompone in due sotto-problemi, prima si effettua la richiesta ad R1 e successivamente da R1 parte una richiesta per l'indirizzo MAC di X; si noti quindi che il pacchetto deve essere inviato al router di default e nel caso in cui l'indirizzo MAC di esso non sia noto, dovrà essere inviata un'ulteriore richiesta ARP. Difatti per ridurre il traffico sulla rete, ogni host mantiene una cache con le corrispondenze tra indirizzo logici e fisici.

11.5 Protocollo RARP

Il protocollo **Reverse ARP** svolge il ruolo opposto ad ARP, ovvero dall'indirizzo fisico si ottiene quello logico. E tale circostanza occorre quando si hanno dei terminali senza disco e quindi non vi è la possibilità di memorizzare in maniera permanente determinate informazioni (Diskless).

11.5.1 Funzionamento RARP

L'host A conosce il proprio indirizzo MAC, ma non conosce il proprio indirizzo IP. L'host B (server RARP) conosce l'indirizzo IP di A, quindi viene effettuata sulla rete in broadcast una **RARP request** da parte dell'host A e l'host B risponde con un messaggio di **RARP reply** contenente l'indirizzo IP di A.

11.5.2 Altre Soluzioni

Il protocollo RARP è stato sostituito da altri protocolli più flessibili e potenti tra cui il **DHCP**, Dynamic Host Configuration Protocol, che utilizza un server DHCP configurato appositamente per fornire informazioni (non solo indirizzi IP) ai nodi che lo richiedono in cui lo scambio avviene tramite messaggi caratterizzati da una struttura specifica (Ci ritorneremo). Si ha anche il BOOTP (BOOTstrap Protocol).

12 Lezione n12 17 11 2021

12.1 Livello rete (liv 3): Instradamento dei dati

L'instradamento è una funzione di natura logica, che permette di guidare l'informazione di utente verso la destinazione, ovvero permette l'inoltro di pacchetti ai dispositivi intermedi (Permette di estrapolare informazioni riguardanti la sorgente e la destinazione dal pacchetto IP).
Ogni router IP riceve datagrammi IP da un interfaccia, li esamina (In maniera indipendente da quelli ricevuti precedentemente) e li inoltra su un'altra interfaccia; Generalmente ogni router IP presenta un indirizzo IP per ogni interfaccia.

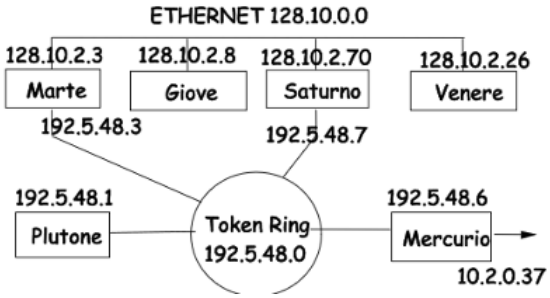


Figure 76: Esempio Router IP

Il livello OSI in cui vengono eseguiti i protocolli di instradamentoe gli algoritmi associati, come anticipato dal titolo, è il terzo (3) ovvero il livello di rete; in particolare, hanno la possibilità di interpretare gli indirizzi IP, poiché quest'ultimi saranno ricevuti in maniera esplicita. Ogni nodo presente nella rete **deve** conoscere l'indirizzo del router.

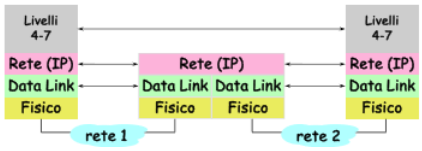


Figure 77: Architettura OSI

Ogni router implenta i 3 livelli (sopra citati), e per tale motivo svolge una duplice funzione; Da un lato riceve i frame dall'interfaccia di ingresso, che vengono **Decapsulati** e inoltrati al "processo" di livello 3 che ha il compito di esaminare Indirizzo Sorgente e Destinazione e prendere decisioni in base a quest'ultimi (Scegliere la prossima Interfaccia di uscita). La seconda funzione è che, scelta la prossima interfaccia di uscita, ha il compito di incapsulare il frame e successivamente inoltrare il flusso di dati ottenuto, al prossimo router o alla destinazione (Se la destinazione è all'interno della medesima sotto-rete).

Una componente importante, è la **Net-id**, poiché ogni router instrada i datagrammi verso la rete logica di destinazione e non verso il singolo host e questo permette di semplificare di molto quelle che sono le tabelle di instradamento. A differenza della componente Host-id, presa in considerazione solo dai router della rete logica di destinazione, la Ned-id viene utilizzata anche dagli **Algoritmi di Instradamento** per determinare la sequenza dei router da attraversare.

Il router effettua due tipi di instradamento IP:

- **Instradamento Diretto**: Si appilca quando la trasmissione di un datagramma IP avviene tra due host connessi alla stessa sotto-rete.

- **Instradamento Indiretto**:
  Si applica quando la trasmissione di un datagramma IP deve attraversare almeno un router, cioè avviene tra due host connessi su differenti sotto-reti.

La decisione tra instradamento diretto e indiretto viene presa in base alla corrispondenza tra l'indirizzo di destinazione e gli indirizzi delle sottoreti a cui il nodo è direttamente connesso.

1 Caso **La sottorete logica coincide con la sottorete fisica**:
l'host mittente confronta la parte del proprio indirizzo corrispondente a **Net-id+Subnet-id** con la parte in questione dell'indirizzo dell'host di destinazione (Se non si usano le subnet il confronto avviene solo con Net-id). (Viene utilizzato instradamento diretto).

2 Caso **La sottorete logica è una parte della sottorete fisica**:
l'host mittente usa **l'instradamento diretto** solo per gli host della **stessa sotto-rete logica** (Altrimenti indiretto); Inoltre si ha che l'host è configurato con l'elenco delle sotto-reti logiche direttamente raggiungibili

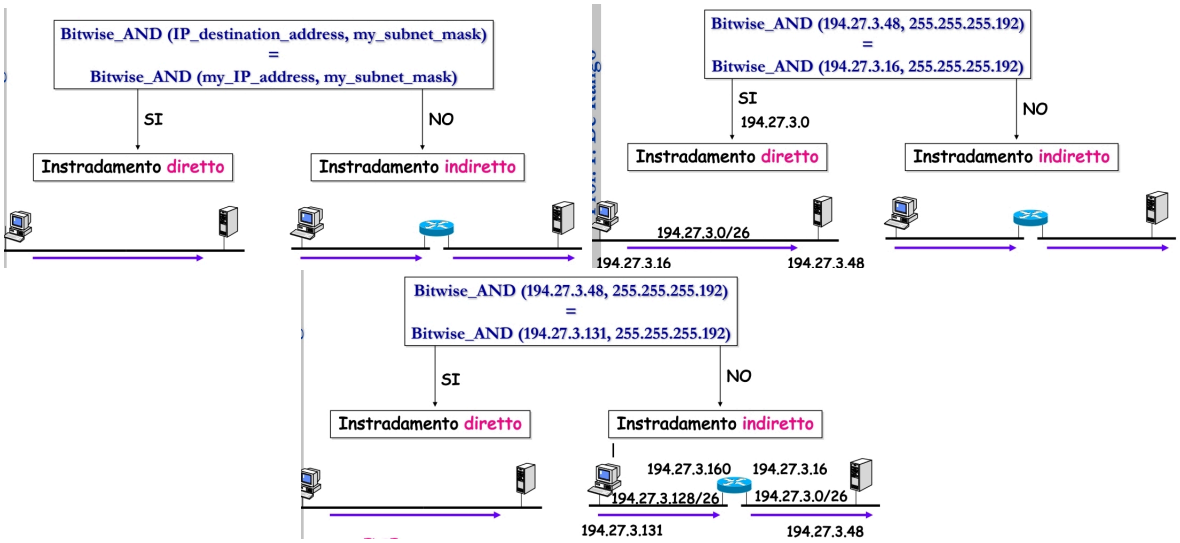


Figure 78: Esempi Instradamento Diretto-Indiretto

12.1.1 Instradamento Diretto

Se un host IP deve consegnare un datagramma ad un altro host attraverso una sottorete comune, l'host sorgente deve conoscere l'indirizzo di sottorete del destinatario (Indirizzo Fisico), altrimenti il datagramma, quando ricevuto, verrebbe scartato al livello due, proprio per una non corrispondenza di tale parametro.
La corrispondenza tra indirizzi IP (indirizzi di livello 3) e gli indirizzi di livello 2 è gestita dal protocollo ARP (Address Resolution Protocol) discusso nella lezione precedente.

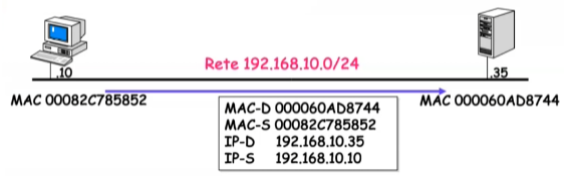


Figure 79: Instradamento Diretto

Attenzione: Ciò non vale solamente nel caso di due host, ma anche nel caso di un host e un router o tra due router diversi.

12.1.2 Instradamento Indiretto

l'instradamento indiretto può essere visto come una concatenazione di instradamenti diretti.

- L'host mittente identifica il router più vicino a cui inviare il datagramma IP utilizzando la sotto-rete fisica a cui è connesso (instradamento diretto)
- il router esamina il datagramma ricevuto e decide verso quale altro router indirizzarlo, nel cammino verso la rete logica di destinazione (Instradamento diretto).
- Nella sotto-rete di destinazione il datagramma viene inviato dal router allo specifico host tramite instradamento diretto.

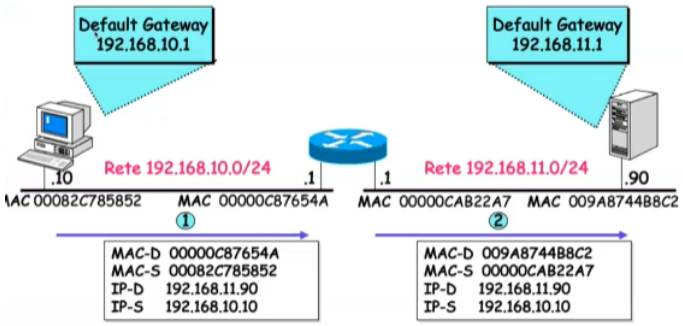


Figure 80: Esempio Instradamento Indiretto

Gli indirizzi di livello 3 non cambiano mai durante il tragitto di un datagramma, mentre quelli di livello 2 individuano gli apparati interessati alla trasmissione del pacchetto all'interno di una particolare sottorete.

12.1.3 Tabelle di Routing

Il meccanismo operativo usato per l'instradamento IP è basato su una tabella di instradamento (routing) che ogni host/ruoter mantiene allo scopo di conoscere le possibili destinazioni e i modi per raggiungerle. Una **Tabella di Instradamento** contiene le coppie N, R, dove N è l'indirizzo della rete di destinazione e R è l'indirizzo IP del prossimo router verso la rete di destinazione. (Solo nella rete di destinazione, R diventa l'indirizzo IP dell'host di destinazione).
La tabella di instradamento, specifica però, solo **un passo** lungo il cammino verso la destinazione; perciò un router non conosce il cammino completo del datagramma ma solo il passo successivo; quindi la tabella contiene indirizzi R raggiungibili solo attraverso una singola sottorete.

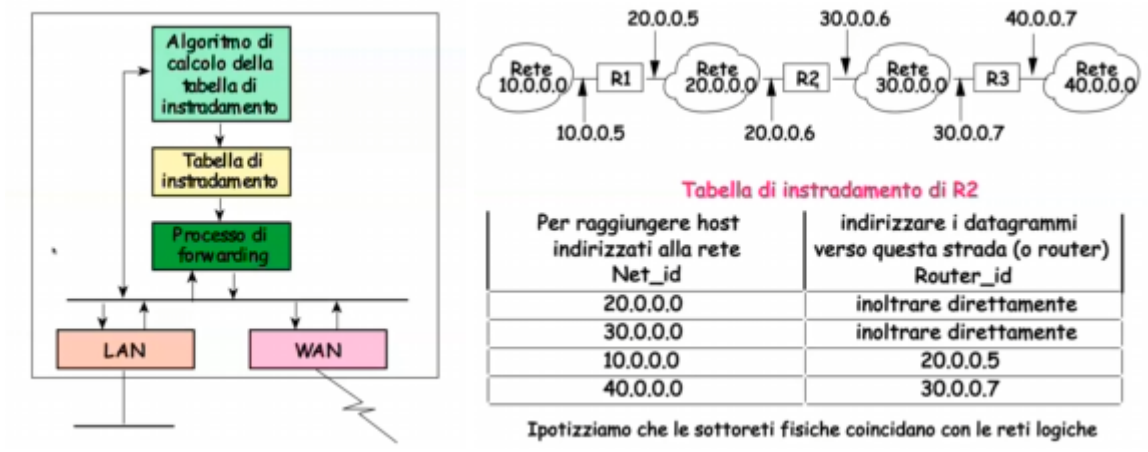


Figure 81: Architettura logica di un router e Esempio Tabella di routing

Le tabelle contengono informazioni, solo riguardanti le reti logiche di destinazione (network prefix, next hop), al fine di nascondere i dettagli inerenti la inter-rete, mantenere piccole le tabelle di instradamento e consentire un instradamento efficiente. Difatti la **Dimensione** della tabella di routing non dipende dal numero di host, ma dal **numero di reti logiche interconnesse**.
Anche se una tabella di routing, può contenere un ulteriore informazione, detta Metrica, che definisce la distanza dalla destinazione, per permettere l'ottimizzazione delle decisioni sull'instradamento, da parte degli algoritmi di routing; Può essere espressa secondo varie unità di misura (distanza fisica, numero di salti, di sotto reti da attraversare o costo del percorso in termini di affidabilità).

La metrica associata ad ogni hop da router a router può essere:

- **inversamente proporzionale alla banda del ramo**.
- **proporzionale al carico istantaneo sul ramo**.
- **proporzionale al costo d'uso del ramo**.
- **qualsiasi combinazione tra i precedenti criteri**.

In generale possiamo dire che il router sceglie il percorso che minimizza il costo (Percorso a costo minimo) in base ad **algoritmi di routing**, ma per determinare il percorso a costo minimo, il router deve avere informazioni sui costi dei percorsi alternativi, acquisite tramite **protocolli di routing**.

Algoritmi e Protocolli **NON** sono la stessa cosa, i primi effettuano un calcolo del percorso date delle info, i secondi definisco la modalità con cui i router scambiano informazioni

Le tabelle di routing possono essere costruite in modo:

- **Statico**:
  Le tabelle sono definite manualmente dal gestore di rete e restano invariate fino al successivo intervento del gestore; vengono utilizzate negli host e nei router di piccola dimensione, in zone della inter-rete relativamente stabili e necessitano dell'utilizzo di criteri fissi di instradamento (Algoritmi Deterministici).
  In particolare un **unico cammino permanente** è individuato tra ogni coppia di nodi origine e destinazione nella rete e la variazione si ha solo nel caso di mutamento della topologia della rete (La scelta del cammino è indipendente dal caricato istantaneo sulla rete).

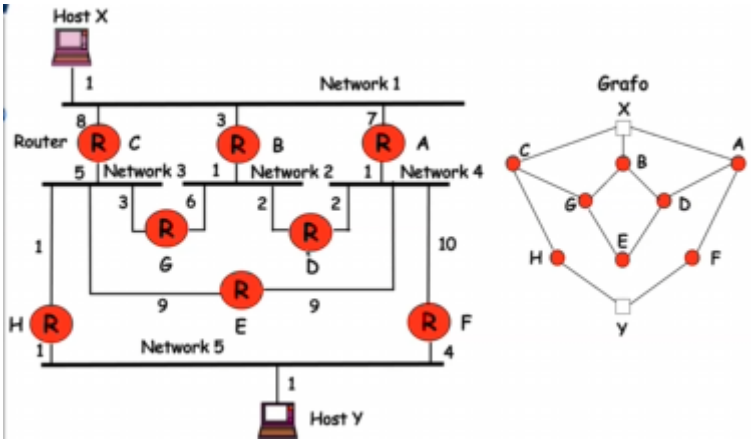


Figure 82: Esempio

Vantaggi: Di facile implementazione
Svantaggi: scarsa flessibilità

- **Dinamico**:
  Le tabelle sono calcolate dinamicamente in funzione della topologia di rete e dello stato dei link (traffico in rete) e necessitano di criteri adattivi di instradamento (Algoritmi Adattivi).
  All'accensione un host/router inizializza la sua tabella di routing usando le informazioni contenute in un dispositivo di memoria di massa o interrogando opportuni server; in seguito ogni router aggiorna o incrementa le informazioni di instradamento dialogando con altri router tramite eventuali protocolli di routing. Tali protocolli sono implementati nei router e si occupano, esaminando la rete a intervalli regolari, di acquisire, appunto, nuove informazioni di instradamento.
  La necessità di un aggiornamento dinamico delle tabelle è dovuto al fatto che internet non è "stabile", difatti nuovi host e sottoreti vengono aggiunti ed eliminati di frequente e molti percorsi diventano disponibili e non, e inoltre si ha che lo stato di occupazione delle risorse di rete varia continuamente.

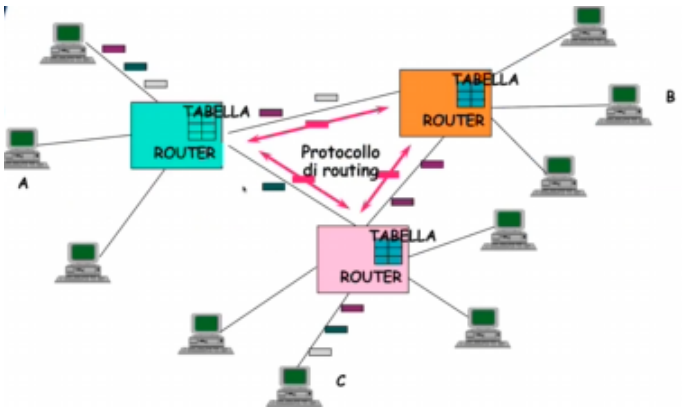


Figure 83: Esempio

Com'è possibile vedere dall'immagine, e come detto in precedenza, i protocolli di routing sono utilizzati dai router per determinare il percorso per aggiungere le reti non direttamente connesse

Vantaggi: Migliore efficienza e controllo della congestione. **Svantaggi**: Decisioni di instradamento complessi da cui consegue un carico elaborativo maggiore nei router, eventuali effetti collaterli come il loop e richiedono scambi di informazioni tra i nodi sullo stato della rete e ciò determina ulteriore traffico addizionale in rete non trascurabile.

12.1.4 Sistemi Autonomi

Si definisce **Sistema Autonomo** una porzione di rete (insieme di host, router e sottoreti) amministrata da un unico gestore. I nodi che instradano messaggi all'interno dell'AS, sono detti **interior router**, mentre queli che istradono messaggi tra AS diversi sono detti **exterior router**. Si osservi che ogni AS è libero di scegliere qualsiasi protocollo di routing da usare tra i router interni al suo sistema autonomo.
Ogni AS sono identificati da un **AS number** assegnato da RIPE o interNIC, univoco a livello mondiale e un AS può contenere uno o più domini di routing (Un sistema autonomo che contiene dei sistemi autonomi più piccoli che interagiscono tra loro).

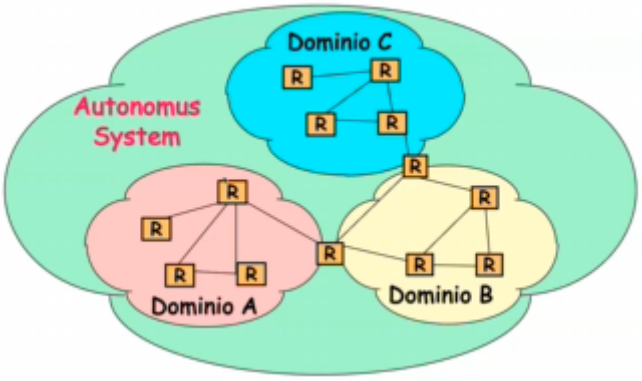


Figure 84: Autonomous System

Generalmente si ha che tutti i nodi presenti all'interno dell'AS devono essere raggiungibili con percorsi interni e per raggiungere un nodo esterno si dovrà passare attraverso router "di confine" **border router** in cu sono implementate politiche di rounting scelte dall'AS. Il ruolo di **border router** è assegnato a uno più router interni che hanno l'obiettivo di informare il mondo esterno della topologia dell'AS, comunicando con altri border router di un altro AS (Anche se caratterizzati da protocolli di routing interni diversi).

Alcuni router all'interno del sistema autonomo vengono chiamati **Core Router**, e contengono le informazioni di instradamento sufficienti per inviare un datagramma verso qualsiasi destinazione, difatti ogni AS deve comunicare loro, quali reti logiche fanno parte del sistema stesso. (Rappresentano la back bone, l'infrastruttura, del Sistema Autonomo).

Le informazioni di instradamento riguardanti i cammini all'interno di un AS e la raccolta di dati da inviare ai core router sono gestiti per mezzo di **Interior Gateway Protocols** (IGP).

Le informazioni di istradamento riguardanti cammini che coinvolgono più di un AS sono gestite mediante **Exterior Gateway Protocols** (EGP).

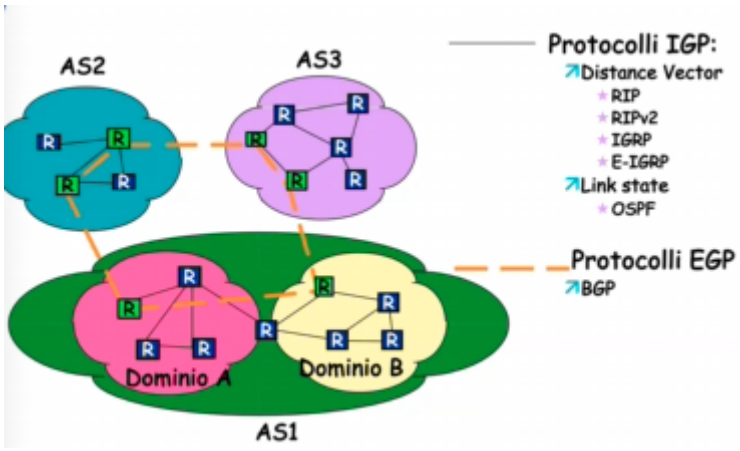


Figure 85: Esempio vari AS

I messaggi di tutti i protocolli di routing sono trasportati all'interno di datagrammi IP.

12.2 Teoria dei Grafi

Un grafo $G < V, E >$ è dato da un insieme V di nodi e un insieme E di archi, ove ogni arco connette una coppia di nodi.

12.2.1 Proprietà dei grafi

- La cardinalità di $|V|$ dell'insieme dei nodi è detta ordine del grafo G.
- La cardinalità di $|E|$ dell'insieme degli archi è detta dimensione del grafo G.
- Due vertici i, j sono detti **adiacenti** se sono connessi da un arco (i, j).
- L'arco i, j è detto **incidente** sui nodi i e j.

- Due archi incidenti sulla stessa coppia di vertici si dicono **paralleli**.
- Un arco incidente su un solo nodo è detto **loop**.
- Un grafo **senza loop** e **senza archi paralleli** è detto **grafo semplice**.
- Un **grafo semplice** è detto **albero** (tree) se:
  - Tra ogni coppia di nodi i e j esiste un unico cammino semplice.
  - Detto N il numero di nodi, il numero di archi è $N - 1$ e il grafo è connesso senza cicli.

  Un qualsiasi nodo dell'albero può essere scelto come nodo radice; quindi l'albero si costruisce disponendo i nodi in livelli successivi a partire dalla radice (livello 0). E si ha che in un **Albero** ogni nodo, tranne la radice ha un solo nodo padre, zero o più figli e se un nodo ha zero figli allora è detto nodo foglia.
- Un grafo può essere rappresentato da una matrice, detta **Matrice di Adiacenza** A di dimensioni $|V| \times |V|$, il cui generico elemento a_{ij} è posto ad 1 se l'arco $(i, j) \in E$ 0 altrimenti. Tale matrice è **simmetrica** rispetto alla **diagonale principale** poiché un arco è definito come una coppia "non ordinata" di nodi, quindi $(i, j) = ((j, i)$.
- **Un cammino** (path) tra due nodi i e j è una sequenza di nodi e archi, a partire dal nodo i fino al nodo j, tale che ogni arco è **incidente** al nodo precedente e successivo. Si osservi che un cammino in cui ogni nodo e ogni ramo appaiono una sola volta è detto **cammino semplice** (Simple path).
- Il minimo numero di rami che compone un cammino tra due nodi i e j è detta **Distanza tra i due nodi**.
- Un **Ciclo** è un cammino in cui il nodo di partenza coincide con il nodo di arrivo.
- Un **grafo orientato** è un grafo in cui gli archi hanno un verso di percorrenza, cioè ogni arco è definito come una coppia ordinata di vertici. (Questo significa che la matrice di adiacenza di un grafo orientato non è necessariamente simmetrica)
- Un **grafo pesato** è un grafo in cui a ciascun arco (i, j) è associato un numero w_{ij} che identifica il peso dell'arco. (La matrice di adiacenza di un grafo pesato contiene i pesi degli archi).
- Si definisce **Lunghezza di un cammino in un grafo pesato** la somma dei pesi associati agli archi del cammino. (Non sempre distanza minima coincide con il cammino a lunghezza minima).
- Un **Sottografo** di un grafo $G < V, E >$ è un grafo ottenuto dal grafo G scegliendo un sottoinsieme di archi e di nodi $\in G$ e per ogni arco scelto devono essere compresi, nel sottoinsieme dei nodi, i nodi in cui l'arco è incidente.
- Un **Sottografo** T di un grafo G è detto **Spanning Tree** di G se:
  - T è un albero;
  - T include tutti i nodi di G;

  Uno **Spanning Tree** di G si ottiene da G rimuovendo gli archi in modo da eliminare i cicli (il sottografo è connesso e non esiste un unico Spanning Tree).

12.2.2 Teoria dei Grafi applicata in Reti

Una qualsiasi rete a pacchetto può essere modellata come un grafo (orientato e pesato) ove i nodi rappresentano router/commutatori e gli archi sono le linee trasmissive.
L'istradamento (routing) di un pacchetto equivale alla ricerca di un cammino nel grafo associato alla rete e si ha una **Ricerca del cammino a minima distanza** nel caso di **Grafo non Pesato** e di **Ricerca del cammino a minima lunghezza** nel caso di **Grafo Pesato**. Quest'ultimo permette di tenere in considerazione parametri come costi, congestioni e capacità.
Quindi in Internet, le decisioni di routing sono equivalenti al problema di trovare un percorso in in grafo, e si basano sul costo minimo:

- Se il criterio è il numero di hop, ove ogni hop/arco vale 1, questo corrisponde a trovare il cammino di distanza minima tra ogni coppia di nodi nel grafo della rete
- Se ad un hop è associato anche un costo (che come detto precedentemente può dipendere dalla capacità, dal carico ecc) questo corrisponde a trovare il cammino di lunghezza minima tra ogni coppia di nodi nel grafo pesato della rete.

Gli algoritimi di routing a costo minimo, più diffusi e basati sul calcolo del minimo spanning tree sono gli algoritmi di Dijkstra e di Bellman-Ford.

13 Lezione n13 24 11 2021

13.1 Algoritmi d'instradamento

L'algoritmo di Dijkstra individua il cammino a lunghezza minima tra un nodo s e tutti gli altri nodi di un grafo G procedendo in modo da aumentare progressivamente la distanza. L'algoritmo procede a passi successivi:

1. Al passo **k-esimo** sono individuati i soli **k nodi** raggiungibili dal nodo sorgente tramite i cammini a costo minimo. Tali k nodi formano **l'Insieme** T.

2. Al passo **k+1-esimo** è aggiunto un ulteriore nodo all'insieme T, caratterizzato dal cammino a costo minimo dal nodo s che transita esclusivamente nei nodi dell'insieme T.

3. L'algoritmo termina quando si sono esplorati tutti i nodi, ovvero $|T| = |N|$ ove N è l'insieme dei nodi del grafo.

```
//Inizializzazione(k=1)
    T={s};
    L(n)=w(s,n) //w(i,j) peso del ramo i j ed L(n) costo del cammino minimo tra il
    nodo s ed il generico nodo n con s!=n

//aggiunta di un nodo(passo K)
-trovare il nodo x !appartenente in T a distanza minima da s, cioè tale che:
    L(x)= min_{j!inT} L(j)
-aggiungere all insieme T il nodo x ed il ramo incidente a x che contribuisce a L(X)

//aggiornamento dei camminimi minimi
    L(n)=min[L(n),L(x)+w(x,n)]
    per tutti i valori di n !appartenenti a T
```

Listing 1: Algoritmo di Dijkstra

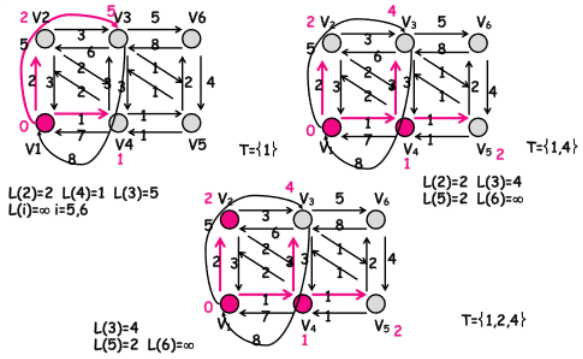


Figure 86: Esempio Algoritmo di Dijkstra

Come è possibile osservare dallo pseudo-codice dell'algoritmo:

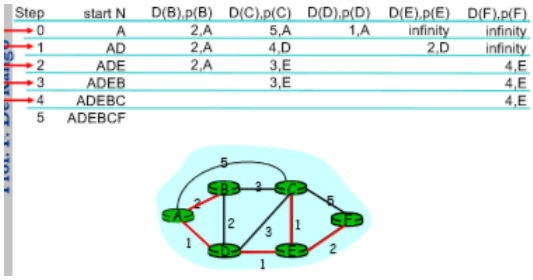
- Al termine L(n) indica il costo del cammino a costo minimo tra i nodi s ed n;

- Al termine l'insieme T è uno spanning tree del grafo di partenza contenente i cammini a costo minimo tra il nodo sorgente e tutti gli altri nodi.

- La complessità dell'algoritmo è $O(V^2)$; difatti intuitivamente ci sono $|V|-1$ iterazioni (A ogni iterazione si aggiunge un nodo in T) e, per ogni iterazione, il numero di operazioni effettuate è proporzionale a $|V|$.

Vi è un altra implementazione dell'algoritmo di Dijkstra caratterizzato dalla presenza del vettore dei predecessori; in particolare con $p(n)$ si indica il predecessore di n, utile al fine di esplicitare attraverso quale nodo raggiungere direttamente l'i-esimo nodo indicato dal vettore e costruire il minimo albero di copertura.

```
Initialization:
    T={s}
    for all nodes n
        if n adjacent to s
        then L(n) = w(s,n)
        elese L(n) = infty
Loop
    find n not in T such that L(n) is a minimum
    add n to T
    update L(k) for all k adjacent to n and not in T:
        L(k) =min(L(k),L(n)+w(n,k))
        /*new cost to k is either old cost to k or known shortest path cost to
            n plus cost from n to k*/
until all nodes in T
```



Step	start N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					

Figure 87: Esempio Dijkstra

Ora la complessità temporale, dati n nodi resta $O(n^2)$ e non è pari a $\theta(n^2)$ poichè esistono ulteriori implementazioni in grado di risolvere il problema in $O(nlogn)$.

13.2 Algoritmo di Bellman-Ford

L'algoritmo di **Bellman-Ford** individua il cammino a lunghezza minima tra un nodo s e tutti gli nodi di un grafo G **Aumentando progressivamente il numero di rami**.

L'algoritmo procede in tal modo:

1. al primo passo individua i cammini minimi tra il nodo sorgente e gli altri nodi con il vincolo che i cammini contengano al più 1 ramo.

2. al secondo passo si trovano i cammini minimi tra il nodo sorgente e gli altri nodi con il vincolo che i cammini contengano al più 2 rami

3. si itera il procedimento sino al valore massimo di rami possibili in un cammino

- **Inizializzazione**

  $L_0(n)=\infty$ per tutti i valori di $n \neq s$

  $L_h(s)=0$ per tutti i valori di h

- **Aggiornamento**

  - per tutti i valori successivi di $h \geq 0$ e per ogni $n \neq s$, calcolare:

$$L_{h+1}(n) = \min_j[L_h(j) + w(j,n)] \qquad (1)$$

  - connettere il nodo n con il nodo predecessore j che raggiunge il minimo ed eliminare le connessioni di n con altri nodi predecessori individuati in precedenti iterazioni

- **Termine**

  - l'algoritmo termina quando all'iterazione h+1 non cambia il valore dei percorsi minimi rispetto all'iterazione precedente, cioè $L_{h+1}(n)=L_h(n)$ per ogni n

Figure 88: Algoritmo di Bellman-Ford

Supposto:

- N: insieme dei nodi del grafo
- s: nodo sorgente
- h: massimo numero di rami in un cammino correntemente consentito dall'algoritmo
- w(i,j): peso(costo) del ramo (i,j)
- $L_h(n)$: costo del cammino a costo minimo tra il nodo s ed il nodo n, con il vincolo che il numero di rami non sia superiore ad h

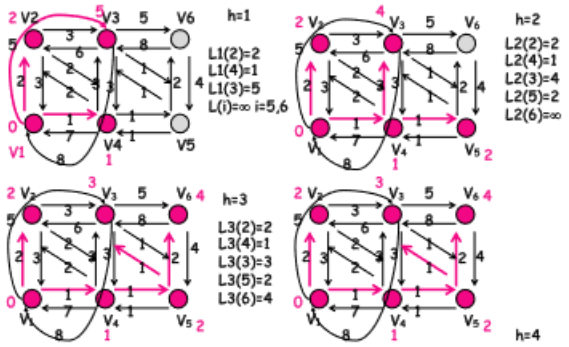


Figure 89: Esempio Algoritmo di Bellman-Ford

Al termine si individua uno **spanning tree** del grafo di partenza che contiene i cammini a costo minimo tra il nodo sorgente e tutti gli altri nodi.
La complessità dell'algoritmo è $O(|V||E|)$, ovvero nel caso peggiore in cui $E = |V|^2$, si ha $O(|V|^3)$ poichè ci sono $|V| - 1$ iterazioni del loop interno e ogni iterazione richiede l'esame dei pesi di ogni ramo.

13.3 Bellman-Ford vs Dijkstra

I due algoritmi (in condizioni statiche di topologia e costi dei link) convergono alla stessa soluzione.

In particolare, L'algoritmo di **Dijkstra** richiede che un nodo conosca l'intera topologia della rete e il peso di tutti i rami e necessità un colloquio tra tutti i nodi.

L'algoritmo di **Bellman-Ford** richiede la conoscenza dello stato dei rami uscenti da un nodo insieme alle informazioni provenienti dai nodi vicini e necessità di un colloquio solo tra i nodi adiacenti (implementazione distribuita)

L'algoritmo di **Bellman-Ford**, nel caso peggiore, è più complesso, anche e su reti di non elevata complessità Ford e Dijkstra si equivalgono.

13.4 Interior Gateway Protocols

Gli interior Gateway Protocols (IGP) sono i protocolli che sono eseguiti all'interno di un **Singolo Sistema Autonomo**. Il modo in cui i protocolli IGP determinano le tabelle di instradamento può essere vario ed utilizzare svariati algoritmi, ognuno dei quali ha i suoi pro e contro.

- **Global information**:
  Tutti i router hanno informazioni complete sulla tipologia della rete e sui costi dei link. Gli algoritmi utilizzati sono di tipo **"Link State"**.

- **Decentralized information**:
  I router in questo caso, hanno informazioni riguardanti solamente i nodi e i costi dei link adiacenti. Gli algoritmi utilizzati si basano sul **Distance Vector** e per reperire informazioni sulla topologia della rete vengono utilizzati processi di computazione iterativi; In particolare si può avere un loop infinito se non si trova una soluzione di metrica a costo minimo (Problema). Performanti su reti di piccole dimensioni.

- **Static**:
  Le rotte Statiche vengono utilizzate se non vi sono continui cambiamenti.

- **Dynamic**:
  Permette di effettuare un ricalcolo delle rotte immediato, nel caso di mutamento della topologia della rete e dei costi dei link. (Aggiornamenti Periodici).

Come riportato precedentemente, la maggior parte dei protocolli IGP si basa su due classi di filosofie di routing:

1. **Distance-Vector Routing**: Viene utilizzato l'algoritmo di Bellman-Ford, da facile implementazione, in grado di impiegare meno risorse sul ruoter con un diminuimento però dell'efficienza.(Adatto a reti piccole).

2. **Link-State Routing**: Viene utilizzato l'algoritmo di Dijkstra, più complesso e più costoso in termini di risorse, però adatto per reti grandi proprio per la sua efficienza (Molto Elevata).

3. **Distance Vector Avanzato**: Il professore di questa filosofia di routing non ha detto quasi niente. Caratterizzato da algoritmi più efficienti che oltre a conteggiare gli hop, calcolano metriche congiunte,

	Algoritmo	Protocollo
Link State	Dijkstra (SPF)	OSPF
Distance Vector Tradizionale	Bellman-Ford	GGP, RIP IGRP (Cisco)
Distance Vector Avanzato	DUAL	EIGRP (Cisco)

Figure 90: Protocolli di routing

Descritti ora in maniera abbastanza semplice, andiamo ad approfondire meglio il discorso.

13.5 Distance Vector

I protocolli di tale classe si basano sull'algoritmo di **Bellma-Ford distribuito**.
Ipotizzando un funzionamento sincrono di tutti i nodi della rete, ogni nodo s ha la possibilità di eseguire, contemporaneamente a tutti gli altri nodi, le seguenti operazioni:

1. calcola $L_1(n)$ con n!=s ed invia il risultato ai nodi adiacenti.
2. calcola $L_2(n)$ con n!=s ed invia il risultato ai nodi adiacenti.
3. ...
4. calcola $L_h(n)$ con n!=s ed invia il risultato ai nodi adiacenti.

Cioè ogni iterazione dell'algoritmo di Bellman-Ford viene eseguita in parallelo da tutti i nodi del grafo (incrementando h di 1) e la procedura ha termine in al più N operazioni.
In realtà i protocolli della classe **DV** utilizzano la versione **asincrona** dell'algoritmo di B-F, poiché la sincronia tra i nodi è difficile da ottenere (quando e come dovremmo avviare l'algoritmo? e come si interrompe l'algoritmo se le condizioni di rete cambiano prima del suo termine?):

- Ogni nodo s esegue in modo asincrono il calcolo:

  $$L(n) = min_{j \in N(s)}[L(j)] + w(j, n)]$$

  usando l'ultima stima di L(j) ricevuta dai nodi adiacenti N(s).
- comunica periodicamente il risultato ai noid adiacenti $N(s)$

L'algoritmo ha termina e restituisce una soluzione se la dinamica delle variazioni è più lenta della velocità di convergenza dell'algoritmo. Si osservi che si potrebbe ottenere una **Bad News Phenomenon** in caso di variazioni particolari da cui ne consegue una convergenza molto lenta e un numero elevato d'iterazioni.

Tali protocolli oltre a restituire e quindi mantenere la tabella d'instradamento, necessità di mantenere un ulteriore struttura dati all'interno dei router, per ogni sua linea attiva, detta **Distance Vector**. Il Distance Vector, contiene informazioni ricavate dalle tabelle d'instradamento, costituita da un insieme di coppie $[Indirizzo, distanza]$ in cui la distanza è espressa generalmente, in numero di hop e costo.

In particolare si ha che tutte le volte che un router modifica la sua tabella di routing, la invia ai router adiacenti(cioè direttamente connessi alla stessa rete) sotto forma di distance vector. Il router che riceverà il DV opererà così:

a Quando riceve un messaggio (distance vector) da un router adiacente confronta ogni coppia (destinazione, costo) con il contenuto della sua tabella di routing e:

- Se al destinazione non è in tabella allora crea una nuova entry per la nuova destinazione.
- Se la destinazione è in tabella e il next hop è il router adiacente che ha fatto l'annuncio allora aggiorna il costo ed il next hop nella entry
- se la destinazione è nella tabella e il costo indica un percorso migliore allora aggiorn il costo e il next hop nella entry.

b A intervalli regolari trasmette ai router adiacenti un messaggio (distance vector) che riporta tutte le coppie (Destinazione, costo) contenute nella tabella di routing.

13.5.1 Struttura Distance Vector

In realtà ogni nodo mantiene al suo interno tre vettori:

- Vettore dei costi uscenti dal nodo

$$W_x = \begin{vmatrix} w(x,1) \\ \vdots \\ w(x,M) \end{vmatrix}$$

- Vettore delle distanze minime verso le altre reti.

$$L_x = \begin{vmatrix} L(x,1) \\ \vdots \\ L(x,N) \end{vmatrix}$$

- Vettore next hop verso le altre reti.

$$R_x = \begin{vmatrix} R(x,1) \\ \vdots \\ R(x,N) \end{vmatrix}$$

- M : numero di reti a cui il nodo x è direttamente connesso
- w(x,i) : costo associato al ramo uscente dal nodo x verso la rete i ($1 \leq i \leq M$)
- N : numero di reti nella configurazione
- L(x,j) : stima della lunghezza del cammino minimo dal nodo x alla rete j (distance vector)
- R(x,j) : next-hop router nel cammino a minima lunghezza dal nodo x alla rete j

Figure 91: Parametri

Ogni nodo applica l'algoritmo di Bellman-Ford distribuito e periodicamente (Generalmente 30s) scambia il proprio Distance Vector L_x con quelli dei nodi vicini. Sulla base delle informazioni contenute in tutti i distance vector ricevuti, il nodo x aggiorna i suoi vettori nel seguente modo:

$$L(x, j = min_{y \in A}[L(y, j) + w(x, N_{xy}]$$

$$R(x, j) = y$$

Ove A è l'insieme dei nodi vicini al nodo x e N_{xy} rete che interconnette i nodi x e y.

Esempi su PDF

13.6 Link-State

Un protocollo di tipo **Link-State** assume che ogni router dispone della mappa completa della rete su cui calcolare gli instradamenti ottimali utilizzando l'algoritmo di Dijkstra; La mappa è costruita direttamente dai router tramite l'utilizzo di **Link State Packet** (LSP).
In particolare si ha che quando un router si attiva, determina il costo dei link su ognuna delle sue interfacce, e tramite protocolli di neighbor greetings apprende la propria situazione locale (riguardanti i nodi adiacenti) e tali informazioni sono inviate a tutti gli altri router della rete tramite LSP.

13.6.1 Struttura LSP

Il LPS generato contiene al suo interno:

- identificativo del router che emette il LSP.
- stato di ogni link connesso al router.
- identità dei nodi adiacenti connessi.
- numero di sequenza per il LSP, utilizzato per individuare diverse versioni emesse dallo stesso router per scegliere la più recente.

Ogni router contiene un LSP database in cui memorizza il LSP più recente generato da ogni router; grazie a questo database ogni router costruisce una mappa della rete con i costi associati. Ma attenzione, la mappa logica della rete, è solo un informazione necessaria e sufficiente per elaborare le tabelle d'instradamento di un router e non è la tabella d'instradamento del router.

Il calcolo della tabella d'istradamento di un router avviene indipendentemente dagli altri router, applicando alla mappa della rete, l'algoritmo di Dijkstra, che trasforma una rete magliata in una

rete ad albero scegliendo i percorsi a più basso costo, e in cui la radice è il router che sta applicando l'algoritmo.

Da questo albero vengono estratte per ogni destinazione il next-hop e il costo totale del percorso. Quindi i router non cooperano per calcolare le tabelle d'instradamento (Come nei protocolli DV) ma cooperano per mantenere la mappa di rete aggiornata, poi ogni router calcola, in modo autonomo, la propria tabella d'instradamento.
In caso di variazioni significative dello stato dei link, il router invia un nuovo LSP in rete.
Un aspetto critico però è la sincronizzazione dei router, se i router hanno diversi LSP database, allora sono possibili dei LOOP.

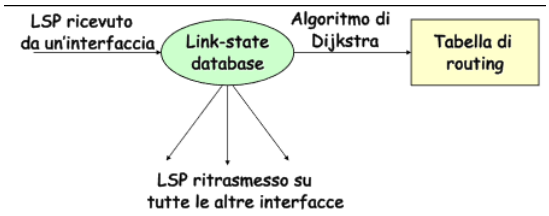


Figure 92: Funzionamento

13.7 Riepilogo

- Distance Vector:
  - Ogni router invia le informazioni di routing ai router adiacenti.
  - l'informazione trasmessa è una stima del costo verso tutte le reti.
  - l'informazione è trasmessa su base periodica regolare
  - un router determina l'informazione sul next-hop usando l'algoritmo di Bellman-Ford distribuito sulle stime dei costi ricevute
- Link-State:
  - ogni router invia le informazioni di routing a tutti gli altri router.
  - L'informazione trasmessa è il valore esatto del costo dei link verso le reti adiacenti
  - l'informazione è trasmessa quando avviene un cambiamento.
  - un router costruisce prima una descrizione della topologia della rete e poi usa un algoritmo di routing qualsiasi per calcolare le informazioni sul next-hop(tipicamente Dijkstra)

14 Lezione n14 01 12 2021

14.1 Lo strato di trasporto

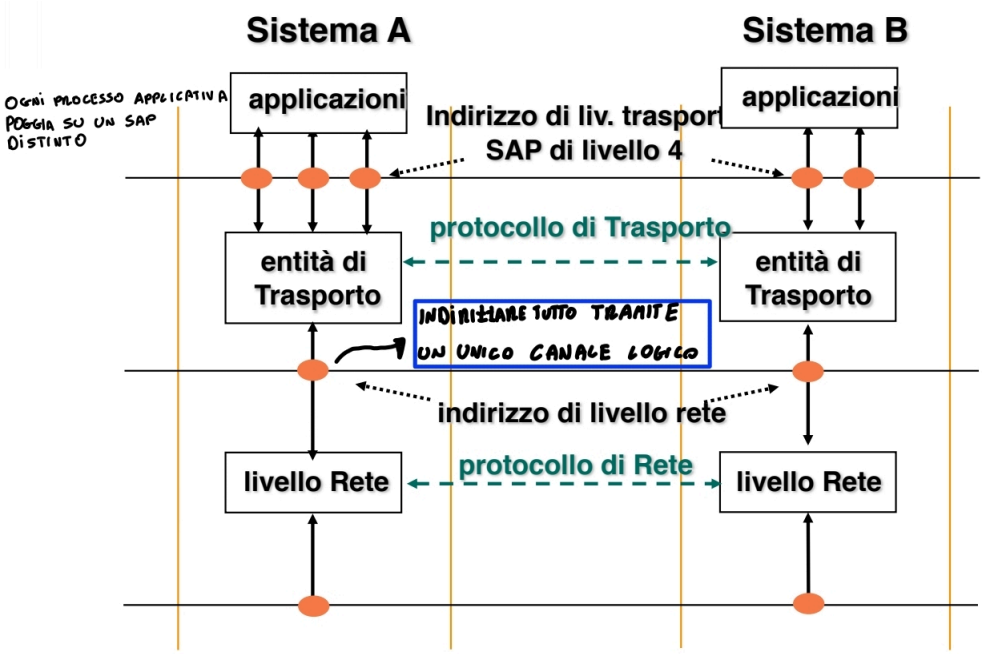
Si osservi che a livello di rete non si è minimamente discusso di processi applicativi e neanche di affidabilità.
I processi sono elementi attivi negli host (identificati come sorgenti e destinazioni di pacchetti, dal punto di vista dell rete) e producono/consumano i messaggi. Terminali, file, dispositivi I/O comunicano tra loro tramite processi. Un processo deve distinguere tra più flussi di comunicazione con altri processi, quindi si assume che ogni processo abbia un numero di porte attraverso le quali comunicare con le porte di altri processi.
Tutta la comunicazione è di tipo inter-processo e fornisce flussi bidirezionali di dati su connessioni logiche tra le porte dei processi coinvolti.

Il servizio di comunicazione fornito dal livello di trasporto, che dipende anche dal livello di rete sottostante, può essere:

- Affidabile con garanzia di consegna dei messaggi nel corretto ordine.
- Non affidabile nel quale viene implementata di fatto la sola funzionalità di indirizzamento.

Nella suite IP, sono definiti due tipi di trasporto **Transimission Control Protocol**(TCP) orientato alla connessione e affidabile e **User Datagram Protocol** (UDP) senza connessione e non affidabile (ma utile nel descriminare i processi applicativi che il livello di rete non riesce a fare).

Si osservi che il livello trasporto nelle reti IP è implementato solo nei sistemi finali, e per svolgere i suoi obiettivi ha come **primo compito** indirizzare i **SAP** su cui vi sono i diversi processi applicativi in esecuzione su un host (es. HTTP, FTP, SMTP). Lo strato di trasporto è responsabile di distinguere, all'interno di uno stesso host, il processo applicativo destinatario (o sorgente) dei dati o i diversi utenti che fanno uso di uno stesso host.



L'indirizzo IP identifica l'host e non gli utenti o i processi associati all'host, difatti l'indirizzato usato dai protocolli di trasporto è il **numero di porta** rappresentato da un intero postivo. I punti

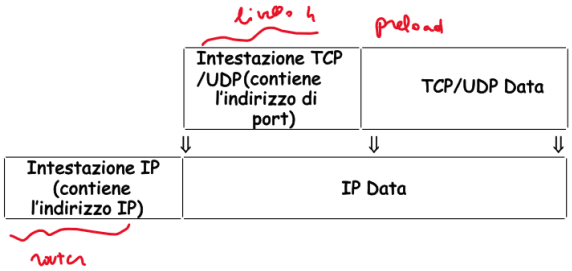


Figure 93: Strato di trasporto

di accesso/destinazione detti "porte" appena introdotte, in terminologia OSI, altro non sono che SAP dello strato 4 tra gli strati di trasporto e applicativo.

Riassumendo si ha:

- Indirizzamento logico di livello 4 avviene tramite le "porte".
- Indirizzamento logico di liveelo 3 avviene tramite gli indirizzi IP
- Indirizzamento Fisico a livello 2 avviene tramite gli indirizzi MAC.

In particolare si ha che **L'indirizzamento completo TCP/IP** è costituito dall'insieme di indirizzo IP e del numero di porta e identifica univocamente un processo in esecuzione su un host. La divisione dei compiti fra lo strato di trasporto (UDP,TCP) e IP è la seguente:

- Lo strato **IP** si occupa del trasferimento dei dati fra elaboratori collegati alle reti interconessse; IP identifica gli host sorgente e destinazione. Ip_Address è contenuta nell'intestazione dell'unità dati di IP.
- Lo strato UDP o TCP si occupa dello smaltimento dei dati fra sorgenti o destinazioni multiple all'interno dello stesso host tramite il numero di porta. La componente "port" è contenuta nell'intestazione del livello trasporto.

14.1.1 Porte TCP e UDP

Riassumendo il numero di porta è il mezzo con cui un programma client indirizza un programma server; più specificatamente per richiedere un certo servizio, fornito da un processo (server) residente su un host remoto, un applicativo client deve aprire una connessione con la macchina di destinazione sulla porta server che individua quel particolare servizio (Es un client FTP per connettersi ad un FTP server deve indicare l'indirizzo IP dell'elaboratore e il numero della porta associata al servizio FTP).
Si distinguono dunque:

- **La porta destinazione** (server port):
  Individua il particolare servizio che l'applicativo client sta richiedendo all'applicativo server. Il valore associato alla server port deve essere univocamente assegnato al servizio ed ha valore globale, ovvero deve essere conosciuto da tutti i client che voglio accedere a tale servizio.
- **La porta sorgente** (client port) permette allo stesso client di elaborare diverse sessioni dello stesso servizio e il suo valore viene scelto dall'applicativo cliente ed è solo locale.

L'assegnamento può avvenire in due modi:

- **Assegnazione Universale**: assegnati dall'IANA in maniera globale e univoca e terminati processi applicativi (well-known ports).
- **Assegnazione dinamica**: assegnati dal processo applicativo sorgente dopo una fase di negoziazione col processo applicativo destinazione e validi per una determinata sessione.

Il numero naturale che individua la porta è di 16 bit e si hanno due gruppi:

- **0 a 1023** è lo spazio riservato per le porte privilegiate o **well known port** per indirizzare un certo servizio.
- **1024 a 65535** è lasciato libero per le porte utenti, cioè quelle scelte dell'applicativo client come porta sorgente.

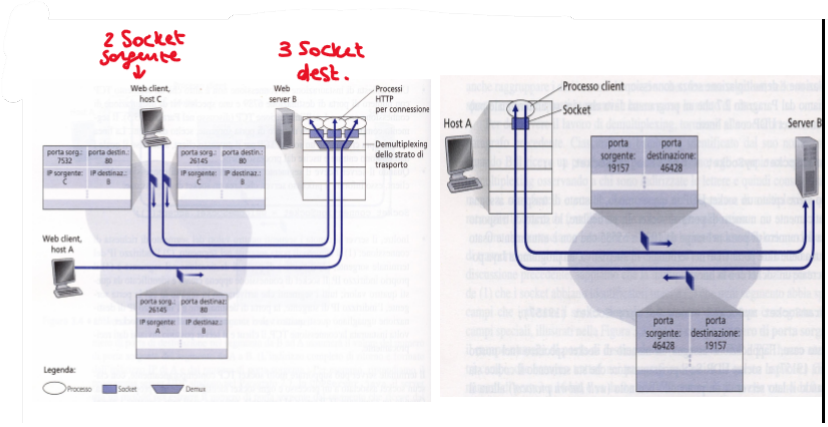


Figure 94: Esempio utilizzo Porte

Servizio	Porta	TCP	UDP
FTP	21	✓	
Telnet	23	✓	
SMTP	25	✓	
TFTP	69		✓
DNS	53	✓	✓
HTTP	80	✓	
SNMP	161		✓

Figure 95: Porte associate agli applicativi principali

14.2 Protocollo UPD

Il protocollo UPD è il protocollo più semplice, in grado di usare il servizio di comunicazione e le funzionalità IP facendo colloquiare processi remodi; in particolare aggiunge due funzionalità a quelle di IP:

- L'indirizzamento delle applicazioni, o meglio, il de/Multiplexing delle informazioni tra le varie applicazioni.
- Una checksum (opzionale) per verificare l'integrità dei dati (blando controllo di errore sull'header dei messaggi).

Tale protocollo fornisce un servizio di tipo **datagram**, che non garantisce la consegna e che non esercita nessun controllo sul flusso e riordinamento delle unità informative [**Connectionless** (pacchetti fuori sequenza), **non affidabile** (pacchetti persi), **senza controllo di flusso** (saturazione ricevitore)].
E non prevede meccanismi di recupero da errore, come ritrasmissioni in caso di errori/perdite e quest'ultimi se necessari vengono demandati (gestiti direttamente) dall'applicazione.

14.2.1 Pacchetto UDP

UDP accetta dal livello superiore dati senza vincoli sulla loro lunghezza, eventualmente li frammenta e li invia in datagrammi IP distinti

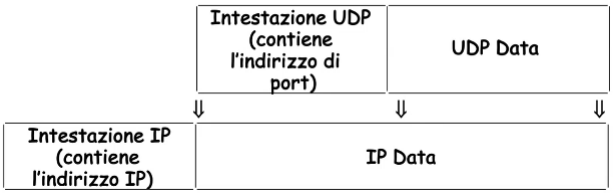


Figure 96: Struttura Pacchetto UDP

Si osservi che l'unità dati UDP (datagramma utente) ha lunghezza variabile, e viene incapsulata in IP ed indirizzata con il campo Protocol pari a 17.
A differenza dell'intestazione TCP di 20 byte l'intestazione di UDP è lunga 8 byte.

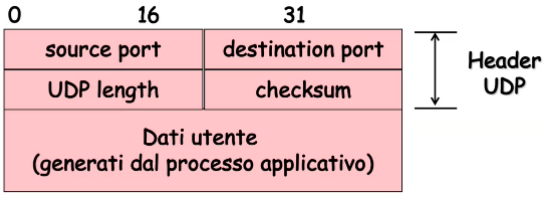


Figure 97: Struttura UDP

- **Port Number** (16 bit) indirizzi delle porte sorgente e destinazione.
- **Length** (16 bit) è la lunghezza in byte del datagramma UDP (header+dati) il minimo valore è di 8 byte quando la parte dati è vuota.
- **Checksum** (16 bit) campo opzionale per il controllo di errore e non si usa principalmente in reti altamente affidabili per ridurre il carico di processamento di un datagramma; Però dati che IP non fa controllo di errore, la checksum può verificare l'integrità dei dati, una volta giunti a destinazione.

14.2.2 Controllo di errore

La checksum copre tutto il datagramma utente UDP(header e dati) e anche il cosiddetto **pseudo-header** che è considerato solo ai fini del calcolo della checksum ma non viene trasmesso a destinazione.
Lo pseudo-header è costituito:

- Da alcuni cami dell'header Ip (32 bit) indirizzi IP sorgente e destinazione (liv3);
- Dal campo Protocol che identifica il protocollo UDP (8 bit e liv 3);
- Lunghezza del datagramma UDP (16 bit);
- 8 bit di Padding per avere una lunghezza totale che è un multiplo di 16 bit.

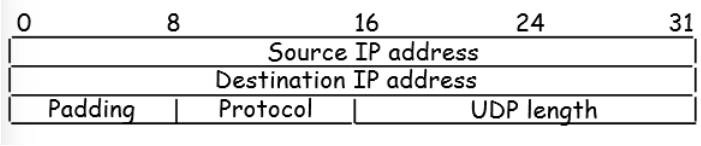


Figure 98: Pseudo-Header

La chechsum applicata allo pseudo-header permette di verificare che il datagramma sia giunto alla destinazione corretta; il solo controllo sull'header UDP non fornirebbe questa garanzia poiché non contiene l'informazione sul sistema di destinazione.

Riflessione: A quanto ho capito, il checksum viene generato prima di effettuare la trasmissione del pacchetto sulla base dello pseudo-header che non viene trasmesso (ma viene trasmesso la checksum) quando arriva a destinazione viene generato nuovamente lo pseudo-header e applicata la checksum (contenuta nel pacchetto UDP arrivato a destinazione)

14.2.3 Congestione

Putroppo per la mancaza di feedback, uno dei problemi maggiormente riscontrati è la congestione; difatti se il ricevitore riceve troppi pacchetti inizia a scartarli, ma non riesce in alcun modo ad avvisare la sorgente, quindi continuerà a scartare pacchetti.

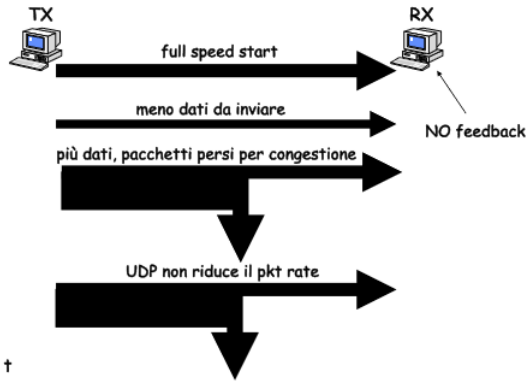


Figure 99: Esempio Congestione

14.3 Transimission Control Protocol (TCP)

TCP è un protocollo "con connessione" che fornisce un servizio affidabile "end-to-end" tra coppie di processi su host remoti ed effettua le seguenti funzioni:

- Indirizzamento di uno specifico utente all'interno di un host (multiplazione e de-multiplazione delle informazioni).
- trasfermento di un flusso informativo continuo e bi-direzionale (il flusso va da sorgente-dest e viceversa) byte stream (orientato ai byte), ma non strutturato, di dati tra host remoti(funzione di segmentation e reassembly).
- controllo e recupero di errori
- gestione delle connessioni
- controllo flusso
- controllo congestione
- riordinamento delle unità informative.

14.3.1 Indirizzamento TCP

Per permettere ai processi in un Host di usare contemporaneamente le funzionalità della comunicazione TCP, quest'ultimo fornice un set d'indirizzi o porte in ogni host (Multiplexing).

L'indirizzo completo TCP/IP, come già abbiamo detto precedentemente è costituito dall'insieme di indirizzo IP e numero di porta (indicato in TCP con il nome di socket) e identifica univocamente un processo in esecuzione su un host.

$$port@IP_{Address} = port@Host_{Id}.Net_{Id}$$

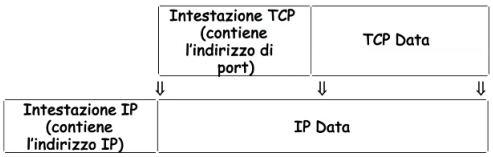


Figure 100: Intestazione TCP

Una connessione TCP è identificata dalla coppia di socket(sorgente e destinazione) associata ai due processi (end-pioint) che hanno stabilito la connessione; Difatti un numero di porta può essere usato per più connessioni, ma l'insieme dei socket di sorgente e destinazione identifica univocamente una connessione (Si osservi che un end-poit può essere impegnato allo stesso tempo in più connessioni TCP, Socket Diverso implica Flusso Diverso).

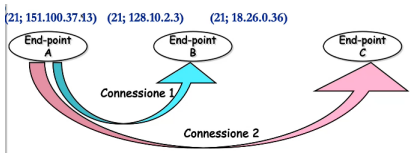


Figure 101: Esempio Connessione TCP

L'assegnazione del numero di porta può avvenire in maniera:

- Statico: Identificativo è staticamente associato all'applicazione e sono utilizzati identificativi inferiori di 255.
- Dinamico: L'identificativo è assegnato direttamente dal sistema operativo al momento dell'apertura della connessione.

In alcuni casi per un migliore utilizzo delle interfacce utente alcuni applicativi effettuano l'indirizzamento automatico delle porte.

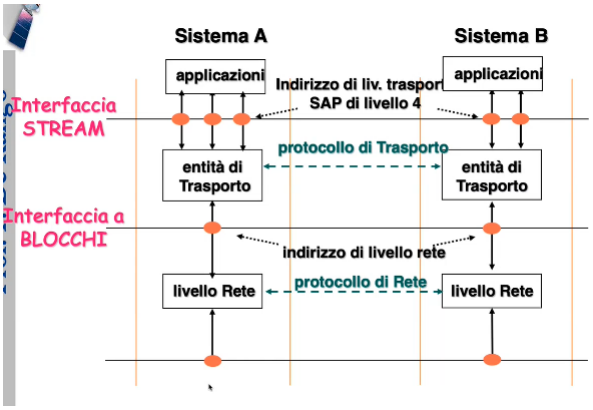


Figure 102: Trasferimento Dati TCP

14.3.2 Formato dell'unità dati TCP

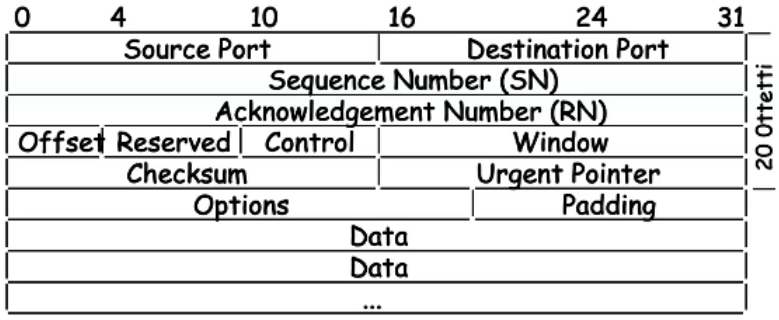


Figure 103: Formato unità dati

L'header è lungo 20byte se il campo opzioni non viene utilizzato. Il campo dati può essere vuoto. Nello specifico si hanno i seguenti campi:

Attenzione: sulle slide si parla anche di altro, ma lui le ha chiamate "chiacchiere" quindi non penso vadano fatte.

- **Source Port** (16 bit): definisce l'indirizzo logico del processo sorgente dei dati.
- **Destination Port**(16 bit): definisce l'indirizzo logico del processo destinatario dei dati

- **Sequence Number**(32 bit): numero di sequenza in trasmissione (SN); contiene il numero di sequenza del primo byte di dati contenuti nel segmento a partire dall'inizio della sessione (se $SN = m$ ed il segmento contiene n byte il prossimo SN sarò pari a $m + n$.

- **Acknowledgement Number** (32 bit): numero di sequenza in ricezione (RN); nei segmenti in cui il bit ACK è a 1, contiene il numero di sequenza del prossimo byteche il ricevitore si aspetta di ricevere. Il meccanismo di riscontro usato è cumulativo, così l'ACK di un SN X indica che sono stati ricevuti tutti i byte fino a X escluso X. In caso di connessioni interattive bidirezionali gli ack sono inviati in piggybacking (nei segmenti di risposta contenenti dati utente) I numeri SN E RN vengono utilizzati per il controllo d'errore e di flusso. (Capiremo meglio come gestirlo quando parleremo di flusso).

- **Offset** (4bit): contiene il numero di parole 32bit contenute nell'intestazione TCP (da Source port a Padding). L'intestazione TCP non supera i 64 byte ed è un multiplo di 32.

- **Reserved** (6 bit): Riservato per usi futuri, contiene zeri.

- **Control bit** (6 bit): i bit di controllo sono 6:

  - **URG**: vale 1 quando il campo Urgent Pointer contiene un valore significativo; è usato per indicare dati urgenti che vengono trasmessi al di fuori del controllo di flusso (le informazioni verranno lette con priorità massima) con un meccanismo di segnalazione end-to-end tra processi remoti. (agisce su lato destinazione)
  - **ACK**: vale 1 quando il campo Acknowledgement Number contiene un valore significativi.
  - **PSH**: vale 1 quando l'applicazione esige che i dati forniti vengano trasmessi e consegnati all'applicazione ricevente prescindendo dal riempimento dei buffer allocati fra l'applicazione e TCP e viceversa (di solito il riempimento dei buffer scandisce la trasmissione e la consegna dei dati).(agisce su lato sorgente)
  - **RST**: vale 1 quando un malfunzionamento impone il reset della connessione
  - **SYN**: indica l'inizio della connessione; vale 1 solo nel primo segmento inviato durante la fase di sincronizzazione fra le entità TCP.
  - **FIN**: indica la fine della connessione: vale 1 quando la sorgente ha esaurito i dati da trasmettere (Utilizzato insieme a SYN per aprire e chiudere una sessione).

- **Window** (16 bit): larghezza della finestra per il controllo di flusso; il TCP ricevente riporta al TCP trasmittente il valore di una "window", che contiene il numero di byte che, a cominciare dal valore del campo Acknowledgement Number, il TCP ricevente è disposto a ricevere. Il controllo di flusso è orientato al byte. (Probabilmente può cambiare in maniera dinamica ma il prof è crashato quindi non si sente).

- **Checksum** (16 bit): contiene la sequenza che permette al TCP ricevente di verificare la correttezza del segmento; protegge l'intero segmento più alcuni campi dell'header IP, cioè uno pseudo-header di 96 bit. (TCP Lenght è la lunghezza del TCP header più la lunghezza dei dati in byte,senza i 12 byte dello psuedo-header e tale quantità non viene trasmessa ma solo calcolata,).

- **Urgent Pointer** (16 bit): indica i dati urgenti all'interno del segmento; contiene il numero di sequenza dell'ultimo byte di dati che devono essere consegnati con urgenza al processo ricevente. Tipicamente sono messaggi di controllo che esulano dalla comunicazione in senso stretto. A tale traffico ci si riferisce di solito con il nome di out-of-band.

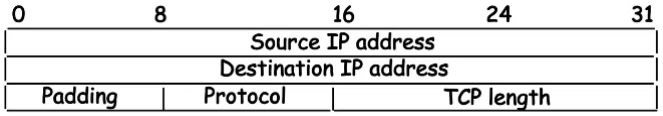


Figure 104: Pseudo-Header nel caso TCP

- **Options**(lunghezza variabile): sono presenti solo raramente; le più nota è Maximum Segment Size che serve per comunicare al TCP trasmittente la dimensione massima del segmento accettabile in ricezione; il campo è inviato soltanto nella richiesta iniziale di connessione (segmenti con bit SYN settato); se non è specificata tale opzione è ammessa qualsiasi dimensione del segmento.

- **Padding**(lunghezza variabile): contiene sempre degli zeri e serve per far si che si abbia una lunghezza totale multipla di 32 bit.

14.3.3 Trasferimento dati TCP

TCP accetta dal livello applicativo un flusso continuo di **dati non strutturati** (byte stream), li frammenta e li invia in unità dati distinte, detti segmenti. È bene osservare che la lunghezza massima dei segmenti, detta **Maximum Segment Size** (MSS), già introdotta in precedenza, viene "negoziata" durante la fase di apertura delle connessioni ed è comunque **inferiore a 64 kbyte** (Visto l'utilizzo di un unico pacchetto IP e dipende anche dall'MTU).

Per la costruzione del segmento a destinazione, il TCP utilizza dei buffer software per immagazzinare i byte dei dati; in particolare vi sono due fasi:

1. Il TCP decide in maniera autonoma quando trasmettere un segmento, anche se è possibile forzare la trasmissione (l'applicazione esegue una funzione di push).

2. Dal lato ricevente il TCP raccoglie i byte in un buffer, il quale quando il buffer è pieno o viene ricevuto un URG flag, trasmette al livello superiore sottoforma di un flusso continuo (L'applicazione assorbe i byte del buffer in base alla sua velocità di elaborazione dell'informazione).

14.3.4 Instaurazione di una connessione

Prima d'iniziare il trasferimento d'informazioni verso un utente remoto, deve essere instaurata una connessione, tramite un meccanismo di sincronizzazione noto come three-way handshake (stretta di mano in 3 fasi).

14.3.5 Enumerazione Segmenti

Prima di vedere nel dettaglio come si instaura precisamente il three-way handshake, è bene precisare in che modo i segmenti vengono numerati nella rete al fine di garantire l'ordinamento.

Per evitare la presenza di segmenti duplicati ritardati (stesso numero di sequenza) si effettua l'implementazione in ogni host di un **clock** (non sincronizzato con gli altri), che incrementa un contatore binario a passi regolari il quale valore è utilizzato per numerare i byte generat i (influenzato

dalla velocità di rete). Si osservi che non deve accadere che nella medesima sessione, si possa creare una **ciclicità nella enumerazione**.

$$\text{frequenza con cui posso numerare i byte } \rightarrow \frac{1}{VMbyte/s} = x\ ms$$

Si osservi che la velocità è in Mbyte/s (Si divide per 8 se la velocità è espressa in Mbit/s).

Riassumendo enumerare i segmenti ci da la possibilità di **creare ordinamento**, capire se un determinato **segmento è obsoleto** e di generare all'interno di una sessione **numeri di segmenti univocamente determinati**.

In termini di **ciclicità delle enumerazioni dei segmenti** ogni volta che si instaura una connessione l'host trasmittente sceglie in modo pseudo-casuale il numero di sequenza iniziale (ISN) di 32 bit da cui partire, ponendolo uguale al valore del suo contatore binario, sincronizzato al clock locale. Lo spazio dei numeri di sequenza risulterà abbastanza ampio evitando la **ciclicità** anche in presenza di un clock abbastanza veloce.

Il numero di sequenza si sceglie in modo pseudo casuale tra 1 e 2^{32}=4.294.967.269

Ma qual è nello specifico la relazione che vi è tra l'enumerazione dei segmenti e il tempo di vita dei segmenti stessi, detto **Maximum Segment Lifetime** (MSL)?

In particolare si vuole che MSL sia minore del tempo di ciclo di clock. La cadenza del clock dipende dalla velocità della rete e quindi ragionevolmente si ottiene che:

$$\text{Ore necessarie per compiere un ciclo completo di valori di ISN: } \frac{1}{V \quad Mbyte/s} \cdot 2^{32} = x \text{ ore}$$

$$MSL << x \text{ ore}$$

Tipicamente MSL è dell'ordine di decine di sec (120sec).

Riassumendo: Ogni segmento occupa uno o più numeri di sequenza nello spazio da 1 a 232; i numeri usati da un segmento sono considerati "occupati" finché passano MSL sec (quite time) per accertarsi che quei segmenti siano scomparsi dalla rete.

14.3.6 Instaurazione di una connessione

Introdotta l'enumerazione dei segmenti possiamo ora parlare del meccanismo di sincronizzazione necessario per l'utilizzo three-way handshake al fine d'instaurare una sessione.

La sincronizzazione tra le due entità TCP, consiste nell'invio del proprio ISN (inserendo nel segmento anche un'informazione sulla massima dimensione MSS del campo dati utente, alla fine verrà scelto quello minore tra i due) e nella ricezione di una conferma dall'altra parte in un messaggio di acknowledgment:

1. $A \rightarrow B$: SYN (Synchoronize) my sequence number is X.
2. $A \leftarrow B$: ACK your sequence number is X.
3. $A \leftarrow B$: SYN (Synchoronize) my sequence number is Y.
4. $A \rightarrow B$: ACK your sequence number is Y.

La procedura è nota come three-way (tre messaggi) handshake proprio perchè i passi 2 e 3 sono combinati in un unico messaggio.

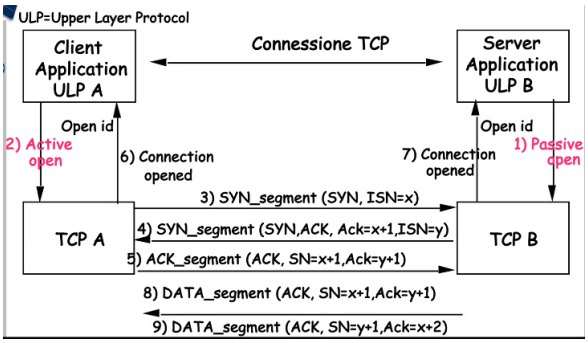


Figure 105: Riepilogo

La scelta del MSS di cui avevamo parlato in precedenza dipende da due fattori:

- La dimensione della memoria a disposizione delle entità TCP
- La dimensione della Maximum Transfer Unit (MTU) della sottorete a cui è connesso l'end-system resa nota all'entità IP e TCP dal driver di rete.
  Il calcolo dell'MSS avviene sottraendo all'MTU la dimensione degli header IP e TCP (in genere le Opzioni non si usano).

Più specificatamente, il TCP trasmittente, nel primo segmento (SYN) se non specifica il valore di MSS si usa il valore di default che è pari a 536 byte; il datagramma IP sarà di dimensione pari a 576 byte (40 byte di Header IP+ TCP) che è una dimensione che tutti i sistemi d'Internet devono necessariamente accettare e gestire (MTU quindi di 576 byte).

Si osservi che per aumentare l'efficienza vi è la tendenza a usare valori più grandi dell'MSS ma questo è possibile effettuarlo con protocolli diversi come IPv6 attraverso la quale si arriva a una MTU di 1280 byte.

14.3.7 Rilascio della connessione

La chiusura di una connessione avviene attraverso i seguenti passi: Il secondo modo di passi avviene

1) A ------> B FIN (ho finito, non ho più dati)
2) A <------ B ACK (OK)
3) A <------ B FIN (ho finito anch'io)
4) A ------> B ACK (OK)

- oppure

1) A ------> B FIN (ho finito, non ho più dati)
2) A <------ B ACK (ho ancora dei dati da trasmettere)
3) A <------ B FIN (ho finito anch'io)
4) A ------> B ACK (OK)

Figure 106: Chiusura Connessione

quando la chiusura avviene in maniera asincrona (A ha finito ma B ancora no). Quindi si finisce di trasmettere e successivamente si chiude la sessione.

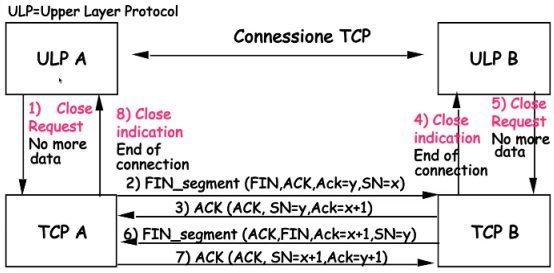


Figure 107: Esempio Chiusura Connessione

14.3.8 Trasferimento Dati parte 2

La strategia utilizzata dal TCP per il trasferimento dati, usa riscontri solo positivi detti **Positive Acknowledge with Retransmission** PAR ed è basata sull'uso di finestre in trasmissione e in ricezione.(Riscontro Cumulativo)

Vi sono anche delle versioni alterate del TCP che utilizzano i NACK ovvero ACK negativi, stile GoBackN, ma ne parleremo in un secondo momento.

La finestra in trasmissione, a differenza di casi precedenti, non lavora a livello di trama o pacchetto ma a livello di byte (TCP vede il flusso di dati in trasmissione come una sequenza di byte); La **dimensione** della finestra in ricezione coincide con quella in trasmissione e ne consegue che specifica sia il numero di byte che un entità TCP può trasmettere senza ricevere riscontri, sia il numero di byte che un'entità TCP può ricevere fuori sequenza.
La dimensione della finestra non è decisa dal mittente, ma è comunicata al mittente dal destinatario "ogni volta" che questo emette un segmento (campo Window 16 bit) e quindi la dimensione può cambiare dinamicamente. La finestra per tal motivo è detta **TCP Advertised Window** (finestra "pubblicizzata" dal destinatario al mittente).

14.3.9 Controllo di Flusso

Il controllo di flusso è una procedura attuata in modo coordinato tra due entità TCP, sorgente e destinatario, intesa a limitare il flusso dei dati trasmessi, in funzione delle risorse a disposizione "nei sistemi terminali" e prescindendo dal traffico presente in rete. (Coinvolge in modo esclusivo solo sorgente e destinazione).
L'obiettivo è di evitare che un mittente invii dei segmenti ad un destinatario che "in quel momento" non è in grado di riceverli (a causa di Risorse o di memorizzazione).

L'implementazione del controllo di flusso nel TCP è simile al meccanismo utilizzato per il controllo d'errore (a finestra scorrevole):

- L'ampiezza della finestra è variabile
- Il meccanismo è orientato al byte: la finestra rappresenta, istante per istante, il numero massimo di byte che possono essere trasmessi verso il destinatario

Quindi lo schema si basa sul campo Windows (16 bit) e il mittente ricevuta la larghezza in byte della finestra utilizzerà tale valore finché non riceve dal destinatario un successivo segmento con dimensione di finestra diversa.

Ipotizziamo di ricevere un riscontro con numero di sequenza pari a x e dimensione di finestra pari a W.

$$ACK_{Number} = x \quad Windows = W$$

Ciò significa che sono riscontrati tutti gli ottetti ricevuti fino a quello numerato con $X - 1$ e che il trasmittente è autorizzato a trasmettere, senza attendere altri riscontri, ulteriori W ottetti (ovvero fino all'ottetto con numero pari a $x + W - 1$).

Solitamente la dimensione della finestra del ricevitore dipende dalla sua capacità di bufferizzare un determinato quantitativo di byte, determinabile con:

$$n \cdot MSS$$

dove n è un numero intero per evitare una segmentazione poco efficiente.

Ipotizzando una finestra in ricezione di 301 byte e la MSS di 100 byte, il TCP trasmittente sarebbe indotto a formare segmenti da 1 byte, a cui aggiungere header TCP e IP (20 byte ciascuno) (inefficienza, effetto indisiderato).

Prima di vedere come risolvere questo caso d'inefficienza analizziamo un primo esempio:

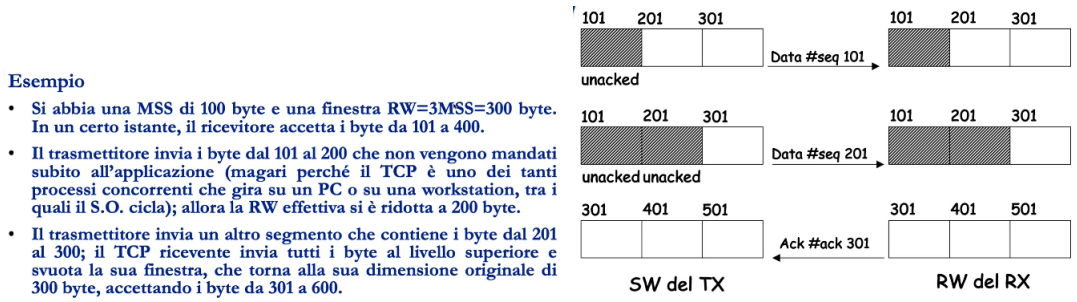


Figure 108: Esempio 1

Altri Esempi su pdf

14.3.10 Sindrome della silly window

La **Silly window syndrome** si riscontra quando l'applicazione sorgente passa i dati al suo TCP a grossi blocchi mentre l'applicazione ricevente assorbe i byte lentamente. (Elevatissimo overhead conseguenza spreco di risorse)

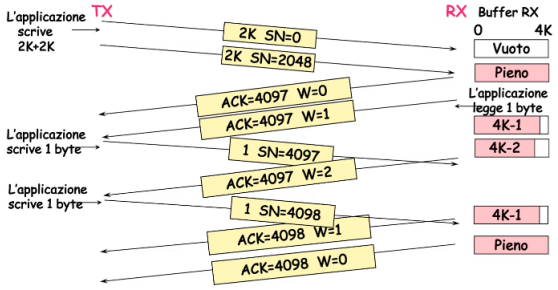


Figure 109: Esempio

Per superare il problema si sono introdotti i seguenti meccanismi:

- Lato ricevitore: Nessun Window Update per piccole variazioni della finestra di ricezione, ovvero aspetta di avere almeno 1 MSS o la metà della finestra di ricezione prima d'inviare un Windows Update (Il ricevitore mente indicando un buffer pieno fino a che il buffer non si è svuotato per metà o per una quantità pari a 1MSS).
- Lato trasmettitore: Il sender deve evitare di emettere segmenti troppo piccoli, ovvero più piccoli di 1/2 MSS a meno che non sia sollecitato tramite un PUSH dei dati.

15 Lezione n15 7 12 2021

A inizio lezione si è visto un video che riassumeva ciò che è stato riportato nei precedenti capitoli

15.1 Continuazione Descrizione TCP

15.1.1 Controllo di flusso: throughput

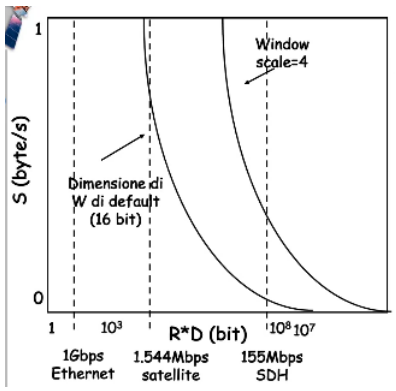
Il throughput S $byte/s$ di una connessione TCP dipende dalla **dimensione della finestra TCP** in byte W , dal **ritardo di propagazione** in sec tra sorgenete e destinazione TCP D e dal **bit rate** in $byte/s$ della sorgente TCP R.

$$S = \begin{cases} 1 \text{ se } W > RD \\ \frac{W}{RD} \text{ se } W < RD \end{cases} \tag{1}$$

Nel calcolo di S viene ignorato l'overhead del segmento TCP e in particolare si ha che:

- Una sorgente TCP emmette una sequenza di byte verso la destinazione; il primo byte impiegherà un tempo D per arrivare a destinazione, più un tempo D per far si che il riscontro torni alla sorgente.
  Durante questo tempo $2D$ la sorgente potrebbe trasmettere $2D * R$ byte, però in realtà la sorgente è limitata dalla dimensione della finestra TCP W finché non riceve alemeno un riscontro.
  Quindi se $W > RD$ il riscontro arriva prima che la finestra si chiuda (si saturi) e si ottiene il throughput max (1) sulla connessione, se invece $W < RD$ (il sender si ferma in attesa del riscontro per la riapertura della finestra W) il max throughput normalizzato è dato dal rapporto $\frac{W}{RD}$.

W è il massimo inviabile in base al protocollo.
RD è il massimo numero di byte/segmenti fisicamente inviabile, che metto sul canale fino a riempirlo tutto, che prescinde dalla log del protocollo.



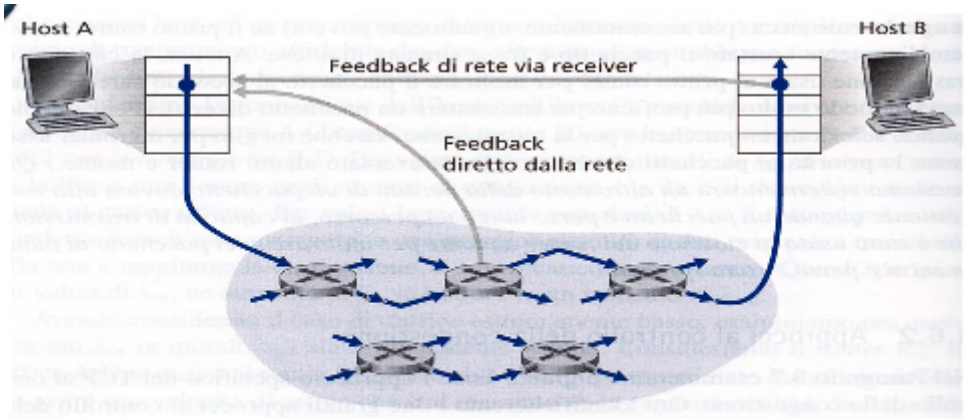
All'inizio si pensava che TCP potesse risolvere, in qualunque circostanza, qualsiasi tipo di problema, però ci si è accorti con il tempo che con l'utilizzo di alcuni canali si ottiene $W << RD$ ovvero il prodotto **Banda Ritardo** è molto elevato, e qualunque sia la tecnica che si cerca di applicare, dal rapporto $\frac{W}{RD}$ si otterrebbe sempre un S molto basso.

Per risolvere tal tipo di problema si è utilizzato un fattore di scala che permette di allargare ulteriormente la finestra.

15.1.2 Controllo di flusso: Window size

La dimensione della finestra di 64KB può essere un problema per alcuni tipi di linee $esHighDelay$, difatti con alti ritardi di propagazione ed elevate velocità di trasmissione, la finestra si svuota velocemente, ma il sender non può inviare altri dati finché non riceve il riscontro.
Si proposta dunque la seguente soluzione:

- **Window scale option**:
  La negoziazione di un fattore di scala che estende la dimensione di Window di altri 16 bit (fino a 2^{32} byte).



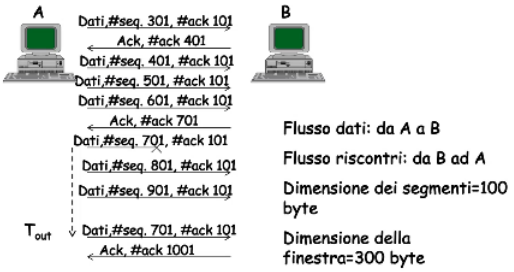
Procedura di controllo della congestione

15.1.3 Controllo e recupero di errore

La strategia utilizzata dal TCP per il recupero di errore è di tipo Go-back-n con riscontri solo positivi (PAR) ed è basata sull'uso di finestre in trasmissione e in ricezione:

- Ogni segmento contiene una checksum che il ricevitore usa per verificare l'integrità dei dati. Se il segmento è corrotto viene semplicemente scartato, dopo un timeout il segmento viene ritrasmesso. Se invece è corretto viene inviato un segmento di acknowledge.

Quindi il **meccanismo di recupero di errore** si basa sull'uso di un **timeout**, la ritrasmissione, da parte del mittente, è quindi innescata dalla mancata ricezione degli ACK entro un tempo predefinito **retransimission timeout** RTO (oltre la quale il segmento è dato per preso).



L'implementazione di tal tipo di meccanismo, necessita di un buffer per la sorgente così che, il segmento inviato ma disperso o corrotto possa essere rinviato allo scadere del timer RTO.

Il **Dimensionamento del timeout** RTO è un aspetto critico per le prestazioni del TCP poiché minore è il timeout, in meno tempo si interrompe la trasmissione:

- Ma **valore troppo piccolo**, i segmenti in ritardo (prima ancora di arrivare), a causa di congestione, potrebbero essere considerati persi e quindi ri-trasmessimi aumentando così la congestione e di conseguenza ulteriormente le ri-trasmissioni.
- Ma **valore troppo grande**, la risposta a un evento di perdita sarebbe troppo lenta con conseguente perdita di efficienza e minore velocità di trasferimento

Difatti il timeout è determinato tramite uno schema "adattivo" in base al valore stimato del **round-trip delay** RTT
L'RTT è il tempo di andata e ritorno tra sorgente e destinazione impiegato da un'unità di dati, ovvero il tempo che passa tra l'invio di un segmento e la ricezione del relativo ACK. Quindi si hanno tre contributi:

- **Ritardo di trasferimento in un verso della comunicazione**;
- **Ritardo di trasferimento nell'altro verso**;
- **Tempo necessario al destinatario per rispondere (tempo di risposta)**;

In generale, la determinazione dei tempi di trasferimento nel RTT è complessa in Internet perché il ritardo di trasferimento è molto variabile e influenzato dallo stato di congestione momentanea delle sottoreti attraversate. Per avere stime più precise del ritardo di trasferimento si avrebbe bisogno di una totale cooperazione tra sottoreti diverse a basso livello, ma ciò è impossibile (Quello che abbiamo fatto fino a mo è astrarci il più possibile per permettere una maggiore interconnettività tra reti, andremmo incontro a tutto ciò che si è detto in precedenza).
Si è trovata quindi una soluzione introducendo una stima del RTT basata su una media pesata dello stesso (media esponenziale) detta **Smoothed Round Trip Estimate** $SRTT$.

1. L'entità TCP memorizza l'istante di partenza di un segmento, in una memoria destinata alla gestione della connessione e inizializza un timer.
2. Alla ricezione di un riscontro valido, il mittente calcola il tempo trascorso, cioè l'RTT corrente e aggiorna il valore stimato di SRTT.

$$SRTT(k+1) = aSRTT(k) + (1-a)RTT(k+1)$$
$$0 \leq a \leq 1$$

Tanto più il valore del peso a (smoothing factor) è vicino a 0, tanto maggiore sarà il peso dato all'ultima osservazione di RTT e inoltre si avrà un maggior aggiornamento del SRTT. Valori maggiori di a invece rendono il SRTT insensibile a brevi variazione del ritardo di trasferimento (Media mobile, la stima è data da quella fatta in passato e da quella corrente, dando un certo peso al passato e al presente).

Si osservi che usando un valore costante di a, e sviluppando l'espressione riportata in precedenza, si può notare che $(1-a) \leq 1$ e di conseguenza ogni termine successivo da un contributo di a sempre più piccolo

Una volta stimato SRTT come calcolo l'RTO? La specifica TCP prevede che il valore di RTO sia proporzionale a SRTT secondo la seguente espressione:

$$RTO(k+1) = min[UBOUND, max[LBOUND, b \cdot SRTT(k+1)]$$

- **LBOUND** e **UBOUND** sono i prefissati limiti inferiore e superiore di RTO (Valore tipico 1s e 1m).
- **delay variance factor** b è una costante il cui valore deve essere scelto tra:

$$1.3 < b < 2.0$$

- Il lower bound dovrebbe essere misurato in frazioni di secondo (per accomodare LAN ad alta velocità) e l'upper bound dovrebbe essere $2 * MSL$, cioè 240 secondi.

15.2 Lo strato di Trasporto: Controllo Della Congestione

15.2.1 Algoritmo di Karn e Jacobson

Ci sono dei problemi legati al calcolo del RTO specificato, in particolare la misura accurata del RTT è difficile quando ci sono ritrasmissioni e l'algoritmo per calcolare SRTT è inadeguato, poiché si assume che la varianza nei valori di RTT sia piccola e costante
Questi problemi sono risolti dagli **algoritmi di Karn e jacobson**

- Nel calcolo di SRTT, l'algoritmo di Karn stabilisce quali campioni considerare nella stima del RTT; in particolare ignora i riscontri dei segmenti ritrasmessi.
- L'algoritmo di Jacobson introduce la misura della varianza del RTT, importantissimo sopratutto quando i data rate sono molto bassi (situazioni durante la quale la variazione delle dimensioni dei pacchetti causa un'ampia variazione di RTT) ideale quindi nelle reti a basso throoughput.

15.2.2 Algoritmo di Jacobson

L'obiettivo è determinare una formula che sia più "generale" rispetto al SRTT, in particolare se non esiste una varianza del RTT nella rete o se esiste è una varianza è molto ridotta possiamo ricadere nel SRTT ma quando la variazione del ritardo di rete inizia a essere considerevole il sistema deve essere capace di adattarsi.
Fondamentalmente si propone di utilizzare il calcolo della deviazione standard (Ipotizzando "n" campioni di RTT e di utilizzare la varianza per misurare la variazione di RTT):

$$s^2 = \frac{1}{n} S|RTT - SRTT|^2$$

La deviazione standard è quindi 1/n (il numero dei campioni) che moltiplica la sommatoria di ogni RTT corrente meno il valore medio del RTT quindi l'SRTT può essere visto come la media campionaria del campione di RTT e quindi si vuole verificare ogni campione corrente come si discosta dalla sua media (questo discostamento lo elevo al quadrato poichè la deviazione standard non entra nei campo dei numeri negativi, evitando effetti che possono tra loro compensarsi).
La stima della deviazione standard di RTT è troppo complessa per essere effettuata su ogni segmento quindi lo stesso Jacobson suggerì di usare la **Deviazione Media** MDEV(RTT) dei campioni del RTT (o errore di predizione medio) come stimatore della deviazione standard di RTT:

$$MDEV(x) = E[|X - E[X]|]$$

$$MDEV^2(RTT) = 1/n(S|RTT - SRTT|)^2$$

Se gli errori di predizione sono distribuiti normalmente, $MDEV$ è una buona approssimazione di SDEV ed è molto più facile da calcolare (Si discosta di solo $1,25$ volte). Una volta calcolata MDEV per determinare RTO Jacobson propose:

$$RTO(k+1) = SRTT(k+1) + u \cdot MDEV(RTT)$$

$$u \text{ fissato solitamente a } 4$$

Tale formula ci permette di dire:

- Se $MDEV(RTT) - > 0$ per lungo tempo, allora c'è poca differenza tra i valori campionati di RTT e la stima del SRTT (SRTT è una buona stima) (ricadiamo nella formula precedente all'introduzione degli algoritmi.)
- D'altra parte, se RTT ha un'alta varianza (MDEV(RTT) grande) allora la stima non è molto affidabile ed è opportuno avere un margine di sicurezza proporzionale all'incertezza.

Un altro modo per determinare SRTT(k+1) è il seguente:

- Introduciamo il calcolo dell'errore di predizione di RTT, SERR(K+1)

$$SERR(K+1) = RTT(K+1) - SRTT(K)$$

- con valori tipici di a pari a $0.1 - 0.2$ avremo:

$$SRTT(k+1) = (1-a)SRTT(k) + aRTT(k+1) =$$

$$= SRTT(k) + a(RTT(k+1) - SRTT(k)) =$$

$$SRTT(k) + aSERR(k+1)$$

- la nuova predizione è basata sulla vecchia più una frazione dell'errore nella predizione.

Successivamente Jacobson propose ulteriormente di usare la tecnica di media esponenziale per la stima di RTT anche per la stima di MDEV, indicata con SDEV quindi:

$$RTO(k+1) = SRTT(k+1) + uSDEV(k+1)$$

I valori consigliati sono $a = 0.125 = \frac{1}{8}$ e $u = 4$.