Appunti di Reti

Settimana 1

Lezione 1

Rete = “insieme di mezzi e utilità che danno servizi”; se riesco ad estrarre un grafo e tracciarlo, ho una rete.
Classificazione di reti: tipo, mezzo di trasmissione, uso, taglia, tipologia.
Tipo: broadcast, canale di comunicazione condiviso.
Multicast: comunicazione possibile a stotogruppi
Point-to-point: connessioni multiple tra coppie di macchine.
LANs,
Wired LANs: ethernet, token rings
Wireless LANs: wifi, wimax
MANs: reti universitarie che connettono LANs diverse.
Tipi: bus, anello, albero, stella
Rete: distribuita su più layers che si parlano tra loro tra servizi e protocolli.

Lezione 2

Design dei layers

- Addressing → gestire mittente e destinatario
- Error control → controllo degli errori
- Flow control → controllare i flussi, impedire sovraccarichi
- Multiplexing → simulare più canali usando un solo canale di comunicazione
- Routing → decide che cammino intraprendere per raggiungere la destinazione

Servizi:

- funzionalità che un layer offre (in concreto tramite un'interfaccia).
- Ce ne sono di orientati alla connessione o connectionless.

2 modelli di riferimento: OSI (Open System Interconnection e TCP/IP

Physical → manda/riceve bit sul media fisico
Data link → incapsula le unità di informazione da trasmettere; trova errori di trasmissione.
Network → consegna dei pacchetti (fornisce dei routing attraverso sistemi intermedi se necessari).
Transport → fornisce il trasferimento dati tra i processi.
Session → stabilisce e mantiene le sessioni di comunicazione da parte dell'applicativo.
Presentazione → si assicura che i dati siano rappresentati nel formato opportuno per macchine diverse; include l'encrypting.
Applicazione → la parte dell'applicativo che gestisce la rete

Vantaggi: architettura a strati, maggiore interoperabilità
Problemi TCP/IP

- modello OSI è più fine; servizi interfacce, protocolli ma non sono nettamente separati.
- Non c'è un modello generico; il modello è stato fatto per retro-fitting.
- No layers physical e data link.
- Il layer host-to-network non è un vero layer.

OSI è fallito, *bad* timing, *bad technology*, *bad* implemaentation, *bad* politics
Modello ibrido usato nella pratica:
5 → Application layer
4 → Transport layer
3 → Network layer
2 → Datalink layer
1 → Physical layer

Strato fisico, bandwidth → capacità del canale
Tipo di trasmissione: wired e wireless
Wired: coppia annodata, cavo coassiale, fibre ottiche

Coppia annodata (UTP, unshielded twisted pair), coppia di fili annodati tra loro; il twist serve a limitare l'interferenza reciproca (crosstalk).
UTP3 → 250MHz
UTP5 → 600 Mhz

Cavo coassiale:

- Schermatura molto migliore dei cavi UTP.
- Molto usato per cavo coassiale e MAN
- Bandwidth: circa 1GHz

Fibre ottiche

- Un raggio di luce dentro ad una fibra con angoli differenti; interno in vetro.
- Quando connetto 2 cavi in fibra ottica si perde però un 10-20% di luce; se li unisco con la fusione perdo 1%
- è dielettrica e ho pochissime interferenze elettriche.
- Molta banda
- più piccola e pesa meno; più difficile da intercettare

Trasmissione wireless

- lo spettro elettromagnetico
- la trasmissione radio
- la trasmissione a microonde
- onde infrarossi
- trasmissioni luminose

Politiche dello specchio elettromagnetico

- Come si assegnano le frequenze?
- Beauty-contest; frequenze a chi fa più del bene
- Lotteria/asta, chi fa l'offerta migliore
- Banda ISM (Industria, Scientifica, Medica): libera

Lezione 3

La trasmissione radio → omnidirezionale → vantaggio: non servono particolari allineamenti tra trasmettitore e ricevente.
Onde radio a bassa frequenza (AM) passano gli ostacoli ma disperdono facilmente. Onde radio ad alta frequenza: non passano bene gli ostacoli, assorbite dalla pioggia.
Nelle bande VLF, LF ed MF, le onde radio seguono la curvatura terrestre.
Nella banda HF, rimbalzano sulla ionosfera.

La trasmissione a microonde

Sopra i 100MHz, le onde viaggiano quasi in linea retta, e quindi possono essere meglio focalizzate e viaggiare distanze più lunghe.
Però svantaggio corrispondente, trasmettitore e ricevente devono essere almeno allineati.
La rete a microonde è stata l'asse portante del sistema telefonico americano per le chiamate a lunga distanza.
MCI = microwave communications inc.
Si usa anche per distribuire il segnale televisivo e telefonico.
Come per le onde ad alta frequenza, le microonde non passano bene gli edifici; sono soggetti a serie interferenze atmosferiche.
Visto il relativo affollamento, ci si spinge a frequenze sempre più alte che però sono ancora più a

rischio di pioggia.
Vantaggio: basso costo → successo di MCI.
Interessante notare: l'altro competitor che è emeros Sprint, non ha seguito le microonde per tutt'altro motivo. Hanno usato la rete ferroviaria già esistente per cablare tutto l'America.

Trasmissioni a Infrarosso e Millimetriche

- usate nei telecomandi
- direzionale
- problema delle microonde amplificato: non passano gli oggetti solidi (aumentando la frequenza, andiamo sempre + verso la luce e sempre meno radio).
- Vantaggi: economiche, maggiore sicurezza

Trasmissioni luminose

- sono state in uso per molto tempo (già nelle battaglie al tempo dei romani).
- Reinterpretate più tardi (1880, il fotofono)
- con l'avvento del laser, erano tornate in auge;
- proprietà del Laser: fascio di luce con poca dispersione, arriva focalizzato a buona distanza.
- Rispetto al fotofono, funziona anche di notte
- Problemi: se c'è pioggia o nebbia, funziona male; ci sono probs anche col sole.

Satelliti di comunicazione

- Satelliti geostazionari (GEO)
- Satelliti medio-orbitali (MEO, medium-earth orbit satellites)
- Satelliti basso-orbitali (LEO, low-earth orbit satellites)
- confronto satelliti-fibra

Notare

- Il compromesso: più basso è il satellite, più satelliti servono (svantaggio), ma...i tempi di latenza diminuiscono come visto nella tabella (vantaggio), e non solo: la potenza richiesta diminuisce molto (la potenza va circa col quadrato dell'altitudine) e inoltre il lancio costa molto meno (lanciarlo a solo 5km costa meno di lanciarlo a 35000km).

Vsat: very small aperture terminals (1 metro).

Satelliti MEO

- orbita media
- “in medio stat virtus...”
- infatti i primi satelliti nella storia sono stati MEO.
- I primi sono stati inviati nel 1957 → Sputnik, primo esempio di comunicazione satellitare perchè trasmetteva.
- Esempio recente di satelliti MEO: il GPS (30 satelliti)
- di chi è? Del DoD (dipartimento della difesa statunitense), Ronald Reagan (1983).
- KAL007, Korean Airlanes volo 007, parte dall'Alaska e deve arrivarea in Korea senza GPS; sbagliano rotto → vengono abbattuti dai Russi; Bill Clinton (2000) → precisione da 100m a 20m.

Satelliti GEO

- inventati dallo scrittore di fantascienza Arthur Clarke nel 1945.
- Per molti aspetti, sono i più appetibili.
- Stanno sull'equatore in orbita circolare
- Limite: massimo 180 satelliti per evitare interferenze
- uso tipico dei GEO: direct broadcast satellite (DSB), televisione via SAT, banda Ku ad alta potenza; esempi: Sky Television (1989), i Free-to-Air (Astra, Hotbird).

Satelliti LEO

- sono i più appetibili, a basso costo
- Iridium, sistema a 77 satelliti; è un sistema unico perchè copre l'intera superficie terrestre poli inclusi. I satelliti sono spesso visibili nelle notti stellate (non sono UFO), lasciando scie luminose chiamate fuochi di Iridio; sono 6 collane di satelliti intorno alla Terra; ogni SAT → 48 celle → 1628 celle mobili che coprono la Terra.

Lezione 4

Iridium

- comunicazione intra-satellite, i satelliti coumicano anche tra loro perchè non sempre ci sono stazioni a terra.
- Lanciato il 1 novembre 1998; la prima chiamata fu fatta da Al Gore.
- Il 13 agosto 1999, il progetto ha dichiarato bancarotta!
- Motivo: costi troppo alti rispetto al GSM, mercato troppo ampio (mondiale), telefoni troppo ingombranti.
- La frase famosa: *"Iridium will succeed because every time we estimated the growth of cellular phone, we were low by a factor of four"* (Bary Bertinger, Motorola).
- Tuttavia: nel 2001 Iridium è stato riacquistato a basso costo ed il servizio riattivato.
- Attualmente, più di 169000 utenti (DoD, marina, aviazione, governi, industrie, petroliere, scienziati, world travelers).
- Inoltre, fra parte del nuovo Tsunami Warning System.
- Data rate di Iridium: molto basso, da 2200 a 3800 baud → richiedere tecniche avanzate di compressione, che vedremo in seguito.

Altro esempio di satelliti LEO: **GlobalStar**.

- Sistema molto diverso da Iridium: serie di "ripetitori bent-pipe", con stazioni gateway nel mezzo; non c'è comunicazione tra satelliti.
- Meno satelliti, 52 (rispetto ai 66 di Iridium); satelliti meno costosi di Iridium (ripetitori), peggio di Iridium, non copre tutta la superficie terrestre, ma in molte aree non necessita di un telefono apposito, avendo un accordo con molti gestori GSM.
- Costi di gestione e facilità d'uso migliori rispetto ad Iridium.

Come si rottama un satellite?

Lo si lascia alla deriva.

Feng Yun 1C, vecchio satellite da rottamare; e la situazione attuale in LEO?

- Più di 44 milioni di detriti nella fascia 0,1-1cm
- 2 milioni di detriti nella fascia 1-10cm
- 34000 detriti più grandi di 10cm

Quel che è peggio...

- Danni non solo in LEO
- Per arrivare a MEO e GEO occorre sempre passare per LEO...
- Pericolo della cosiddetta sindrome di Kessler (effetto a cascata), come una palla di neve che rotola → valanga.
- Risultato: umanità intrappolata.

E quindi, cosa si può fare?

- Occorrerebbe rottamare "ecologicamente" i satelliti.
- Come? Si potrebbe farlo derivare dall'orbita e farlo entrare nell'atmosfera (-->bruciarlo), ma pericoloso per i frammenti; è costoso derivare verso la Terra.
- Paradossalmente, la soluzione migliore (per ora) è invece l'opposta: invece di avvicinarlo all'atmosfera per bruciarlo, si allontana nell'**orbita cimitero**.

E per i satelliti che ci sono già?

- Ogni satellite o frammento (>= 10cm) vieen tracciato.

- Il problema è che sono tantissimi e le misure non sono accurate.

Satelliti vs. Fibre Ottiche

- Storicamente, 20 anni fa, i satelliti sembravano la promessa del futuro per le comunicazioni.
- Nel 1984 introducono il Revised Telecommunication Act e la concorrenza. Di lì a poco seguirà l'Europa.
- Rapido progresso della rete, passaggio al coassiale e alla fibra, abbattimento delle tariffe
- ora è la fibra a vincere sul satellite.
- Ancora peggio per il SAT, perchè l'introduzione delle tecnologie wifi e ibrida (cavo+wifi) ha sfavorito il SAT
- SAT vince ancora per zone poco popolate, reti strategiche e militari, utenti dedicati; ed inoltre per fare broadcasting (Sky, GPS). Vantaggio: è una rete unidirezionale (cioè passiva).
- Ultimo esempio: satellite che non è né LEO, né GEO, né MEO; satellite Molniya, con una proprietà particolare (orbita ellittica, resta fermo su 2 punti).

Le basi teoriche della comunicazione

- Analisi di Fourier
- Segnali bandwidth-limited
- Data rate massimo di un canale

Problema fondamentale: quanta informazione possiamo trasmettere in un canale?

Misure della capacità di trasmissione:

- Misura fisica: la bandwidth (larghezza di banda) → strato fisico
- Misura informativa: data rate (bit rate, bps, baud rate, bauds) → strato più alto

Bandwidth, definizione informale: la capacità; definizione formale: l'insieme di frequenze che trasmettiamo sul canale, si misura in Hertz.

Data rate, informalmente quanta informazione passa in un secondo; più formalmente quanti bit al secondo (bit rate). Baud rate, abbiamo un alfabeto di trasmissione che ha un certo numero di simboli; la misurazione è in funzione di quell'alfabeto.

Esempio: posso trasmettere un impulso usando 4 frequenze: il mio alfabeto è formato da 4 simboli, però ognuno di questi simboli porta l'informazione di 2 bit. Il bit rate sarà il doppio del baud rate.
In generale: qual è l'informazione in bit esprimibile da un alfabeto con V simboli?
Log in base 2 di V
Bit rate = baud rate * log in base 2 di (V)

Materiale: la quantità di informazione dipenderà dal materiale e dal segnale. La potenza per creare trasmissioni ad alta frequenza cresce col quadrato della stessa. Quindi si fissa sempre un certo limite di banda (limite di potenza che impieghiamo).

Settimana 2

Lezione 1

Fourier: più aggiungo armoniche, migliore è l'approssimazione che ottengo.
Con pochi simboli riesco a ricreare la forma d'onda che mi serve; per migliorare la precisione aggiungo qualche simbolo.

Relazione tra data rate e armoniche nel telefono

Bps	**T(msec)**	**First harmonic (Hz)**	**#Harmonic Sent**
300	26,67	37,5	80
600	13,33	75	40
1200	6,67	150	20
2400	3,33	300	10
4800	1,67	600	5

9600	0,83	1200	2
19200	0,42	2400	1
38400	0,21	4800	0

Onda quadrata 400Hz
Onda “a sega” 400Hz (si usano tutte le armoniche, senza saltarne); per certi dispositivi è più facile produrre onde di questo tipo rispetto ad altre.
Harmonic Decay = 1/k^2

Casi più complessi: musica

- Spettro sparse (per Elisa)
- Spettro medium
- Spettro full smooth
- (Beethoven – Ode to joy 4th)
- Differenze armoniche e timbriche che nel tempo, in un genere comune (vecchio e nuovo).
- (Astor Piazzolla – Adiòs Nonino)
- Gotan Project – Vuelvo Al Sur

Attenuazione:
Ogni impulso energetico trasmesso in un mezzo che non sia il vuoto subisce una attenuazione in potenza. (es. stadio con altoparlanti e ascoltatore a 500m di distanza).
Attenuazione in decibel: 10 log in base 10 (Potenza trasmessa / Potenza Ricevuta)
La cosa brutta....
L'attenuazione dipende dalla frequenza...quindi una forma d'onda in generale subisce attenuazioni diverse a seconda delle sue componenti nella trasformata di Fourier.
La **bandwidth** è limitata e dipende fortemente dal mezzo di trasmissione.
Torniamo al problema fondamentale....

- Fissata la bandwidth del canale, c'è un limite massimo alla quantità di informazione che possiamo trasmettere?
- Risposta: Sì

Teorema di Nyquist

- Il data rate massimo (bit al secondo) è:
- 2H log in base 2 di V
- Dove H è la banda massima, e V i livelli del segnale che vengono usati.
- Esempio nel telegrafo (V=2): il data rate massimo è 2H.

Rumore

- Il Teorema di Nyquist vale per un canale ideale dove non ci sono interferenze (rumore).
- In generale però ci sono sempre interferenze (Date dal mezzo di trasmissioen e dall'ambiente)
- c'è anche da considerare la potenza del rumore R.

Segnale e Rumore

- Il rapporto tra la potenza del segnale (S) e la potenza del rumore del canale (N=Noise) si indica con **S/N**.
- Spesso si indica anche in scale logaritmica usando i decibel, definiti come 10 log base 10 (S/N). (simbolicamente è detto l'**SNR**).

Caso limite col rumore?

- Si può generalizzare il Teorema di Nyquist tenendo anche conto del rumore?
- Sì: Teorema di Shannon (-Hartley)
- Il massimo data rate (bit al secondo) è **H log base 2 (1+S/N)**

Approssimazione usando l'SNR in decibel

- Nel caso “ottimo” in cui c'è un altissimo S/N, si ha:
- Data rate max = (circa) H / 3 * SNR
- Attenzione che sono data rate massimi, cioè i massimi fisici; raggiungerli poi in pratica è quasi impossibile.

Altri problemi...

Non c'è solo l'attenuazione, ma anche altri tipi di distorsione delle armoniche.
Dispersione (cambia la forma d'onda) a seconda della frequenza.
Problema MOLTO GRAVE per le lunghe distanze (se mi cambia la forma d'onda, sto distruggendo informazione).

Esempio “sonoro”: onda originale → onda dopo effetto di dispersione.

Soluzione?

Ovviamente, dei ripetitori, che limitino la lunghezza massima di ogni tratto e ritrasmettano il segnale corretto.
Ma c'è di meglio: i SOLITONI, scoperti da Russel Scott (1808-1882); scoperti tempo fa ma permettono di superare la resa incondizionata data dal fattore di dispersione.
Famoso come ingegnere navale, la Great Estern, il Titanic dell'epoca. Famoso anche per l'effetto Doppler, ha fatto la prima misura sperimentale.
Union Canal (Edimburgo); nel 1834 descrive un'onda anomala, che chiama “The Wave of Translation”; e la insegue per chilometri, finchè poi non ce la fa più e la lascia andare; descrive l'onda con disegni vari.

Era il SOLITONE

- Lo ricrea a casa in piccole taniche d'acqua e lo descrive
- All'epoca, 1834, in totale contrasto con le leggi dell'idrodinamica (Newton e Bernoulli).
- Solo oltre 50 anni dopo, nel 1895, si è capito il perchè della sua esistenza.

In Natura...

Oltre che nello Union Canal, si trovano spesso anche come onde normale e come sub-onde. (esempio: stretto di Gibilterra).

I Solitoni,

- hanno praticamente dispersione nulla.
- Morale, funzionano splendidamente nelle fibre ottiche...
- in alcuni test, hanno passato 10000 Km senza dispersione.

Per finire....divertitevi anche voi coi solitoni...

- Diventate l'anima della festa: create un solitone a collisione dimensionale che dura minuti interi....
- Slogan: il solitone che dà soddisfazione
- Vi basta una piscina e buona volontà: immergere un frisbee infilzato da un bastone, in acqua in modo perpendicolare e girarlo, poi ritirarlo fuori.

Lezione 2

Le grandi reti

- Le grandi reti di comunicazione sono al giorno d'oggi varie.
- Cosa vuol dire grande rete? Una rete mondiale
- Di sicuro, la più “antica” è il ***telegrafo. (wired)***.
- Diamo quasi per scontata l'esistenza di una grande rete, ma la strada per arrivarci è stata tutt'altro che facile.

Attenzione che....

- Molti usano la parola telegrafo (dal greco tele= lontano and graphein = scrivere) per ogni

comunicazione di messaggi che non richieda l'uso di lettere, quindi anche prima dell'epoca elettrica.
- Ad esempio quindi anche i ***segnali di fumo*** degli indiani. Esempio il telegrafo ottico (detto semaforo). Usato da Napoleone, grande vantaggio strategico.

Si usava?
- Si, era la rete di comunicazione che permise a Napoleone la suia serie di vittorie fulminee...
- Richiedeva una torre ogni 20km
- Data rate: circa **2 parole al minuto**
- Restò attiva in Francia per oltre 50 anno, dal 1792 al 1846, e fu usata anche in Europa e negli USA.

Telegrafo “wired”
- Vari inventori, dal 1775: l'ultimo fu Morse (1837) che però fu il primo ad effettuare trasmissioni con ripetitori (per via del problema dell'attenuazione).

Fax
- Il fax (trasmissione di immagini), un “overlay” (sopra-strato) sopra al telegrafo wired, fu inventato più tardi: **Alexander Bain** (1843) e, indipendentemente, un monaco italiano, **Giovanni Caselli** (1855) che fu il primo a proporlo commercialmente (linea Parigi-Lione).

L'espansione del telegrafo
- Finora c'è sempre stato il vincolo territoriale.
- Il vero salto di qualità verso una rete di comunicazione mondiale avviene quando si cerca di superare gli oceani, con il progetto dei cavi sottomarini e l'idea rivoluzionaria del cavo transatlantico.

Sott'acqua
- I cavi erano sempre, stati sulla terraferma, ma la necessità inglese di comunicare con l'Europa porta alla ricerca di tecnologie per cavi sottomarini, culminando nel primo cavo telegrafico tra Inghilterra e Francia nel 1850.

Frederick Newton Gisborne
- L'anno successivo, 1851, Gisborne si procura fondi americani e fonda una compagnia per collegare parti dell'America del Nord con cavi subacquei.
- Nel 1853 ***fallisce*** (…!)
- Nel 1853 si incontra con Cyrus Field (occhio pazzo), che ha l'idea visionaria di passare un cavo addirittura sotto l'Atlantico.

Field e il suo pazzo sogno...
- Entusiasta dell'idea, Field fonda una compagnia apposita, la New Yotk, Newfoundland.....

Il primo tentativo
- Con una cavo composta da sette cavi in rame coperti da 3 strati di gomma speciale, coperti da un fascio annodato di fili di ferro.
- Peso complessivo: circa 640 Kg al Km.
- Seia il Governo Inglese che quello Americano (per un voto!) approvano il progetto e danno dei finanziamenti.

1857
- Comincia lo srotolamento del cavo, usando 2 navi da guerra riadattate
- Il primo giorno il cavo si rompe, viene recuperato, e “riannodato”.
- Dopo qiualche giorno, si rompe di nuovo, ma in quella che ora viene chiamata “la fossa del telefrafo” profonda 3200m
- → il cavo è perso.

Il secondo tentativo...
- L?anno dopo (1858), si riprova, con un'altra tattica: dividere il cavo a metà, partendo con una nave dall'America e con l'altra dall'Europa ed incontrandosi nel mezzo.

- Il cavo si rompe dopo 5km, riannodato; poi dopo 100km, riannodato; poi dopo 370km...e qui rinuniciano.

Il terzo tentativo

- Un mese dopo, ci riprovano
- Questa volta, il cavo non si rompe, e in sei giorni viene srotolato tutto: il 5 agosto 1858 il cavo viene collegato. (nasce la prima grande rete dell'umanità).
- Il 16 agosto avviene il primo messaggio telegrafico trans.oceanico, tra la regina Vittoria e il presidente Buchanan.
- L'entusiasmo salì alle stelle :-)

1858, data storica per le reti

Data rate?

- Il data rate è di 0,1 parole al minuto (un carattere ogni 2 minuti)
- Il primo msg di saluto del 1858 (poche righe) ci mise 17 ore per essere trasmesso.

Due settimane dopo

- Wildman Whitehouse, il capo-elettricista del progetto, ha la brillante idea di cercare di aumentare la velocità di trasmissione aumentando il voltaggio (overclock).
- Risultato: il cavo va in sovraccarico e si rompe (!!!).

E allora?

- Molti insinuano che non sia mai stato collegato, e che sia stato annunciato solo per salvare la società di Field dal crollo in borsa (antenata della Parmalat?)
- E infatti, la società fallisce!

Sei anni dopo...

- Ci vogliono altri 6 anni (1864) perchè Field, che non si è perso d'animo, trovi i capitali necessari per rifondare un'altra compagnia
- Stavolta però, licenzia Wildman Whitehouse (l'overclockaro)

Nel frattempo

- Il progresso tencologico porta al design di un nuovo cavo:
- 7 cavi di rame di alta qualità, uniti da un composto dielettrico, circondati da 4 strati di gomma speciale alternatid a1 anltro composto dielettrico, poi iuta, 18 cavi di acciaio sottile, coperti da tessuto resistente di Manila.
- Peso complessivo: il doppio del vecchio cavo.

Il quarto tentativo...

- L'anno successivo (1865) il cavo è pronto:
- 14260km di cavo vengono ammatassati su una nave, la Great Eastern , che comincia poi il lento srotolamento sull'Atlantico (15 luglio).

16 giorni dopo...

- Il 31 luglio 1865, dopo aver srotolato 1968km, il cavo si rompe e si perde nelle profondità, la compagnia di Field fallisce nuovamente.

L'anno successivo, 1866, il quinto tentativo

- Field ha la testa dura, e rifonda un'altra compagnia per riprovarci
- Nel 13 luglioi 1866 la Great Eastern ci riprovarci
- 2 settimane dopo, la nave tocca terra, il 27 luglio 1866

La mattina dopo....

- I presidenti si scambiano gli auguri

12 gg dopo

- La Great Eastern riparte per un'altra missione: cercare di recuperare il cavo che si era perso nel quarto tentativo.
- L'opinizione pubblica pensa siano matti...
- Con un'ancora a uncino, setacciano per giorni la zona dove si era inabissato il cavo a 4km di

profondità.

Due settimane dopo...

- Agganciano il cavo
- Grandi festeggiamenti sulla nave!
- Ma il cavo non è agganciato perfettamente all'ancora, e mentre cercano di portarlo a bordo, si stacca e si inabissa nuovamente!!

Dopo altre 2 settimane...

- …riescono nuovamente a riagganciare il cavo.
- Stavolta stanno molto più attenti, e pian pianino cercano di portare il cavo sulla nave.
- Ci mettono 26 ore, ma alla fine ce la fanno.

Il giorno dopo...

- Annodano il vecchio cavo con un nuovo raccordo, e continuano lo srotolamento.
- Pochi giorni dopo, il 7 settembre 1866 toccano terra: ci sono ora non una ma due linee sull'Atlantico.

Data rate

- Col primo cavo del 1858, il data rate era 0,1 parole al minuto
- Col nuovo cavo del 1866, il data rate è aumentato a 8 parole al minuto.
- Solo nel ventesimo secolo il data rate è stato poi aumentato a 120 parole al minuto, cambiando tipo di cavo.
- Sono poi stati introdotti ripetitori per amplificare il segnale lungo la via.

Un anno dopo, 1892

- Thomas Edison inventa il telegrafo a due vie (Duplex Telegraph); fino a quel momento tutte le comunicazioni erano half duplex, cioè a una via.

Altri otto anni dopo...

- Nel 1902 viene passato un cavo anche sotto al Pacifico, e quindi il telegrafo per la prima volta fa il giro del mondo.

54 anni dopo...

- Nel 1956, viene installato TAT-1, il primo cavo telefonico transatlantico, supportando 36 canali telefonici.

32 anni dopo

- Nel 1988, viene installato TAT-8, il primo cavo telefonico transatlantico in fibra ottica.

E in Italia?

- Ci sono cavi sottomarini, e dove? Il lago di Como e paesi vicini.

La seconda Grande Rete

- Il sistema telefonico, sviluppato inizialmente come un overlay sopra alla grande rete telegrafica.
- Tanenbaum, come altri, attribuisce l'invenzione a Bell (poche ore prima del rivale Gray), nel 1876. BUUH!
- Antonio Meucci, Johan Philipp Reis importanti

Cronologia

- 1849: Meucci dimostra un prototipo del telefono all'Avana
- 1854: Meucci dimostra il telefono elettrico a New York
- 1860-1861: Reis dimostra un prototipo del telefono elettrico
- 1871: Meucci ottiene un caveat perchè il brevetto costa $250 che lui non ha.
- 1872: Meucci mostra il suo telefono al vicepresidente della American Distric Telegraph Co di New York lasciandogli copia del caveat.
- 1872-73: Meucci rinnova il caveat, continuando a chiedere supporto a Grant
- 1874: Meucci chiede indietro a Grant il caveat, ma Grant non glielo ridà dicendo di averlo perso.

- 1874: Meucci non ha i soldi per rinnovare il caveat ($10).
- 1872: il telefono di Reis viene dimostrato a NY
- 1874: Gray dimostra il suo prototipo di telefono musicale
- 1876: Bell deposita il brevetto del telefono poche ore prima di Gray.
- Chi ha inventato il telefono secondo voi? Bell?!

Eppure---

- Disinformazione pazzesca, anche su Internet, dovuta al fatto che Bell era americano mentre Meucci e Reis europei (guardate un po' in giro e fatevi un'idea imparziale...!)

Lo stesso caso Elisha Grey vs. Alexander Bell

- Bell deposita il suo brevetto poche ore prima di Grey
- In realtà, Grey deposita un caveat circa 2 ore prima di Bell.
- Solamente, il brevetto di Bell viene registrato prima (ma fa fede l'orario di consegna).
- Il punto vero è che poi Grey ha rinunciato al suo caveat.
- Bell riusa idee di Grey per costruire il suo telefono.
- Certe parti del brevetto sono praticamente prese dal caveat di Grey ma senza le figure.
- Perchè Gray rinuncia al suo caveat?
- Lo consiglia il suo avvocato che lavorava per la Bell company.

Telefono all'inizio (1876)

- Ogni linea era (point-to-point) fra coppie di telefoni.
- Risultato non molto conveniente; per parlare da una casa all'altra, bisognava tirare un cavo per collegarle.

Il passaggio successivo (1878)

- Gli switching center (centralini)
- Centralini, dove fisicamente un operatore collegava il vostro cavo con quello dell'altro utente desiderato.

Man mano che la rete cresceva...

- I centralini avevano troppe connessioni, e quindi furono creati centralini di secondo livello per connettere i centralini di primo livello.

E via via...

- ...man mano che la rete cresceva, ad un certo momento conveniva salire ancora di livello gerarchico
- si è arrivati fino al quinto livello di centralini
- La rete telefonica così ottenuta si chiama **PSTN (Public Switched Telephone Network).**

Situazione schematica

Il “**local loop**” è anche chiamato più familiarmente “**ultimo miglio**”, tipicamente composto da ***UTP categoria 3***.

Parentesi sulla liberalizzazione

- Abbiamo già accennato tempo fa che nel 1984 gli Stati Uniti compiono una grande liberalizzazione.
- Spezzano il monopolista telefonico, **AT&T** in oltre **23 compagnie.**
- Sancendo inoltre il diritto ad ogni nuova compagnia di essere inserita nell'infrastruttura telefonica nei gradi più bassi (vicini all'utente).

Ancora più competizione:

- Nel 1995 permettono la commistione tra compagnie telefoniche, tv via cavo, telefonini.
- Nel 1996 sanciscono il diritto alla portabilità del numero.
- 1996...confrontate con l'Italia in leggerissimo ritardo...

Torniamo all'infrastruttura telefonica

- come si trasmettono i dati?
- Tipicamente, ci si è convertiti al digitale
- Tranne (nella maggioranza dei casi) che per il local loop, l'ultimo miglio. Per passare da analog a digitale e viceversa, serve un modem (modulatore-demodulatore).

Problema: corrente continua o alternata?

- Le onde digitale hanno solitamente un amplio spettro di frequenze, e quindi presentano seri problmi di attenuazione e distorsione
- si è scelta la corrente alternata (dà meno probs di attenuazione e distorsione).

Come si trasmette il segnale digitale?

- Essenzialmente tre modi di base:
  - Modulando in ampiezza
  - Modulando in frequenza
  - Modulando in fase

Modulazione in ampiezza
Modulazione di frequenza, detta anche frequency shift keying
Modulazione in fase, detta phase shift keying; a cambiamenti di simboli, corrisponde un cambio di fase.

Andiamo oltre: modulazione di fase

- Possiamo usare vari “sfasamenti” in ogni singolo impulso, in modo da avere un alfabeto di simboli più capiente e quindi, a parità di **baud**, aumentare il **bit rate**.

QPSK

- Quando si usano 4 sfasamenti (tipicamente, i simmetrici 45°, 135°. 225°, 315°)
- ...si ha un alfabeto di quattro simboli
- Il bit rate è doppio rispetto ai baud
- Questa tecnica si chiama QPSK (Quadrature Phase Shift Keying)

Spingiamo l'acceleratore?

- Allora, potremmo ad esempio aumentare il numero di sfasamenti possibili nella modulazione di fase
- Man mano che aumentiamo, aument...
- Problema: se cambiassimo solo la fase, avremmo poi che le differenze di fase diventano molto piccole e quindi poco distinguibili, con tutti i rischi che seguono...
- Situazione del tutto simile, se ricordate, ai satelliti GEO.

Lezione 3

Problema

- Se cambiassimo solo la fase, avremmo poi che le differenze di fase diventano molt

Approccio migliore

- Combinare più tipi di modulazione assieme: “due gusti is meglio che one”
- Un buon approccio è tenere la frequenza al massimo (cosa che semplifica di molto la generazione del segnale e la sincronizzazione), e mescolare insieme le modulazioni in ampiezza e fase.

Ripensando ai satelliti

- Equivale in un certo senso ad aver sia più satelliti nella stessa orbita (modulazione di fase), però anche spaziandoli in altitudine (modulazione di ampiezza): in questo modo ci stanno molti più satelliti (→ segnali).

Constellation diagrams

- Diagramma per il QPSK
- Mescoliamo allora fase ed ampiezza: QAM-16, (QAM: Quadrature Amplitude Modulation)

- Spingiamo ancora: QAM-64

Confronto tra QPSK, QAM-16 e QAM-64

- QPSK: 4 simboli, bitrate **doppio** rispetto ai baud
- QAM-16: 16 simboli, bitrate quadruplo rispetto ai baud, due volte meglio del QPSK
- QAM-64: 64 simboli, bitrate sestuplo rispetto ai baud, tre volte meglio del QPSK.

Nota...qualcuno se lo sarà chiesto...

- Le scelte della combinazione ampliezza/fase nei QPSK, QAM-16 e QAM-64 sono le migliori?

Risposta: NO!

- Un po'.....

I QAM ottimali

- Sono i “circular QAM”
- Esempio di “QAM-8” ottimale

In generale, trovare il QAM ottimale è tutt'altro che banale

- Esempio: il seguente QAM.16 circolare è molto meglio di quello rettangolare, ma non è ottimale.

In pratica...

- Si usano i QAM rettangolari invece di quelli circolari e/o ottimali perchè sono più facili sia da generare che da decodificare

Torniamo alla linea telefonica...

- Se si calcola il limite fisico di Shannon della linea telefonica tra due utenti che usano il modem, si ottiene:
- 35000 bps circa (!), circa 35k
- → il meglio che un modem.....

Cosa usano i modem? Standards principali in uso:

- V.21 (1964): modulazione di frequenza, 300bps
- V.22 (1980): modulazione di fase, 1200bps
- V.22bis (1984): QAM-16, 2400bps
- V.32 (1984): QAM-32 con error correction (9600bps)

Modem in pratica

- V.32bis (1991): complicato, 128 simboli, 14400bps
- V.34 (1994): complicato, 12 bit per simbolo (28800bps)
- V.34bis (1996): complicato, 14 bit per simbolo (33600bps) che è il limite fisico
- Siamo praticamente al limite fisico 35000!

Ma allora...

- Com'è che i modem che conosciamo noi vanno a 56kbps?

Il limite fisico della linea telefonica

- Per evitare interferenze di vario tipo ed avere un suono più pulito, la banda che passa per il local loop è filtrata: arriva all'incirca a 4000Hz.
- Il telefono è una linea duplex (a due vie): su una linea di questo tipo (da telefono a telefono) il limite fisico calcolato è appunto 35000bps.

Però...

- La situazione cambia quando uno dei due riceventi non è un altro telefono (col suo collo di bottiglia), bensì un servizio digitale. Il provider che è dentro alla rete, si mette nella linea telefonica quindi non mette i filtri...
- In quel caso il limite fisico praticamente raddoppia 70000bps.

Quindi...

- Nel caso ci si colleghi al computer di un altro amico via telefono, siamo vincolati a 33600bps...

- ...ma se ci colleghiamo i Internet o altro servizio digitale, possiamo raddoppiare la velocità.

Allora:

- Eravamo arrivati al V34bis con 14bits per simbolo a 2400baud.
- Potremmo aumentare la frequenza arrivando a 8000 baud con 8bits per simbolo, ottenendo modems da 64000bps (64k!!)
- Quindi 56k certamente possibile... ma come mai non ci colleghiamo a 64k?

64k e 56k

- Il motivo è tecnico/politico: per vari motivi, gli Stati Uniti usano solo 7 invece che 8 bits per i dati.
- A 7 bits per simbolo, si arriva esattamente a 56000 bps (56k).
- L'Europa e il resto del mondo si sono adattati (downgradati) allo standard americano di 56k.

Morale

- Morale: ogni connessione modem in Europa e nel mondo va circa il 15% meno velocemente di quello che potrebbe.
- E dualmente, paghiamo tutti circa il 14% in più quando scarichiamo dati via telefono.

Gli standards più veloci

- Interessante notare che sono asimmetrici:
- V.90 (1998): 56000 bps in download, 33600 bps in upload
- V.92 (1999): 56000 bps in download e 48000 bps in upload

Fax

- Vediamo ora uno dei primi overlay di telefono: il fax.
- 1843: Alexander Bain, primo brevetto
- 1861: Giovanni Caselli, il monaco inventa il Pantelegrafo (come overlay del telegrafo), il primo fax ad essere commercializzato (Francia, poi UK, Italia, Europa).
- Notare: non c'è ancora il telefono
- Viene commercializzato per trasferire immagini, non si capisce il grande potenziale anche per il testo.
- Il primo fax (Parigi-Lione)
- Non c'è stato un grande successo, fino al secolo successivo, quando il Giappone introduce il fax per le comunicazioni
- Motivo ovvio: trasmettere ideogrammi tramite immagini (fax) è molto più pratico che usare una codifica classica tramite alfabeto.

Tipi di fax

- Gruppo 1 e Gruppo 2, ora obsoleti
- Gruppo 3: 6-15 secondi per trasmettere una pagina (dopo il tempo di connessione iniziale).
- Super Gruppo 3 ancora più veloce (c'è poi un Gruppo 4 per linee digitali).
- Risoluzione: ci sono varie modalità
- Le tre classiche sono Standard, Fine e Superfine

Risoluzione Fax (orizzontale x verticale)

- Standard: 200x100 dpi
- Fine: 200x200 dpi
- Superfine: 200x400 dpi
- La massima risoluzione permessa da un fax del Gruppo 3 è 400x400 dpi (la minima è invece 100x100 dpi).

Fax – standard di trasmissione

- V.27 (1988): 4800 bps, PSK
- V.29 (1988): 9600 bps, QAM
- V.17 (1991): 14400 bps, TCM

- V.34 (1994): 28800 bps, QAM
- Il super Gruppo 3 usa anche V.34bis (1996): 33600bps.

Fax: il limite

- Notare: il fax si ferma al V.34bis, perchè una connessione fax è point-to-point, cioè va ad un altro fax, e quindi ha il limite fisico di 35000bps (non riesce a fare il raddoppio di velocità che è riuscito ai modem).

xDSL

- Andiamo oltre, e parliamo delle xDSL, Digital Subscriber Line
- Le DSL nascono essenzialmente dalla spinta di Internet: 56k sono una buona velocità, ma spesso sono troppo pochi (e la situazione è peggioata con l'arrivo del Web...)

Competizione

- La tv via cavo aveva il coassiale, in grado di servire 10Mbps (!)
- Il satellite poteva servire 50Mbps (!)
- → c'era bisogno di molta più.....

Il grosso problema

- Abbiamo visto che coi modem 56k si è praticamente raggiunto il limite fisico ed oltre non si può andare
- Come hanno fatto le compagnie telefoniche ad andare oltre?

Soluzione possibile

- Cambiare il cavo UTP 3 del local loop e relative interfacce telefoniche.
- Conseguenze: costi ENORMI.

L'altra soluzione...

- Ricordate che la banda telefonica è di 4000Hz
- Ottenuta però tramite filtraggio, visto che per la voce le frequenze più alte non servono e anzi interferiscono.
- Perchè allora non rimuovere i filtri?

Infatti

- E' quello che si fa con le xDSL: il filtro di banda viene rimosso, ottenendo una banda possibile (sul classico cavo di rame UTP 3) che passa da
- 4000 Hz a 1100000 Hz (1,1 Mhz)

Però...

- Occorre anche cambiare qualcosa nel local loop, sennò i telefoni, progettati per ricevere segnali fino a 4KHz, riceverebbero invece onde a 1,1MHz
- Per un filtro che toglie il provider, si mette dunque un altro filtro, uno splitter, dall'utente.

Splitter

- Lo splitter separa il segnale telefonico da quello “extra” xDSL per i dati.

Notare

- Uno splitter costa molto poco (perlomeno a chi lo fabbrica...vergognosi poi certi prezzi di vendita in Italia...), perchè è un componente passivo che divide solo il segnale in due, la parte fino a 4000Hz (POTS, Plain Old Telephone System) e quella sopra i 4000Hz.

Notare ancora...: il caso dei telefoni multipli

- ogni splitter genera interferenze
- La cosa migliore quando si hanno più telefoni in casa sarebbe quindi avere un solo splitter in casa, e collegare tutti i telefoni alla parte di cavo a banda 0-4000Hz.
- ...piuttosto che installare splitter ad ogni presa telefonica della casa/appartamento.
- Una volta la cosa non era comoda da fare (se ci sono più prese in casa, occorre fare un rewiring)
- ...ma ora ci viene in aiuto il wireless: la soluzione ottimale sarebbe collegare un solo telefono con tecnologia DECT.....

Problema: Bandwidth e distanza per le xDSL col cavo UTP3
Ora capite...

- perchè certi servizi pubblicizzati come internet super-veloce in realtà per certi funzionano bene e per altri invece sono lenti...
- Dipende molto da quanto fisicamente siete vicini al provider (!)
- Per motivi pubblicitari invece di dà la velocità in caso ottimale ("fino a" …)

Settimana 3

Lezione 1 (07/02/2011)

Come si trasmettono i dati su xDSL?

- Siccome uni dei requisiti è trasmettere sia voce che dati, è ovvio che dev'esserci multiplexing (più canali virtuali schiaffati in un unico canale fisico).
- Abbiamo già visto come funziona lo splitter, quindi è altrettanto ovvio il tipo di multiplexing: in frequenza.

FDM

- In multiplexing in frequenza si chiama Frequency Dvision Multiplexing (FDM).
- In generale, si allocano vari "slot" di frequenza per i vari canali, e poi si fa l'opportuno encoding/decoding.

ADSL

- L'xDSL più comunee è l'ADSL, che sta per Asymmetric DSL.
- Asimmetrica perchè, analogamente agli standard per il modem telefonico V.90 e V.92, dà più spazio al downstream piuttosto che all'upstream.
- Divisione tipica della banda ADSL (FDM); da 0 a 4khz (PSTN), da 25875khz a 138khz (upstream) e da 138khz a 1104khz (downstream); in più da 4khz a inizio up → spazio sprecato, fascia di sicurezza.

Trasporto dati nell'ADSL

- Il modo preciso in cui si usa l'FDM per gli standard ADSL è il cosiddetto Discrete MultiTone (DMT).
- Si spezza la banda in tanti sottocanali di uguale ampliezza ed indipendenti.

ADSL e DMT (caso tipico)

- 256 canali da 4312,5 Hz ciascuno
- 1 per la voce, 5 vuoti, 32 upstream, il resto downstream

Canali indipendenti?

- Indipendenti significa che ogni canale viene trattato come una connessione telefonica a se stante.
- Si usa tipicamente una specie di V34 ma come nel caso di una singola connessione modem, c'è controllo costante sulla qualità di trasmissione
- ogni canale può essere rallentato/accelerato indipendentemente

I vari standard ADSL

- ADSL Lite: 1,5Mbit/s downstream, 0,5Mbps upstream
- ADSL: 8Mbit/s downstream, 1Mbps upstream
- ADSL2: 12Mbit/s downstream, 1Mbps upstream
- ADSL2 (Annex J): 12Mbit/s downstream, 3,5Mbps upstream
- ADSL2+: 24Mbps downstream, 1Mbps upstream
- ADSL2+ (Annex M): 28Mbps downstream, 3,5 Mbps upstream
- Nota: le ADSL2+ usano una banda doppia, cioè 2,2MHz invece che 1,1Mhz → hanno bisogno di una linea particolarmente buona.

Varianti "all-digital"

- Le ADSL2 e ADSL2+ supportano anche varianti "all-digital" (completamente digitali), in

cui si guadagnano 256kbps in upstream rinunciando alla parte voce (POTS), utile ad esempio per linee dedicate in uffici.

FDM in uso

- Il multiplexing con divisione in frequenza (FDM) non si usa solo per l'ADSL, ma si è usata e si susa anche per il telefono in generale, Internet, e molti altri ambiti di multiplexing.

Il telefono classico

- Pensate alla struttura gerarchica del sistema telefonico descritta la scorsa lezione
- Man mano che si sale di gerarchia, servono cavi che portino sempre più segnali (comunicazioni) contemporaneamente.

Soluzione FDM

- Una soluzione possibile è appunto l'FDM, dove i vari canali voce da 4000Hz si dividono la banda disponibile.
- Esempio group/supergroup/mastergroup
  12 canali voce (4kHz) vanno in 60-108khz
  → x5 (supergroup) → x10 (mastergroup)

Quindi...

- Un mastergroup (terzo livello) può tenere fino a 600 conversazioni contemporaneamente su un solo cavo.
- Con altri standard si arriva fino a 230000 canali voce.

Il caso della fibra ottica

- Nel caso della Fibra, la FDM si chiama WDM (Wavelength Division Multiplexing), parliamo di fasci di luce, quindi lunghezze d'onda.

WDM (Wavelength DM)

Interessante notare

- Gli encoder e i decoder, trattandosi di luce, possono essere costruiti a tecnologia **completamente passiva** (non c'è corrente), e quindi danno componenti molto più affidabili e duraturi.

Fibra e WDM: dove si arriva?

- 1990: 8 canali da 2,5Gbps
- 1998: 40 canali da 2,5Gbps
- 2001: 96 canali da 10Gbps
- 2007: 124 canali da 50Gbps
- Cifre, cifre, cifre...ma in pratica?

Fibra e WDM, cifre in pratica

- 2001: 96x10 = 960Gbps si possono trasmettere centosessanta film al secondo.
- 2007: 124x50 = 6200 Gbps, 1033 film al secondo.

Ancora multiplexing

- Non c'è solo l'FDM (multiplexing in frequenza)
- Un altro multiplexing che si usa è il TDM, multiplexing temporale.
- Nella sua incarnazione telefonica è anche detto PCM (Pulse Code Modulation).

TDM (Time DM): Linea T1

- 24 canali voce (1544 Mbps di dati)

TDM di ordine superiore

- Anche qui si può procedere a cascata T1--> T2 → T3 → T4

Si usa?

- Più di quello che pensate...

WAV, PCM lineare, AIFF

- Un cd audio è una spirale, un canale di trasmissione; stereo → 2 canali; serve del multiplexing, altrimenti come farebbe lo stereo a trasmettere 2 canali in uno unico?

Switching

- Ricordiamo la struttura gerarchica della rete telefonica (che si chiama PSTN, public switched telephone network).
- → abbiamo detto ci sono gli switching center (i centralini che dirigono il traffico sulla gerarchia)
- Ma come lo dirigono?

Due tecniche

Ci sono essenzialmente tre tecniche principali di switching: a circuito, a messaggio, a pacchetto

Circuit switching

- punto a punto; si crea un collegamento fisico tra i due che comunicano, si crea un cammino fisico tra le due persone che vogliono comunicare.

Importante notare

- Per creare un collegamento fisico (end-to-end) occorre del tempo.
- → delay iniziale

Message switching

- Invece di creare il cammino e poi iniziare la trasmissione, lanciamo direttamente il messaggio.
- Quando arriviamo a uno switch, aspettiamo che ci dicano dove andare (cioè, il cammino viene creato man mano che lo attraversiamo).
- Si chiama anche **store-and-forward**.

Store and forward...

- Problemi?

Packet Switching

- Proseguendo in quest'ottica, c'è poi il packet switching.
- Un problema del message switching è che se il messaggio è grosso, occupiamo risorse di uno switch, magari ….....

Divide ed impera

- Col packet switching, si divide il msg in tanti sottomessaggi (packets) di lunghezza massima prefissata.
- In tal modo, non solo non occupano risorse ma permettiamo anche trasmissioni di parti del messaggio **in parallelo**.

Confronto tra circuit e packet

Call setup → required → not needed
Dedicated physical path → yes, No
Each packet follows the same route → yes, no
Packerts arrive in order → yes, no
Is a switch crash fatal → yes, no
Bandwidth available → fixed, dynamic
When can congestion occur → at setup time, on every packet
Potentially wasted bandwidth → yes, no
Store-and-forward transmission → no, yes
Transparency → yes, no
Charging (prezzo) → per minute, per packet

Il circuit switching considerato è quello automatico...

- Ma all'inizio, come abbiamo detto, c'erano i centralinin maniali con operatore
- Si è poi passati al circuit switching automatico, inventato da Almon Strowger nel 1891, (strowger gear)
- Almon Strowger
- Aveva la passione per l'elettronica, ma di mestiere faceva...il becchino!

Strowger e gli switch

- Nella sua città del Missouri c'erano due becchini, lui ed un altro.
- Tipicamente, quando una persona moriva, si chiamava il centralino chiedendo di un becchino.
- Un bel giorno come centralinista è stata assunta la moglie dell'altro becchino! → link costante tra tecnologia e società.

Lezione 2 (08/02/2011)

Premessa di marketing (tecnologia e società)

- Uno dei pochi campi dove (per ora) l'Europa (e pure l'Italia) battono gli USA.
- Quali sono stati i motivi?

Vari motivi ma chiari

- Standardizzazione: l'Europa dopo i primi tentativi, ha capito che occorreva un unico standard europeo.
- Gli Stati Uniti invece, per eccesso di ***liberalismo***, hanno lasciato che ci fossero standard multipli negli USA, creando reti incompatibili.
- Altro motivo: il principio della conoscenza tariffaria
- Essenzialmente: quando usate un servizio a pagamento, dovete conoscerne le condizioni (tariffe).

Condizioni di pagamento

- La telefonia mobile costa di più, ma non è (finora) stato un problema perchè i numeri dei cellulari sono diversi da quelli dei telefoni fissi.
- Negli USA invece, non c'è distinzione, e quindi si è deciso di far pagare il surplus al possessore del cellulare.
- → rallentamento del mercato
- Altro motivo (paradossale): la grande competizione negli USA della telefonia fissa, e conoseguenti tariffe molto basse.
- In Europa invece (spericialmente in Italia) il contrario, e quindi mercati alternativi (mobile) sono fioriti molto più velocemente.

La telefonia mobile: 0G, 1G, 2G, 3G, 4G...

- Si distingue tecnicamente in varie “generazioni”
- La prima generazione (0G e 1G) analogiche, le altre digitali.

Premessa: tutta la telefonia mobile...

- Si basa su un problema fondamentale: la divisione del territorio
- In altre parole, come gestire l'infrastruttura fissa che permette il miracolo della connessione mobile.

L'appiglio fisso...

- ...è lo switching center (“centralino”), che copre una certa zona di territorio: la cella telefonica.

0G: anaglogica (1950 circa)

- Deriva dalle trasmissioni radio (che vedremo), che si sono poi evolute nei cosiddetti sistemi PTT.
- PTT = Push To Talk (l'equivalente delle moderne radioline walkie-talkie o amatoriali: si preme per parlare).
- Un solo canale per ricevere e trasmettere, quindi **half-duplex** → “push” per trasmettere senza ricevere.

Corsi e ricorsi della storia

- 0G, il PTT, è stato reintrodotto molto recentemente in alcuni cellulari di ultima generazione (es. “Moto talk”).
- → essenzialmente un cellulare può anche funzionare da walkie-talkie!

1960s: il sistema IMTS

- Improved Mobile Telephone System
- Passa a due frequenze, quindi non serve il push to talk.
- Aumenta il livello di privacy: finalmente non si sentono le comunicazioni degli altri (pensate ad esempio alle radio dei taxi in uso ancora oggi, o ai PTT).
- Super-trasmettitori ad altissima potenza
- Per evitare interferenze, “celle” di centinaia di chilometri
- 23 canali nella banda 150-450MHz → troppo limitativo.
- Però....Ancora in uso in certe zone remote del Canada (vantaggio: servono pochi ripetitori).

1G: Vent'anni dopo....1982

- I Bell Labs introducono l'AMPS (Advanced Mobile Phone System), anche conosciuto TACS in Italia (Total Access Communication System).
- Differenza fondamentale rispetto all'IMTS, ora le celle sono molto più piccole, 10-20km.

Vantaggi

- → la capacità (utenti serviti) aumenta di un ordine di grandezza
- → E, diminuisce la potenza richiesta per la trasmissione (in ambo i versi), quindi minor costo, ed apparecchi telefonici più leggeri.

Però....

- Celle più piccole portano all'amplificazione di un problema già presente in 0G....
- L'interferenza tra celle!

Come gestire la situazione?

Soluzione: separazione di frequenza, ad ogni cella dò una frequenza diversa dalle frequenza usata dalle celle adiacenti.

Problema....

- Quante frequenze usate?
- In teoria, più frequenze usate per separare le celle, meno banda avete per singola cella.
- Occorrerebbe trovare un numero basso di frequenze che basti a separare tutte le celle.

Beh...non è facile?

Il problema....

- (oltre ad un altro che vedremo dopo), è che nella grande maggioranza dei casi non avete controllo totale sul terreno.
- Pensate a città, strade, fiumi, colline. Ecc..
- Occorre quindi trovare il numero minimo per ogni situazione

Questo numero....

- Deriva da un teorema famoso...e da un altro problema pratico...(carta dell'America divisa in stati col righello).

1852

- Francis Guthrie, cercando di colorare la mappa dell'Inghilterra, nota che sono sufficienti solo quattro colori, e congettura che sia sempre così.

La sfida....

- Varie persone dimostrano il teorema...
- Alfred Kempe nel 1879...
- Nel 1890 ci si accorge che la dimostreazione è sbagliata!
- Alcuni provano che è falso...per poi essere sdimostrati...
- 1976: Il problema resta aperto per 24 anni, finchè non viene dimostrato nel 1976, → teorema dei 4 colori; con una famosa dimostrazione di Kenneth Appel e Wolfgang....

Famosa?

- Famosa perchè la dimostrazione viene fatta al computer.

Dimostrazioni e computers....

- 1976: 500 pagine, 1936 casi...
- Nel 2004, prova formalizzata usando Coq (un proof assistant), da parte di Benjamin Werner e Georges Gonthier.
- (INRIA-Microsoft)

Celle: quando il sistema in certe zone va in overload...

- Si cambia la strutture a celle creando microcelle

Quindi....

- Bastano 4 colori in ogni situazione...in teoria....
- Ma in pratica?
- Nella pratica, dipende: in qualche caso, conviene mettere più colori (frequenze).

Esempio

- Supponete di aver buon controllo sul territorio (quindi, di poter più o meno decidere dove mettere gli switching center).
- Cosa; il caro vecchio foglio a quadretti?
- Occorre capire...il terreno (→ la società)
- Ad esempio, tipicamente i palazzi e le strade sono dritti.

Torniamo al foglio quadrettato...

- come interagirebbe con le strade? Ci sono zone al limite tra una cella e l'altra, questo sistema non va bene, ho troppe linee lunghe e dritte.
- In città, si sfasano le celle usando una matrice esagonale.
- → 7 frequenze (configurazione usata soprattutto in zone densamente popolate).

Ogni cella...

- Ha quindi al centro la stazione base (lo switching center)
- Un cellulare è sempre connesso ad una sola cell, finchè non si sposta....
- → quando il segnale è troppo debole, lo switching office chiede alle celle vicino quanta potenza ricevono dal cellulare.
- Alla cella con potenza più alta viene assegnato il cellulare.

Il passaggio di celle...

- Si chiama handoff
- Richiede in media circa 300msec (0,3s) che sono tanti se c'è una chiamata in corso.
- Nell'hard handoff la vecchia stazione “molla” il cellulare, e poi la nuova lo riaggancia.
- → c'è del lag, e in qualche raro caso (sfiga) la linea cade.

L'handoff “soft”

- L'handoff può anche essere soft: la nuova cella acquisisce il cellulare prima che la vecchia lo lasci.
- Il problema è che il cellulare deve sapersi collegare a due frequenze (due celle) contemporaneamente, cosa che aumenterebbe i costi e la potenza.
- → 1G e 2G non lo gestiscono

Caratteristiche tecniche 1G (TACS / AMPS)

- Ogni cella gestisce tanti utenti, occorre quindi fare multiplex.
- Che tipo di multiplex si usa? FDM (in frequenza).
- Uso più frequenze dentro alla cella e separiamo gli utenti dando freq diverse.
- Usa quindi FDM per gestire 832 canali.
- Ogni canale full-duplex (coppia di canali simplex).
- 832 canali simplex in trasmissione (824-849 Mhz), e
- 832 canali simplex in ricezione (869-894 Mhz).

Capacità reale

- Come per gli stipendi e le tasse: in realtà la capacità di una singola celle è molto meno

- Alcuni canali sono usati per controllo, altri non possono essere usati per via dello smistamento delle frequenze dovuto alle celle.
- → circa **45 canali effettivi per cella**.ù

Funzionamento

- Ogni cellulare ha un **numero seriale** di 32 bit, e un **numero telefonico** di 10 cifre (34 bits).
- Ogni 15 minuti circa il cellulare manda in broadcast i suoi 32+34 bits per registrarsi alla cella più vicina.
- Quando si chiama, si usa il canale apposito (condiviso) per attivare la richiesta.
- In ricezione invece, c'è un canale apposito di **paging (condiviso)**, che i cellulari controllano per sapere se ci sono chiamate che li riguardano.
- Se ne trovano una, rispondono alla cell,a che dà oro un canale esclusivo per la comunicazione.

Da 1G a 2G: il digitale

- Come nel caso di 1G, anche in 2G non si è arrivati ad un unico standard mondiale (sic).
- Gli standard principali: D-AMPS, PDC, GSM, CDMA.

D-AMPS e PDC

- D-AMPS è lo standard statunitense, PDC funziona con la stessa tecnologia ma con piccole differenze per il mercato giapponese
- Come suggerisce il nome, D-AMPS è compatibile col vecchio sistema 1G AMPS: i due sistemi possono coesistere nelle stesse celle.

D-AMPS

- Riusa tutte le bande di AMPS sugli 850Mhz (solo, stavolta, in digitale), ed in più ne ha di altre aggiuntive per aumentare la capacità del servizio:
- Si va su: 1850-1990 Mhz.
- In questa banda, le onde sono più corte (16cm), quindi se si usa la banda extra basta un'antenna più piccola.

Lezione 3 (09/02/2011)

D-AMPS rispetto ad AMPS

- Trasmettendo in digitale, usa anche tecniche di compressione del flusso voce
- Dai classici 56kbps per un flusso, si può passare tramite compressione a...
- 8kbps (a volte anche 4kbps!!)
- → si riescono a mandare sulla linea sei volte gli utenti del vecchio AMPS.

Esempio di compressione (semplificato): delta modulation

- delta perchè invece di mandare info su tutti i punti della curva, mando solo la differenza (delta) tra l'instante precedente e quello attuale. Il delta è fisso, se sull'asse x ho 0 decresce, se ho 1 cresce.
- Le comunicazioni “compresse” vengono quindi gestite in multiplexing usando una classica TDM (Time Division Multiplexing).

Svantaggi

- La qualità del suono è peggiorata da 1G a 2G, perchè 1G è analogico (e non comprimeva i dati), 2G comprime con perdita di dati.

Altra differenza è nella gestione dell'handoff.

- In AMPS è il control switch che se ne occupa per tutti i cellulari
- Funziona, ma potenzialmente è un collo di bottiglia (se ci sono troppi cellulari in una cella?)

In D-AMPS...

- Si ribalta questo paradigma:
- L'onere viene lasciato ai singoli cellulari! Come?

Come in un rapporto di coppia...

- Il cellulare è associato (“partnership”) ad una cella....
- Ma non è schiavo/succube: periodicamente monitora la qualità del rapporto.
- → misura la potenza del segnale (basta fissare la potenza di ogni switch center!)

Handoff in D-AMPS

- ...quando il segnale è basso, è il cellulare a “protestare” con la base.
- Che può disconnetterlo... (lo “molla”)
- … a quel punto il cellulare, tornato single, si guarda in giro, e può riacquisire il segnale più potente (attrattivo) che ha a disposizione....

MAHO

- Questa tecnica innovativa si chiama MAHO (Mobile Assisted Handoff)
- Da notare: il carico sul cellulare è minimo, perchè...
- ...si sfruttano i “tempi morti” dovuti al multiplexing temporale (TDM) per misurare la potenza del segnale!

GSM

- Passiamo ora all'altro sistema che compete col D-AMPS, e che si usa in Italia ed Europa: il GSM
- GSM sta per Global System for mobile communications
- e' stato introdotto nel 1988 per la prima volta in...?
- Finlandia, immense distese e poche case ogni tanto.

Comunicazione nel GSM

- Simile a D-AMPS: si usa FDM (ovviamente, con frequenze diverse), con TDM
- Differenza: i canali del GSM sono molto più ampli di quelli AMPS (200 khz rispetto a 30 khz).
- → tengono più utenti (8 rispetto a 3), ed hanno una data-rate per utente più alto.

GSM e canali complessivi

- 124 canali, ognuno con 8 slots TDM

Nota...il fatto che usi TDM, lo sapevate già...(bip)

- cellulare posto vicino ad uno stereo, rumore.

GSM rispetto a D-AMPS

- Ogni canale gestisce 270833 bps
- → diviso 8 utenti: 33854 kbps (D-AMPS era 8,1 – 16,2 kbps)
- Meno l'overhead → 24,7 kbps
- Dopo la correzione degli errori: **13kbps finali**.
- → molto meglio del D-AMPS (8 / 4 kbps)
- → qualità voce molto migliore, così come trasmissione dati (modem) più decente.

Struttura a cella GSM

- Quattro tipi di celle: macro, micro, pico e umbrella
- Macro: le più grandi, sopraelevate rispetto agli edifici (raggio massimo: 35km)
- Micro: cella più piccola, altezza fino al tetto di un edificio
- Pico: molto piccole, usate per aree molto dense tipicamente indoor
- Umbrella: piccola estensione, usate per coprire i “buchi” tra le celle principali

Altre caratteristiche del GSM

- L'uso della SIM “Subcriber Identity Module”, introdotta in 2G, con 1G non esistevano; in questo modo si stacca il numero telefonico dal cellulare.
- Varie taglie, da 4kb fino a >512Mb.
- Contiene varie informazioni, ma le due più importanti sono la **IMSI** e la **Ki**.

IMSI e Ki

- IMSI: International Mobile Subscriber Identity, è l'identificativo della SIM.

- Ki: è la chiave di autenticazione (→ GSM supporta autenticazione crittografica a chiave condivisa), c'è comunicazione criptata tra la sim e la base.

Collegamento crypto GSM

- Il cellulare manda l'IMSI della SIM all'operatore.
- L'operatore genera un numero casuale e lo manda al cellulare.
- Il cellulare firma il numero con la Ki e lo manda all'operatore.
- L'operatore ha nel suo db l'IMSI e la Ki associata: firma anche lui il numero casuale con la Ki, e controlla che il numero sia lo stesso di quello inviatogli dal cellulare.

CDMA

- Passiamo ora al terzo standard 2G:
  CDMA
- Mentre D-AMPS e GSM sono abbastanza simili (come core, FDM e TDM), il CDMA invece funziona in modo diverso:
- Non usa né FDM né TDM.
- Usa una tecnologia molto importante, che è usata anche da altri sistemi (ad esempio per reti Internet Wireless)

Come funziona il CDMA? Party Internazionale

Pensate ad un Party...

- Tante persone che parlano...c'è il problema di capirsi.
- Tante persone in una stanza, tutti parlano...
- FDM: ogni coppia parla su “toni” diversi.
- Con TDM: quando uno parla, tutti gli altri stanno zitti, e si fa a turni.

CDMA?

- Sta per Code Division Multiple Access, divisione della codifica;
- In realtà la “C” sta per....Casino!!
- Il Party CDMA: ognuno parla contemporaneamente
- E quindi? Si usano lingue diverse (international party)

Come funziona?

- Tutto sta in come si modella il concetto di “lingua diversa”
- Una lingua è composta da un certo numero di parole (oggetti)
- Quello che ci serve è un modo per stabilire se due parole stanno o meno nella stessa lingua

Lingue...spazio di parole...

- Possiamo vedere una lingua come uno spazio...come facciamo ad avere spazi che siano interoperabili ma separabili?
- Ma lo sappiamo già fare....spazi dimensionali!

Quindi, idea...

- Se vedessimo una “lingua” come una coordinata ed avessimo uno spazio multidimensionale?
- Ogni parola in una certe lingua starebbe nell'asse corrispondente

Criterio stessa lingua

- Facile: se stanno sullo stesso asse, allora le parole sono della stessa lingua, se sono diverse sono invece perpendicolari.
- Stessa lingua = stanno nello stesso asse (e hanno proiezione = 0 negli altri zeri).
- Come si fa a calcolarlo? Basta calcolare il prodotto scalare.

Quindi, il nostro criterio:

- Due parole non stanno nello stesso linguaggio se sono “perpendicolari” → ortogonali, cioè con prodotto scalare uguale a zero.

Bene, ma quando nel party parlano tutti?

- Cosa succede quando più persone parlano?
- Che le “parole” si sommano

E come facciamo poi ad ascoltare solo in una lingua?

- Beh, selezioniamo solo la componente che ci interessante
- → facciamo la proiezione del vettore “multilingua” solo su un asse.
- Facciamo il prodotto scalare del vettore per un'asse

CDMA

- CDMA funziona allo stesso modo: si lavora su uno spazio multidimensionale
- Si stabiliscono degli assi adeguati.
- E poi si usano le regole di composizione (somma) e proiezione (prodotto scalare) pere fare encoding/decoding.

Onde ortogonali (“lingue diverse”)

- rappresento un'onda con -1 e 1 e metto in una matrice, se il prodotto mi dà zero → sono lingue diverse.
- 1,1,1,1
- 1,1,-1,-1
- 1, -1, -1, 1
- 1, -1, 1, -1

La proprietà riflessiva

- Invertendo un'onda abbiamo sempre un'onda nello stesso linguaggio (-0 = 0!)

Quindi, abbiamo un modo semplice per creare da una parola in un linguaggio, una parola nello stesso linguaggio: invertire la forma d'onda

→ per avere un alfabeto minimale (due simboli, codice binario), ci basta una sola parola, e poi invertiamo per ottenere l'altra.

La creazione degli assi

- In teoria, per avere n assi, potremmo semplicemente usare....
- 1,0,0,0,...
- 0,1,0,0,....
- 0,0,1,0,...

LINE principale: Life Is Never Easy; siccome la vita non è mai semplice, per motivi tecnici (pensateci...) si devono usare altri assi, con +1 e -1

- Si usano le cosiddette matrici di Walsh, che sono essenzialmente derivate dalle matrici di Hadamard.
- Le matrici di Hadamard sono fatte da valori +1 e -1, tali che ogni riga è perpendicolare (ortogonale) a ogni altra.
- Come si costruiscono? Metodo classico.

Notare

- Il metodo che abbiamo usato (inventato da Sylvester nel 1867) crea matrici di Hadamard di grandezza 2alla n
- Resta però il problema dell'esistenza di matricidi d Hadamard per un numero qualsiasi di linguaggi.

Qualsiasi numero di lingue?

- 1867: Sylvester (2 alla n)
- 1893: 12 e 20 (Hadamard)
- 1933: p+1 con p primo (Paley)
- ...numero minimo per cui non si poteva fare: 92
- 1962: Baumert, Golomb e Hall dimostrano col computer che si può.
- → il numero minimo passa a 428

- 2004: Kharaghani, Tayfeh-REazaie mostrano che si può fare, il numero minimo attuale è 668 (problema dunque ancora irrisolto).

10 febbraio 2011

H1 = [1]
H2 = 1 1
1 -1

Ci sono 4 canali: H4:

1	1	1	1
1	-1	1	-1
1	1	-1	-1
1	-1	-1	1

Facciamoli parlare (4 cellulari)
Italiano: spaghetti, vino
Americano: hamburger, coca-cola
Tedesco: salcicce, birra
Giapponese: sushi, sake'

spaghetti = prima riga di H4	1, 1, 1, 1
vino = prima riga di **H4 riflessa**	-1, -1, -1, -1
hamburger = seconda riga di H4	1 -1 1 -1
coca-cola = seconda riga di **H4 riflessa**	-1 1 -1 1
salcicce = terza riga di H4	1 1 -1 -1
birra = terza riga di **H4 riflessa**	-1 -1 1 1
sushi = quarta riga di H4	1 -1 -1 1
sake' = quarta riga di **H4 riflessa**	-1 1 1 -1

SITUAZIONE

- italiano dice spaghetti	1 1 1 1	Ognuno trasmette la sua
- americano dice coca cola	-1 1 -1 1	forma d'onda
- il tedesco sta zitto	_____	nello stesso istante:
- la giapponese dice sushi	1 -1 -1 1	le form d'onda di sommano:

= 1 , 1, -1, 2 (forma d'onda risultante)

Quando quelcuno vuole ascoltare...

L'italiano:
deve controllare se c'è una parola nella lingua italiana, e se c'è di quale parola si tratta:
(1 1 -1 3) → Corrisponde al punto multidimensionale
(spaghetti: 1 1 1 1)
1+1-1+3 = 4
→ C'è una parola in italiano (altrimenti avrebbe dato **0**)
La proiezione ci da le coordinate e il valore **positivo (la parola è allineata con la corrispondente spaghetti)**, ci dice **spaghetti**.

L'americano:
1 1 -1 3
(ham 1 -1 1 -1-)
=1-1-1-3=-4
→ C'è una parola americana! E' negativo, dunque è **coca-cola**.

La giapponese:
valore positivo → è sushi

Il tedesco:
(1 1 -1 3)
(salcicce: 1 1 -1 -1)
=1+1+1-3 = **0!**
→ NON C'È NESSUNA PAROLA IN TEDESCO

Perché CDMA è stato introdotto recentemente?

1) la soluzione è stata trovata relativamente recentemente.
2) Problema tecnico che ha reso l'approccio difficile:
   CDMA assume che **ognuno dei cellulari parli il suo linguaggio con lo stesso volume. → problemi di attenuazione.**
3) Se qualcuno parla più forte degli altri si sballa tutto!

Questo ha fatto sembrare l'approccio **impossibile** per le comunicazioni mobili, dove la distanza dei cellulari dalle base è variabile
Soluzione: far si che ogni cellulare parli **"più o meno forte"** a seconda do quanto lontano è dalla base, cosicchè la base senta tutti i cellulari allo stesso modo (!)

Come si fa?

La base trasmette sempre a una **potenza fissa** (standard), nota al cellulare
Dalla potenza del segnale che riceve il cellulare può calcolare quanto è lontano dalla base
La base trasmette a una potenza opportuna tipicamente che va con l'inverso della distanza.
Conseguenze:
Nel CDMA si strutta tutto lo spettro di banda possibile
Sfrutta al meglio la caratteristica del traffico voce, **l'intermittenza**
Una voce umana in una conversazione è "attiva" circa il 35-40% di pausa del tempo
→ Nel TDM si spreca tempo per il silenzio, in CDMA invece no (!)

Tra 2G e 3G: GPRS

General Packet Radio Service
E' classificato "2.5G"
Essenzialmente è un overlay dei 2G (GSM e D-AMPS) che permette la gestione del traffico a pacchetti.
Il problema del GSM
GSM essenzialmente è stato costruito per trasmettere voce
Questo significa che se si vuole usare il GSM per trasmetter dati (ad esempio, come modem), ci sono grafi pecche:

1) la principale: viene riservato una **canale intero** alla comunicazione
2) → navigare in internet spreca un intero canale voce anche quando il traffico è poco (la navigazione Web classica ha molte pause)
3) Inoltre, la tariffa corrispondente **è a tempo**, e **non a traffico**.

Quindi GPRS permette analogamente al concetto di switch (di cui abbiamo parlato), la navigazione **a pacchetti** e non a messaggi interi.
→ Con tutti i vantaggi conseguenti: **non si spreca banda**, **non serve un canale dedicato**, **si possono usare tariffe a traffico**.

GPRS e Internet

- Supporta i classici protocolli **IP** e **PPP**.
  - Alloca dinamicamente i canali "Internet" e quelli voce, a seconda delle richieste di traffico.

Tipi di cellulari GPRS
Classe C: si possono connettere o come GSM o come GPRS, e l'utente deve settare manualmente quale comunicazione usare
Classe B: si può connettere o come GSM o come GPRS; ad esempio se si riceve una chiamata mentre si scaricano dati, si sospende il GPRS (classe più comune).
Pseudo classe A: li può usare contemporaneamente, usando una sola frequenza (però la rete deve supportarlo (il cosidetto **dual transfer mode** (DTM)).
Classe A: uso contemporaneo di GSM e GPRS, usando frequenza diverse (non serve una rete speciale, è come avere 2 cellulari uno dentro l'altro: uno che fa il traffico voce e l'altro il traffico dati)

Tra 2G e 3G: EDGE

Enhanced Data rated for GSM Evolution (talvolta anche detto EGPRS)
E' classificato 2.75G.
Oltre alla modulazione di frequenza usa **più modulazioni di fase**, e resta backward compatibile sia con GSM che con GPRS.

- Abbiamo varie versioni
- Velocità variabile: da 64kbps alle nuove 236kbps (Edge Evolution).
- La velocità attuale però varia molto anche a seconda della qualità del servizio implementato.

Altre applicazioni potenziali?

Introducing: 3G !!

Rispetto al 2G: più data rate, più utenti supportati.
Usa tipicamente bande di frequenza più larghe rispetto al 2G.
Due standard principali: W-CDMA e CDMA2000.
W-CDMA: Wideband CDMA, conosciuto in Europa anche in UMTS
→ usa una banda per canale molto larga, **5 Mhz**.
→ Data rate: 384kbit/s

3 (H3G) usa SOLO UMTS.
CDMA200 è l'equivalente statunitense e giapponese.
Velocità massima tipica: **144kbit/s** (3 volte più lento!!!! TIE!) [si è preferito così: hanno più canali]
VS
W-CDMA ha bande di 5MHz mentre CDMA piu strette, di 1.25Mhz
4G?
Ovviamente, con costante domanda.
Oltre l'UMTS:
HSDPA: Hight Speed Downling Packet Access (3.5G) E' l'evoluzione del GSM.
Varianti: 1.8 3,6 7,2 14,4 Mbit/s
E dai 384Kbitps si 3Mbit/s in upload
In italia viaggiamo sui 7.2Mbitps (sempre cifra massima teorica).
HSUPA Highe Speed Uplign Packet Access
considerato 3.75 G

Evolutione di HDSPA con velocità max di 5,6Mbps
Nuova versione 11.5Mbps. Ma se nel 3.5G si arriva a **14,4 Mbps**.
HSUPA è stato sviluppato come standard 3.75G e in quel momento le versioni di HSDPA arrivavano a velocità inferiori (3.6Mbps)
Poi si è visto che HSDPA poteva essere spinto oltre, quindi con costi inferiori mantenendo la retro compatibilità.
Attualmente: 28Mbps in DOWN e 11Mbps in UPLOAD (!)
Velocità massima supportata dallo standard: 42Mbps
HSOPA Hight Speed OFDM Packet Access o E-UTRA.
Usa bande variabili (più sofisticato), da 1.25MHz a 20MHz.
Anche detta **Super 3G**.
Motivo? 326Mbps in DOWNLING, e 86Mb in uplink (!!!)

HSOPA / E-UTRA
Prime versioni già commercializzate (dicembre 2009) in Svezia e Finlandia, con velocità di 50Mbps e 5.3Mbps in uplink.
Da ultimo....
Vediamo un paio di esempi di multiplexing wireless comuni, uno antico e uno più recente.
1) lo **stereo**. Nella stereofonia ci sono 2 canali da trasmettere. Una cosa interessante è la trasmissione radio stereofonica.
FM stereo. Come si trasmette in **stereo** usando FM? Occorre garantire la retro-compatibilità.
Problema del tutto simile per le **trasmissioni** televisive stereo.

- La banda mono va sui 30Hz – 15kHz
- E' stato introdotto un segnale pilota su frequenza più alta di tutte le trasmissioni mono (19kHz), che segnala la presenza del segnale stereo.
- La radio stereo controlla questa banda e vede se c'è il supporto stereo.
- I due canali: S e D vengono trattati così:
  Viene creato un **nuovo segnale** che è la media dei due: M=(S+D)/2
- Questo segnale è quello che si trasmette sulla vecchia banda per garantire la retro-compatibilità.
- Viene inoltre segnato un altro segnale: **E(S-D)/2 (*)** (la media delle differenze dei 2 canali)
- Queste vengono trasmette sui 23-56kHz.
- Una radio stereo ricostruisce un segnale stereo dall'informazione che ha dal canale mono e sommato o sottraendo col segnale E (*) :
  - S = M+E
  - D =M-E

Settimana 4
Lezione 1 (14/02/2011)
FM Stereo (cont)

- Invece, una radio stereo sarà anche in grado di decoficiare l'altro segnale E=(S-D)/2.

Notare

- Tutto questo avviene ad un prezzo: aumenta il rapporto segnale rumore (di almeno un fattore tre rispetto al segnale stereo).
- Per questo motivo, considerando solo la componente mono (M), la qualità del suono può aumentare di molto.
- E per questo molte autoradio passano a mono automaticamente quando la qualità stereo è troppo povera

Inoltre...

- C'è altro multiplexing, nel caso la trasmissione FM supporti anche **RDS (Radio Data System)**.

- I dati RDS sono **digitali** e vengono trasmessi su frequenze **ancora più alte: 57khz** (la terza armonica del segnale pilota stereo (19 khz).
- Il data rate (se mai velo siete chiesto) è molto basso: circa **1,12kbps**

E...

- C'è già l'evoluzione nella radio digitale HD Radio (Hybrid Digital) in aMerica (o via statellite), e DAB (Digital Audio Broadcasting) e DAB+ in Europa.
- Tra i vantaggi: molte più radio!!! (ora si sprecano frequenze!).

Secondo esempio....

- Il digitale terrestre (http://www.youtube.com/watch?v=5uWTjxj6sXU)

Utente "creatorediillusioni"

- "ragazzi mi dispiace per voi, ho provato ma non funziona una mazzza! Vecchio tv catodico, patata 8 cm e cellulare motorola con cavo usvb, apparte vedere malissimo le 3 o 4 reti nazionali, cosa che si fa anche se ci metto io il dito...."
- Utente "SirCrabro", non voglio fare il solito scettico, ma, mentre posso capire che la patata con la stagnola faccia da antenna, non capisco come faccia anche da decoficiatore...come trasforma lo 01010101001 in segnale analogico? Faccio Ingegneria Informatica e non sono l'ultimo arrivato..."

Cosa usa in realtà?

- Divisione di frequenza (OFDM), in maniera molto simile all'ADSL.
- Ergo, quanti "minimodem"...?
- 2048-4096-8192 (!)
- E dentro ad ogni banda di frequenza, usa i **QAM** (QPSK, QAM16, QAM64).
- (e dentro ogni canale, compressione MPEG2)

Lo strato Data Link

- Dopo lo strato fisico, cominciamo a parlare dello strato immediatamente sovrastante, quello che si occupa del collegamento dati.
- Essenzialmente, si occupa del problema dello codifica e decodifica di un flusso di dati.
- Più in dettaglio, questo strato: fornisce una interfaccia allo strato di rete (network layer) sovrastante.
- Si occupa degli errori di trasmissione (error control).
- Regola il flusso di dati a seconda delle capacità della rete e del ricevente (flow control).

Varie possibilità

- Ci sono ovviamente vari modi in cui si può creare un canale:

Unacknowledged connectionless

- In questo servizio, i pacchetti vengono inviati senza aspettare conferma di una eventuale ricevuta (unacknowledged)
- ...e tantomeno senza stabilire una connessione dedicata (connectionless).
- Utile ad esempio per voce/streaming media, o quando il canale è molto affidabile.

Acknowledged connectionless

- L'altro servizio analogo è come il precdente, solo che questa volta i pacchetti ricevuti vengono "confermati" (acknowledged) con ricevuta di ritorno.

Acknowledged connection-oriented

- Infine, il massimo del lusso: una connessione dedicata, con ricevuta di ritorno. :-)

La codifica: framing

- L'approccio classico è di prendere dei pacchetti dati dallo strato superiore (network), e codificarli in appositi frames.

Il problema del contorno....

- Uno dei problemi della trasmissione dei frame è accorgersi di dove un frame inizia e finisce.
- Si potrebbe usare la sincronizzazione degli orologi, ma è impraticabile per vari (abbastanza

ovvi) motivi, quindi si sono scelte altre strade.

Il metodo del character count

- Semplicemente, mettiamo nell'header l'informazione del numero di caratteri che costituiscono il corpo dati (il payload).
- Simile a quello che fanno certi linguaggi di programmazione internamente...

Problemi...

- Se c'è un errore....disastro!
- “è bastato un solo bit sbagliato, per mandare a donne di facili costumi tutto...”

Altri metodi

- Il character count è stato uno dei primi metodi (derivati appunto dai linguaggi di programmazione)
- Ma l'ambiente delle reti, dove ci possono essere errori molto più frequentemente.....

Il metodo dei flag bytes

- Il vero problema del character count è che perdiamo la “sincronizzazione” se c'è un errore
- usiamo un byte speciale (flag bytes) per segnalare l'inizio e la fine di ogni frame.
- Analogo delle virgolette nei linguaggi di programmazione

Flag bytes

- Come per le virgolette, serve un metodo che ci permetta di fare l'escaping delle virgolette stesse, in modo da poter trasmettere qualunque messaggio.
- Nel gergo delle reti, la tecnica si chiama **byte stuffing** (o character stuffing).

Problema

- Un problema è dovuto all'analogo dei linguaggi di programmazione quando si è passati dai caratteri ASCII (bytes) a caratteri più globali (UNICODE):
- usare bytes o in ogni caso grandezze fisse prima o poi non va più bene.

Il bit stuffing

- è l'analogo del byte stuffing, stavolta fatto a livello di bit
- Ad esempio, si prende come “flag” 01111110 (0 – sei 1 – 0)
- Se lo stream di bits ha cinque 1 di fila, allora dopo il quinto bit viene inserito uno 0.

Passiamo ora...

- ad un altro aspetto del data layer, molto importante!

Gli errori

- Se tecniche come quelle che abbiamo visto ci permeettono di sapere come identificare un frame sulla rete, resta in ogni caso il problema più grave: come comportarsi
- ERROR CONTROL:
- Ci sono essnzialmente 2 strategie che possiamo seguire:
- una è fare error detection, cioè sviluppare tecniche che ci dicono se l'errore è stato rilevato; altro è l'error correction, cioè correggerlo.

Error detection

- Ci permette di controllare se ci sono stati errori
- Quindi, nel caso, chiederemo una ritrasmissione dei dati.

Ma....

- Non sempre è sufficiente...!

Ergo..

- Error detection è utile quando il canale ha pochi errori (è molto affidabile), oppure quando un errore non è critico.
- Ma spesso serve anche fare error correction, cioè cercare anche di correggere gli errori.

Error correction

- Anche perchè, nelle reti, quando invece il canale non è molto affidabile, ritrasmettere troppo pacchetti diventerebbe oneroso: facendo direttamente correzione degli errori sui pacchetti, si

evita la ritrasmissione.

Error control: distanze

- Prima di procedere, ci occorre una misura del “danno” che un errore può fare.
- Gli errori meno gravi sono quelli che danneggiano un solo bit del messaggio.
- Poi seguono quelli che ne danneggiano due, etc etc

La distanza

- Diciamo allora che due messaggi (di lunghezza uguale ovviamente) sono distanti 1 se sono diversi solo per 1 bit.
- Distanti 2 se sono diversi per 2 bits, etc etc
- Questa misura di distanza si chiama **distanza di Hamming**.

Sono quindi più o meno potenti a seconda di quanta.....

Error detection

- Vediamo allora cosa si può fare nel caso dell'error detection
- Ovviamente, dovremo avere dell'informazione extra rispetto ai dati (come nel caso del framing) che ci permetta stavolta di trovare gli errori.

Esempio

- Una semplice tecnica è quella del bit di parità (parity bit):
- Ogni tot bits (diciamo, m), inseriamo uno 0 o un 1 a seconda che la somma degli m bits precedenti sia pari o dispari.
- M = 3 : 010 → 0101 101 → 1010

Quanta ridondanza?

- Quanto stiamo “sprecando” per controllo dell'errore?
- Presto detto: 1 / (m+1)
  (ogni m+1 bits, 1 è per il controllo),
- Il data-rate effettivo sarà dunque 1-1/(m+1) = m / (m+1)
- Ad esempio con m=3 → un quarto del datarate sprecato → tre quarti del datarate.

Quanto è potente?

- Facile vedere che, qualunque sia l'm:
- Presi due messaggi diversi, i loro messaggi codificati sono a minimo distanza 2 (se sono a distanza 1 vuol dire che differiscono per un bit, se poi aggiungo il bit di parità, uno avrà parità pari e uno dispari, quindi differiranno anche per il bit di parità, quindi alla fine avrò distanza 2).
- anche se c'è un errore (1bit), un messaggio codificato non può mai diventare un altro messaggio!

Lezione 2 (15/02/2011)

Il fattore “m”

Possiamo quindi avere un errore (un bit “flippato”) ogni m+1 bits → possiamo identificare un **error rate** di **1/(m+1)**.

Il codice migliore di controllo si ha con **m=1** (cioè 1 bit di frame, grandezza frame = 1): in quel caso, possiamo identificare un error rate di un mezzo (cioè, del 50%).

Però, abbiamo tirato la coperta: il datarate è diventato m/(m+1) = 0,5 dimezzato!! Controllo dell'errore alto ma datarate **dimezzato**!!

Ad ogni modo

- Il codice “parity bit m “ ha potenza 1: significa che non appena c'è un errore di “potenza”???????????

Error correction

- Supponiamo ora di non accontentarci di “sapere” se c'è stato un errore, ma anche di volerlo correggere. Si può fare?

- Per capire se e come, vediamo i cosiddetti “**repetition codes**”.

I repetition codes

- I repetition codes sono semplici!
- Nel repetition code R di N: ogni bit si ripete per N volte
- Quindi ad esempio, (prende un bit e lo ripete N volte)
  R3: 010 → 000111000
  110 → 111111000
- R2 è il codice a parità di bit quando il frame è 1.

La potenza dei repetitions

- E' facile vedere che per ogni due messaggi diversi, i loro messaggi codificati con RN sono distanti N.
- Quindi, con EN possiamo fare error detection fino a potenza N-1.

Quindi...

- Possiamo alzare la potenza quanto vogliamo!!

Allora....

- possiamo usare questo codici per fare anche error correction?

Vediamo...

- Error correction significa, dato un valore sbagliato, farlo tornare al valore corretto.
- Se avessimo un valore sbagliato (trovato tramite l'error detection con RN), dovremmo decidere a che valore corretto riportarlo: come fare?
- Potremmo ad esempio portarlo al valore corretto “più vicino” , secondo la distanza di Hamming.
- R10 corregge massimo 4 errori.

Quindi?

- Non dobbiamo avere messaggi distanti <=2K
- Ma sappiamo che in RN i messaggi sono distanti minimo N
- Quindi ci basta porre N > 2K
- E quindi ricavare il K massimo è facile K = N / 2 – 1 se N pari,
  (N-1) / 2 se N dispari.
- Ad esempio, abbiamo che con R3, il K massimo è (3-1)/2) = 1
- → R3, oltre a fare error detection a potenza 2, è un codice di error-correction di potenza 1 (può correggere il messaggio senza bisogno di ritrasmissione quando un bit arbitrario viene flippato).
- Piccolo problema anche qui: si paga un bel prezzo: con RN, il data-rate diventa 1/N :-(

Abbiamo visto

- Un esempio molto semplice di error detection (parity bit m)
- Un esempio molto semplice di error correction (RN)
- Nella pratica?

Nella pratica

- Codifiche di questo tipo si usano dappertutto:

Built-in error detection

- Ad esempio, in molti casi l'informazioen extra per l'error detection viene aggiunta all'inizio, e fa parte dei dati stessi.
- In tal modo, c'è error detection indipendentemente dal metodo di trasmissione.

Esempio: le carte di credito

Il codice di Luhn

- Funziona analogamente al parity bit code.
- Funziona in base 10, quindi è comodo da usare anche per umani

- Come si fa: si raddoppiano le cifre dispari (contando da destra a sinistra).
- Si fa poi la somma delle singole cifre.
- La cifra di Luhn, che si aggiunge, è quanto manca per arrivare a un multiplo di 10.

Esempio:

- Numero 537
- Raddoppiamo le cifre dispari:
- 10 3 14
- Facciamo la somma delle singole cifre: 1+0+3+1+4 = 9
- Cifra di Luhn: quanto manca per un multiplo di 10 → 1
- → Il numero codificato è 5371

Error detection

- L'error detection si fa nel modo ovvio, ribaltando la procedura:
- Si raddoppiano le cifre pari. Si fa la somma delle singole cifre
- Se il risultato è multiplo di 10, allora ok, altrimenti c'è un errore.

Che proprietà ha il codice di Luhn?

- Non è così difficile, ma si può dimostrare che è un codice di errore detection di potenza 1, cioè trova tutti gli errori nel caso che una cifra sia sbagliata.
- Di più: trova anche praticamente tutti gli errori nel caso due cifre contigue siano state invertite (**trasposizione**), es. “25” invece che “52” (tutti tranne “90” ← → “09”).

In pratica...

- I produttori di carte di credito (Visa, Mastercard) partono col loro numero di 15 cifre, e usano il codice di Luhn per creare il numero a 16 cifre che usano direttamente nella carta.
- In questo modo fanno error detection sul 100% degli errori per cifre singole, e sul 98% degli errori di trasposizione.

Ricordate come funziona il GSM?

- Evoluzione di 1G: il numero identificativo del cellulare viene inviato alla stazione
- Il cosiddetto IMEI (International Mobile Equipment Identity)

IMEI...

- Proviamo: *#06#
- IMEI di 15 cifre (o IMEISV di 17).
- Usa error detection con Luhn!!

Altri esempi pratici:

- i codici ISBN per i libri (ISBN10).
- Si costruisce la sequenza a scala.
- S1 S2 S3 …
  data dalle somme parziali delle cifre dell'ISBN (da sx a dx)
- La somma dei due ultimi S deve essere multiplo di 11.
- Esempio: 0130384887
- Sequenza delle S: 0 1 4 4 7 15 19 27 35 42
- Somma delle ultime due: 35 + 42 = 77 (è multiplo di 11, corretto!)
- I codici a barre (**UPC** = Universal Product Code)
- (somma delle cifre dispari) x 3 + (somma delle cifre pari) == multiplo di 10
- Esempio: Fusilli Voiello: 076810 500407 → 26 x 3 + 12 = 90 OK
- **ISBN13**:
- C'è un nuovo tipo di codici ISBN dal 2007: ISBN13
- Usa lo stesso codice di error detection usato dai codici a barre.
- Numeri di matricola delle automobili

- Codici fiscali

Torniamo ai codici di error correction
Abbiamo visto i codici di ripetizione → sono inefficienti e sprecano banda

I codici di Hamming

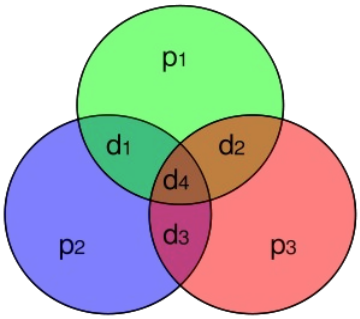
- Sono una famiglia di codici famosissima, fanno parte dei cosiddetti codici lineari perchè le loro operazioni si possono esprimere tramite combinazioni lineari (si usano solo addizioni e moltiplicazioni, amate dai computer).
- Sono sempre codici a blocco, che usano i bit di parità, solo, in modo più efficiente.

Il codice Hamming (7,4)

- Il più semplice e conosciuto è il (7,4).
- Il primo numero dice quanto grande è il frame dopo che ci ho aggiunto il bit di parità
- Un codice Hamming (X,Y) significa che codifica Y bits di dati usandone X in totale.
- Quindi con X-Y bit extra (in questo caso 3 bits).
- Esempio: 0100 → **0100**101

Qual è l'idea?

- I bit extra si “dividono” equamente la parità dei bit dati:



- L'encoding avviene con matrici di zeri e uni; si può scrivere linearmente usando una matrice, che si chiama matrice generatrice:
- G= [**1 0 0 0 0 1 1**]
  [**0 1 0 0 1 0 1**]
  [**0 0 1 0 1 1 0**]
  [**0 0 0 1 1 1 1**]
- Quella in blu è la matrice identità.
- (0100) → 0100**101**
- **2^4 = 16 codici iniziali**
- **2^7 = 128 codici possibili**

Che proprietà hanno questi codici?

- Fanno parte della famiglia più grande, come detto, dei codici lineari (sottospazi lineari di uno spazio vettoriale su campi finiti).

Vediamo le loro proprietà

- Diciamo che il peso (weight) di un messaggio binario è il conto di quanti 1 ha.

Vale allora il...

- **Teorema del peso minimo:** se il peso minimo dei vettori della matrice generatore X per Y è d... allora ogni coppia di messaggi codificati dista almeno d.
- → codice (X,Y,d)
- → conseguenza: il codice corrispondente è un codice:

Ad esempio, Hamming (7,4) ha potenza?

- Peso minimo: 3 → (7,4,3)
- Correzione massima a potenza max = 1

Notare...

- Quanto abbiamo guadagnato: l'altro codice a potenza di correzione 1 che conoscevamo era R3, che aveva come data rate 1/3 = **0,33**
- Qui invece il data rate è 4/7 = **0,57**!!!

Lezione 3 (16/02/2011)

Bella proprietà

dei codici lineari, e quindi anche di quelli di Hamming

L'error detection si fa usando ancora operazioni lineari: una moltiplicazione per una speciale matrice, la **matrice di parità**.

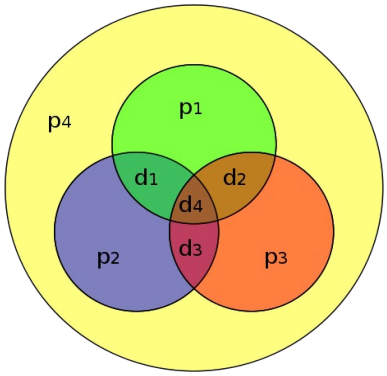
H := [1101100]
3,7 [1011010]
[0111001]

E l'error recovery?

- C'è ovviamente il solito metodo del "più vicino"
- Però pensate alla complessità algoritmica!
- I codici lineari sono molto belli perchè l'error recovery si fa in modo ***molto più veloce*** (usando la cosiddetta *sindrome*!)

Hamming superiori: gli Hamming "ibridi"

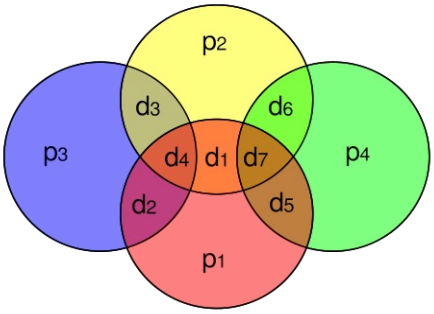
- Hamming (8,4): molto semplice, si aggiunge un altro parity bit



Proprietà di (8, 4)

- La distanza minima è aumentata, stavolta a **4** (è un (8,4,4))
- Quindi ***error-detecting 3*** ed ***error-correcting 1***.
- Il data rate però è sceso: da **0,57** a **0,5**

Hamming superiori: (11,7)



Proprietà:

- Distanza minima: 5 → codice (**(11,7,5)**)
- Quindi, corregge fino a 2 errori
- Data rate?
- Hamming (7,4): **0,57**
- Ora: 7/11 (circa **0,637**)

Nota...ma Hamming come ha fatto a inventarli?

- Ecco il motivo... (1950!), schede perforate!

Notare come...

- Gli Hamming sono una generalizzazione dei codici che abbiamo visto prima (parity bit)!

Esempio: i parity-bit...

- Il parity bit 3 (ogni 3 bit, inseriamo un parity bit P):
- **(a b c) → (a b c P)**
- Possiamo usare la matrice generatrice
- 1 0 0 1
  0 1 0 1
  0 0 1 1
- Notare, peso minimo 2 → codice (4,3,2)

Esempio: i repetition code

- R3 (“balbettiamo” tre volte)
- **(a) → (a a a)**
- Possiamo usare la matrice generatrice
  1 1 1
- Notare, peso minimo 3 → codice **(3,1,3)**

Vediamo ora il problema del burst

Gli errori...

- Se non sappiamo nulla degli errori, non c'è molto che possiamo fare
- (nulla per dire...)
- Ma se conosciamo delle proprietà degli errori, cioè ***se gli errori hanno una struttura***, allora le cose cambiano

I burst

- Nella pratica, in molti casi gli errori non sono del tutto casuali, ma occorrono in ***burst*** (“esplosioni ravvicinate”)
- Ovviamente, questo crea ***seri problemi*** a ogni codice error-correcting!

E quindi...?

- Potremmo tenerne conto e anzi ***sfruttare la loro struttura*** a nostro vantaggio!

Il metodo della matrice invertita (**interleaving**), cambio l'ordine di trasmissione dei bit, in modo da trasmettere i frame in modo diverso; se arriva un burst, l'errore è “spalmato” su più frames.

Il vostro computer

- Nel suo piccolo, c'è anche qui comunicazione dei dati: dal processore alla RAM al disco rigido etc etc.
- Però, le componenti sono molto più affidabili che non una rete esterna, quindi non dobbiamo preoccuparci.

RAM

- Consideriamo ad esempio....

Unità di misura

- Ci serve un'unità di misura grossetta: ad esempio, il quadrilione: 1000000000000000 = 10^15

Dati del vostro computer

- RAM che va a 1600MHz
- 1GB di RAM
- Supponiamo, la probabilità di un errore nella RAM sia bassissima: uno ogni **cento quadrilioni** (!).
- Ogni quanto in media ci sarà un errore?

Dati in RAM

- Un errore ogni 0,08 secondi

E un doppio errore?

- Senza tenere conto dei burst, un doppio errore circa ogni **8 quadrilioni di secondi**.

- **Scelta di design**: se vogliamo migliorare il nostro sistema, dovremmo tenere ben conto degli errori di **potenza 1**, e possiamo tralasciare quelli di **potenza 2**.

E per i dischi rigidi?

- La situazione non cambia di molto (fate il conto): l'access rate è di certo molto minore della RAM, ma non di molti ordini di grandezza, il che significa che ci saranno errori in termini di anni invece che di secondi.

E allora?

RAID

- Redundant Array of Inexpensive (or sometimes Independent) Disks.

RAID0

- Overlay (striping), senza codici di errore

RAID1

- Il codice R2

RAID2

- Overlay a livello di bit con Hamming code
- Dà un altissimo livello di potezione, ma costa molto (per ora).

RAID3

- Overlay con il codice parity

RAID4

- Codice parity

RAID5

- Codice parity distribuito

E così via...

- RAID6,7,1+0,50,51,53,100,3+0,0+1,.....

E la RAM?

ECC RAM

- Usa un codice di error correction, variante di quello di Hamming, di tipo (72, 64)
- Se vi siete mai chiesti perchè le RAM di nuova generazione non andavano più veloci delle vecchie: hanno una penalità dell'11% (il data rate è l'89%).

SECDED

- Attenzione che la scheda madre deve supportare la funzionalità:
- Anche se trovate scritto nelle specifiche che “supporta le ECC”, significa solo che potrete usarle ma non necessariamente che ci sia error correction.
- → deve supportare **SECDED**: Single Error Correction, Double Error Detection.

Altro esempio...

- Le schedine di memoria....
- ECC con Hamming

Andiamo oltre: le missioni Voyager (1977)

- In tutte le trasmissioni spaziali, anche le più recenti, si fa uso di codici di error correction!
- Vediamo come la stessa immagine poi usa error correction!

Lezione 4 (17/02/2011)

Che codice si è usato? (per trasmettere la foto del pianeta Giove)

Il codice **Reed-Solomon**

Essenzialmente...

- E' basato su ***aritmetica polinomiale***, che vedremo meglio quando parleremo dei codici CRC.
- L'idea intuitiva è che si usano i resti di una divisione come bit di parità extra: una divisione “speciale” però perchè fatta con polinomi e non con numeri.
- Fa sempre parte dei ***codici lineari***.

Potenza di Reed-Solomon

- Un codice **RS(X,Y)**, al solito, codifica **Y** parole usandone **X**.
- → può correggere **(X-Y) / 2** errori. (es. se aggiungo 4 bit di parità, riesco a correggere 2 errori, molto potente)
- Inoltre, ha anche altre proprietà interessanti che vedremo fra poco.
- Si usa anche in tantissimi altri contesti, che a differenza delle missioni spaziali sono molto più vicini a noi: CD-ROM ad esempio...
- I CD-Rom usano Reed-Solomon per preservare i dati.

Nei CD!

- Anche nei CD si usano i codici...non ci sono semplicemente i dati musicali memorizzati.
- Si usa come codice di error correction proprio Solomon-Reed!
- In una sua applicazione a due canali, che sfrutta la ridondanza stereo.

Il passo oltre Hamming

- Uno dei grandi aspetti del codice di Solomon-Reed è che va oltre: invece che singoli bits come unità di base, può considerare gruppi di bytes (in generale, di blocchi).
- → permette ad esempio nei lettori CD di sopportare errori di burst lunghi fino a **4000bit**.

Ancora: le Erasures

- Sono errori ma non nel senso di corruzione del dato, ma proprio di cancellazione del dato: il dato non c'è più.

Reed-Salomon e le erasures

- RS(X,Y) corregge fino a X-Y erasures, quindi il ***doppio*** degli errori.
- In generale, può correggere anche errori ed erasures contemporaneamente: gli errori contano doppio delle erasures, quindi **2*errori + erasures < X – Y**

Altri usi di Reed-Salomon

- Nei DVD...
- Nelle ADSL...
- Nel Blu Ray (addirittura “doppio”)
- Nel WiMax

Altri codici

- Ci sono molti altri codici, ma tornando all'ambiente aerospaziale, merita menzionare al volo quello usato nel **Mariner 9** (viaggio su Marte).
- Il ***codice di Hadamard***, basato sulle care matrici che abbiamo già visto nel caso del party internazionale (CDMA e telefonini).

Errori o dati? L'eterno dilemma...

Limite error-rate vs. data-rate? Vediamo con gli Rn

Quindi...

- Man mano che vogliamo ridurre l'error-rate a zero...anche il data rate va a zero.

Non si scappa dal buco nero....

- anche detto **No Free Lunch Principle (NFL)**; nella vita non c'è il pasto gratis.

Shannon, ancora lui

- Ha dimostrato che invece, anche se sembra impossibile, la curva dei codici migliori riesce ancora ad avere un **data-rate positivo**, anche se si vuole un **tasso di errore sempre più piccolo.**
- “anche se arriva Einstein moltiplicato Spock al quadrato...” by MM

Il Teorema di Shannon

- Abbiamo visto un analogo nello strato fisico, qui è più potente:
- Dato un certo tasso d'errore x, vi dice che ci sono codici che possono arrivare ad un ***data rate massimo*** pari all'***entropia*** del canale,
  $H2(x) = x \log (1/x) + (1-x) \log (1/(1-x))$

Conseguenze

- Dischi rigidi “scassoni” del vostro computer con tasso d'errore **0,1**
- Però volete un tasso di errore decente, ad esempio ogni **10^-15**
- → usando tecniche classiche tipo RAID (senza controllo dell'errore, quindi solo ridondanza), servono: **60 dischi**.

Shannon!

- Shannon vi dice che in realtà, c'è un codice per cui per passare dall'errore 0,1 a 10^15 bastano meno di 60 dischi: ne bastano (usate la formula dell'entropia, H): **DUE (!!!)**

Quali codici si avvicinano?

- Vediamo un esempio: i codici sparsi LDPC
- LDPC = Low Density Parity Check
- Uso: TV Digitale, Wi-Max
- LDPC: check, sempre con matrici lineari

Potenza

- MOLTO vicina al limite di Shannon
- Dove si paga?
- Si paga nel seguente senso:
- Il ***decoding*** è **NP-COMPLETO**.

Vediamo ora...

- Un codice che ha un funzionamento simile a Reed-Solomon, solo che è molto più semplice perchè fa solo error detection, e non error correction.

CRC

- Sta per **C**yclic **R**edundancy **C**heck (controllo di ridondanza ciclico)
- Basato sull'aritmetica polinomiale in base 2
- Tecnicamente, il GF(2)[] (l'anello dei polinomi sopra il campo finito con due elementi).

Numeri come polinomi

- Possiamo vedere ogni numero binario come il ***corrispondente polinomio*** in GF(2)[], considerando ogni bit come il coefficiente di potenze sempre crescenti.
- Esempi:

1011 → x^3 + x + 1
11010 → x^4 + x^3 + x

Come funziona l'aritmetica in GF(2)[]?

- Molto semplicemente!
- L'addizione è come fare lo **XOR**!
- La sottrazione è lo stesso dell'addizione!

Proviamo una divisione

- x^4 + x^2 + 1
  ----
  x + 1
  = x^3 + x^2, con resto 1 (il GF(2))

Vediamo:

X+1 | X^4 + 0 + 1
X^4 + X^3

X^3 + X^2
X^3 + X^2
---
0 + 0 + 1

proprio x^3 + x^2 con resto 1

L'idea di CRC

- E'simile a quella dell'algoritmo di Reed-Solomon, e di una varietà di algoritmi simili

Come si fa?

- Si sceglie un polinomio, il cosiddetto ***polinomio generatore*** (diciamo, ***G(x)***)
- Abbiamo un ***messaggio M(x)***
- Potremmo dividere M(x) per G(x), calcolare ***il resto R(x)***, e quindi fare l'encoding trasmettendo **M(x)** seguito da **R(x)**.
- Il problema è che se M(x) è "***minore***" di G(x), allora il resto sarà M(x) stesso, quindi non abbiamo essenzialmente fatto nulla.
- Per risolvere questo problema, allora, moltiplichiamo il messaggio iniziale per x^(grado(G(x)).
- Moltiplicare per x^r un polinomio equivale, ovviamente, a fare lo shift a sinistra di r posizioni.

Quindi....

- L'encoding è semplice:
- Dividiamo x^r * M(x) per G(x)
- Calcoliamo il resto R(x)
- E poi trasmettiamo M(x) seguito da R(x)
- ...che equivale proprio a **x^r * M(x) + R(x)**
- **Esempio:**
  Devo trasmettere 110 11111 (x^4+x^3+x^2+x+1)
  110 = x^2 + x → x^6 + x^5
  R(x) = x^3 + 1 (=1001)

M(x) seguito da R(x): 110 1001 che in polinomio è x^6+x^5+x^3+1

Ma...

- x^r * M(x) + R(x)
- per le magiche proprietà di GF(2)[] è lo stesso che....
- x^r * M(x) – R(x)
- Quindi cos'abbiamo fatto? Abbiamo preso un numero (x^r * M(X)), diviso per G(x), preso il resto e sottratto il resto.
- → abbiamo un numero divisibile per G(x)!

Decoding

- Il decoding è banale: **si divide per x^r** (in altre parole, si taglia via la parte che avevamo aggiunto, con uno shift a destra).

E l'error detection?

- L'error detection è semplicemente prendere il numero trasmesso, e calcolare il resto della divisione per G(x)
- ...che per quanto detto prima, dovrebbe essere **zero**
- Quindi, se il resto è **0**, tutto ok, altrimenti c'è stato un ***errore***.

Potenza?

- Qual è la potenza di un tale metodo?
- Supponiamo ci sia un ***errore***, sono bits che sono stati invertiti
- In altre parole, per l'aritmetica di GF(2)[] è lo stesso che sommare un polinomio di errore
- Esempio:
  110 = x^2 + x
  Ora c'è un errore e il messaggio diventa 111 = x^2 + x + 1 = (x^2+x) + (1)
  110 = 010 = x = (x^2+x) + x^2

Allora:

- Abbiamo il polinomio trasmesso, T(x), più un **polinomio di errore** E(x), e calcoliamo il resto della divisione per G(x):
- (T(x)+E(x)) mod G(x) = (sappiamo che T(x) mod G(x) = 0) → = E(x) mod G(x)
- Quindi, non riusciamo a trovare l'errore solo quando E(x) mod G(x) = 0 ← → **E(x) è divisibile per G(x)**.

Settimana 5

Lezione 1 (21/02/2011)

Singoli errori...

Abbiamo **E(x) = x ^ i (dove i è l'indice della cifra che viene alterata)**

→ basta che G(x) abbia due o più termini

Doppi errori

- **E(x) = x^i + x^j**
- = x ^ j * (x^(i-j)+1)
- → basta che G(x) non sia multiplo di x, e che non divida x^k+1 per ogni k fino al massimo valore di i-j

Ogni errore con un numero dispari di bits

- Basta che x+1 sia un fattore del polinomio (deriva dagli zeri dei polinomi).

I burst errors

- **E(x) = x^i (x^j + … + 1)** è il burst error di lunghezza j +1
- → basta che G abbia grado > j, e termini con +1, e correggiamo tutti i burst lunghi fino a j+1.

→

- In generale (“+1” etc...), buona scelta
- Vaste opzioni: possiamo ***combinare*** polinomi per avere il meglio che ognuno ci offre.

Oppure

- Sceglierne (per quanto possibile) di ***irriducibili***

In generale

- All'aumentare del grado di G, aumenta la potenza: ogni burst error di lunghezza arbitraria più grande del grado di G.
- → probabilità
- **0,5 ^ grado(G(x))**
- esempio (le cose vanno male): E(x) = G(x) * QCA(x)
  G(x) = x^g+x^g-1+...+1 = 11...1
  101101110011111

Cosa vuol dire in pratica?

- Esponenziale col grado di G....
- Potenza dell'esponenziale (sempre sentito nominare, ma in pratica...?)

Pieghiamo...

- Un foglio di carta ha protezione 1, se lo piego in 2, aumento lo spessore e ho protezione 2, continuo a piegare....24 volte? Alto come l'Everest
- ...42 volte? Distanza Terra-Luna
- ...94 volte? Grande come l'universo

Tipi di polinomi

- Vediamo un po' di polinomi e loro usi

CRC-1

- X+1, è uno dei più semplici
- → è il codice parity bit 1

CRC-5

- X^5+X^2+1
- Sono irriducibili

CRC-16

- X^16+X^15+X^2+1
- Dove si usano? Nello standard USB!

CRC-16CCITT

- X^16+X^12+X^2+1
- Dove si usa? Bluetooth

CRC-32

- X^32+X^30+X^26+X^25+X^24+X^18+X^15+X^14+X^12+X^11+X^10+X^8+X^6+...+1
- (Corregge tutti i burst fino a 32, e tutti i burst che alterano un numero dispari di bit)
- **Dove si usa?**
    - Modem v.4
    - Formato **.zip**
    - Standard FDDI (trasporto in fibra ottica)
    - CD-ROM, fornisce la base per Reed-Solomon
    - Ethernet
    - PNG

Passiamo ora...

- All'altro lato della medaglia, nello strato data link
- Abbiamo detto: gestisce gli errori
- ...ed i flussi (flows).

Servono quindi delle regole

- I protocolli, come detto, sono le regole tramite cui si passa un flusso di dati da un punto all'altro di una rete.
- I protocolli si occupano (oltrechè dell'error control) anche del flow control cioè anche del controllo del flusso.

Il problema del flow controllano

- In un canale semplice, se inviamo "***troppi dati***" rischiamo che il ricevente non riesce a gestirli, e che quindi vadano persi.
- In questo caso l'error control dentro ai dati non serve a nulla: anche se il dato arriva correttamente, è il ricevente che non riesce a gestirlo.

Soluzioni?

- Protremmo disegnare la rete in modo da inviare i dati al "giusto" data rate.
- Quest'approccio però è ovviamente limitativo, perchè la capacità del ricevente e/o della linea possono cambiare nel tempo.

Quindi...

- Occorre necessariamente andare su un altro tipo di approcci, dove l'unica soluzione possibile è ***far parlare il ricevente con chi manda i dati***.

I protocolli stop-and-wait

- Sono una famiglia di protocolli in cui si introduce appunto il concetto del parlare tra ricevente e chi invi i dati:
- L'idea è semplice: si manda un blocco dati (un frame), e poi si aspetta (stop and wait) che il ricevente gli mandi un messaggio di conferma (ack), segnalandoci che possiamo inviarne un altro.

Stop-and-wait

- **Vantaggi:** per implementare un protocollo di questo tipo basta un canale half-duplex, visto che non c'è mai comunicazione contemporanea.

- **Svantaggi:** abbastanza lento, più un altro grande svantaggio che adesso vediamo...

Introduciamo ora...

- Un certo signore...Mr Murphy!

Supponiamo...

- ***Di dover lottare con la legge di Murphy***: cioè che nel canale di comunicazione ci sia qualche errore.
- Ovviamente abbiamo l'armamentario dei codici di errore, non c'è problema!
- Quindi, il protocollo andrebbe modificato in questo modo: il receiver invia il messaggio di conferma solo se il pacchetto ***supera il controllo dell'errore***, altrimenti lo ignora.

Allora...

- Ecco cosa può succedere:
- Mandiamo un pacchetto...
- Murphy interviene, e ci corrompe il pacchetto...
- ...il receiver se ne accorge tramite error detection, e quindi non invia il messaggio di conferma....
- Aspettiamo.....infinitamente.

Morale:

- Murphy ci ha fregato!

Ovviamente...

- Così non funziona!
- Soluzione?
- Introduciamo un ***timeout***: se entro un certo periodo di tempo non riceviamo messaggio di conferma, rimandiamo il pacchetto!

In questo modo...

- Noi mandiamo il pacchetto...
- Murphy interviene, e ci corrompe il pacchetto...
- ...il receiver se ne accorge tramite error detection, e quindi niente conferma....
- ...e noi aspettiamo...e aspettiamo....
- ...finchè il timeout è scaduto...
- ...a quel punto rimandiamo...
- ...e il pacchetto arriva!

Tiè a Murphy!

- Vinciamo noi tramite l'idea del timeout: quando non possiamo governare gli eventi; possiamo sfruttare il tempo inserendo il timeout.

Ah, il giorno dopo...

- Noi mandiamo il pacchetto....
- Murphy ormai sa che non c'è niente da fare con noi, e non ci corrompe il pacchetto....
- ...al receiver arriva il pacchetto integro, e quindi ci rimanda il messaggio di conferma....
- ...stavolta però Murphy ritorna all'attacco, e distrugge questo messaggio...
- ...aspettiamo, aspettiamo...scatta il timeout

A quel punto

- A noi non la si fa: timeout scaduto, rimandiamo il pacchetto!
- M. stavolta sta a guardare...
- E il pacchetto dati arriva al ricevente.
- Vediamo la situazione al replay.
- Morale: al ricevente il flusso di dati arriva con dentro due volte di fila un pacchetto dati.

Morale:

- Mai sottovalutare Murphy...!!!

Quindi, non è così semplice

- Un modo per cavarcela da questa brutta situazione è aumentare il carico del messaggio (il frame) non solo con l'informazione per il controllo degli errori, ma anche con informazione sul controllo di flusso stesso.
- Ad esempio, numeriamo i pacchetti, in modo tale da accorgerci se ci sono ripetizioni.

Come si può fare?

- Un modo è appunto attaccare un numero ad ogni frame
- In questo modo se accade la situazione di prima, il ricevente sa che il secondo frame inviato....

Problema

- Quanto lo facciamo grande il numero che attacchiamo ai frame?
- Se il numero è troppo piccolo, rischiamo di non poter trasmettere un flusso di dati che sia più grande.
- Ad esempio, se stiamo trasmettendo un canale televisivo o una telefonata, la durata può essere lunga a piacere.

Pensiamo!

- Pensiamo un attimo a come è fatto il nostro protocollo, e al caso sfortunato:
- In realtà, il problema può avvenire solo fra pacchetti che sono ***consecutivi***
- Quindi, non ci serve distinguere tra due frame qualsiasi, ma ***solo fra frame contigui***.
- → ci bastano ***due simboli***: 0 e 1; ci basta un bit!

Dunque

- I frames vengono spediti usando un bit extra per il controllo di flusso: 0 o 1 alternati tra un frame pari e dispari.

PAR e ARQ

- Siamo andati oltre il semplice stop-and-wait: famiglie di protocolli di questo tipo, dove si ritrasmette tramite timeout nel caso il pacchetto vada perso, si chiamano:
- **PAR (Positive Acknowledgement with Retransmission)** o anche **ARQ (Automatic Repeat reQuest).**

Duplex?

- In questo tipo di protocolli c'è un verso di trasmissione: c'è uno che manda i pacchetti, e un altro che li riceve
- Il caso generale e più utile è però il caso full-duplex, in cui lo scambio dati avviene in tutti e due i versi.

Lezione 2 (22/02/2011)

Full-duplex

- Quando il canale è full-duplex, possiamo dunque prendere un protocollo “simplex” del tipo visto prima, metterne due a ogni lato della linea, e trasmettere da ambo i versi.
- Ma ci sono modi migliori: possiamo gestire il protocollo di trasmissione in modo più efficiente.

Piggybacking

Infatti...

- Un grosso problema è l'***overhead***: praticamente per ogni pacchetto che arriva, ne circola un altro per la riconferma.

La tecnica del piggybacking

- Invece di essere frettolosi e mandare un messaggio di conferma ogni volta che riceviamo un frame...
- ...aspettiamo, e inviamo il messaggio di conferma non da solo, ma in piggyback, sulle spalle del primo frame dati che stiamo inviando noi.

Piggybacking

- In questo modo stiamo sfruttando la banda molto meglio: essenzialmente il messaggio di conferma arriva “gratis”, con un overhead minimo rispetto ai dati che stiamo trasmettendo (una conferma sono pochi bit, rispetto a trasmettere un intero frame).
- Simile a sfruttare meglio i trasporti, ed evitare che un ***camion ritorni vuoto*** dopo aver trasportato un messaggio...

Notare

- Per sfruttare bene la tecnica del piggybacking, bisogna stare attenti a non aspettare troppo per la ritrasmissione (se ad esempio non stiamo trasmettendo dati).
- → funzione bene quando...?

Funziona bene quando

- ...la comunicazione è abbastanza equilibrata (il datarate è abbastanza equilibrato)

Se non c'è equilibrio...

- A manda un frame a B
- B aspetta di fare il piggyback
- Ad A scade il timeout, quindi ritrasmette il frame a B...
- ...B prima o poi fa il piggyback
- → invece di risparmiare, abbiamo incrementato lo spreco di banda (ho spedito 2 volte un pacchetto invece di una).
- In caso di flussi squilibrati, sto sprecando banda....

Quindi...

- Per un corretto funzionamento del piggyback, ci vuole un corretto bilanciamento di flussi, oppure un calcolo accorto del timeout di ritrasmissione (oppure...vedremo dopo...)

Andiamo oltre...

- In tutti questi tipi di protocolli, la logica era sempre quella di aspettare a trasmettere prima di aver avuto la conferma del messaggio precedente...
- Alle volte questo può essere uno spreco...

Esempio

- Avete una comunicazione satellitare con un GEO
- Tempo di trasmissione: 250 msec
- Banda sul canale: 50 kbps
- Taglia di ogni frame: 1000 bit
- Dopo 20 msec avete inviato il vostro primo frame.
- ...e poi ***aspettate*** per la ricevuta di ritorno....
- Supponendo pure Murphy sia in vacanza...
- Quanto aspettate? 250 + 250 = 500msec (***mezzo secondo!***)
- → su ***520msec***, avete trasmesso per ***20msec***
- → siete rimasti bloccati il **96%** del tempo

Il problema in generale...

- C'è quando il prodotto **bandwidth * round-trip-delay** è grande: i protocolli visti funzionano male, perchè state ***sottoutilizzando*** il canale.

L'utilizzo della linea

- Se il canale ha capacità **C** (bit/s), la taglia del frame è **S** (bits), e il tempo di round-trip è **R**,
- ...potete calcolare l'**utilizzo della linea** nel caso di protocolli con ack:
  S / (S+C*R)
- Se **S < CR**, avete un'efficienza ***minore del 50%***.

Soluzione?

- In analogia: state gestendo trasporto merci su camion dall'Italia alla Spagna

- Mandate un cambion di merci, e non ne mandate altri finchè non è tornato un camion con la ricevuta di ritorno.
- Non fareste così, mandereste più camion, preoccupandovi solo dopo un po' se il primo camion non è tornato.
- Quello che si chiama anche ***pipelining***.

Sliding Windows

- La tecnica delle sliding windows sfrutta proprio quest'idea.
- Invece di essere così apprensivi, vi rilassate un po', e vi preoccupate non quando c'è un solo frame del destino incerto, ma un numero maggiore (***n***).
- Tipicamente, il numero maggiore lo prendete una potenza di due (2,4,8,...) così non sprecate bit.

La sliding window

- Tiene conto di quanto siete stressati: più la aprite, più siete rilassati e più pacchetti lasciate andare senza bisogno di conferma.
- La taglia della sliding window può variare sia per il sender che per il recieiver, dando luogo a vari tipi di protocolli.
- In altre parole, come nella vita, tra due che parlando uno può essere più o meno stressato dell'altro.
- Esempio (n=4, size 1);

I protocolli “Go Back N”

- Si hanno quando la taglia della sliding windows di chi riceve è 1: cioè, quando voi siete rilassati (N), mentre il vostro interlocutore è super-apprensivo.
- Funziona bene quando non ci sono molti errori ma il prodotto bandwidth * round-trip-delay è alto.
- Basta che un pacchetto vada male, e devo rispedire tutto.

Notare

- Il vostro “rischio” è aumentato, perchè nel caso peggiore (Murphy docet...) avete n camion da rimandare.

Il rischio si paga, quindi....

- ...dovete avere altri n camion con lo stesso carico pronti, per fronteggiare il caso peggiore.

Quindi

- → dovete avere un buffer di taglia n frames.
- Assieme a n timer per l'eventuale ritrasmissione.
- Ovviamente, se esaurite il buffer, dovete aspettare e non inviare più camion (frames).

I protocolli “selective repeat”

- Sono quelli in cui anche il vostro interlocutore, finalmente, si rilassa un po', e allarga la benedetta finestra....
- Funziona bene, ha il problema che ora anche il receiver deve stavolta allocare un buffer di taglia la sua apertura di finestra.

Buffers

- Notare (l'abbiamo già detto ma meglio sia chiaro):
- La taglia richiesta del buffer è l'ampiezza ***dell'apertura massima*** della finestra, non la grandezza complessiva della finestra.

Problemi delle sliding windows

- Intuitivamente, a parte lo svantaggio in termini di allocazione di risorse, sembra che per la trasmissione convenga permettere una finestra quanto più aperta possibile.
- Invece, se la taglia è troppo grande ovviamente si torna a fare confusione, (specie se non è garantita la sequenzialità del canale)

Esempio

- Avete una finestra di grandezza (ciclo) 4, e apertura 3
- All'inizio, sender e receiver hanno entrambi 0 1 2 3 (i numeri in bold sono gli slot della finestra corrente)
- Il sender invia i pacchetti.....
- Il receiver manda tutti gli ACK corrispondenti, e quindi avanza la sua finestra che passa da:
- **0 1 2** 3
- a
- **0 1** 2 **3**
- Il flusso di dati finora è P0, P1, P2....
- Murphy ha finito il caffè (era un espresso), e rovina l'ack di P0/0
- Il tempo passa, e il sender si insospettisce visto che non riceve l'ack di P0/0, pensando che Murphy c'entri qualcosa...
- → dopo un po' (timeout), rimando il pacchetto P0, sempre come P0/0.
- Il receiver aveva come finestra **0 1** 2 **3**, quindi accetta il pacchetto P0/0 e manda l'ack corrispondente.
- Il sender riceve anche l'ack di P0/0, e quindi tutto contento sposta anche lui la sua finestra a **0 1** 2 **3**, e manda i pacchetti P3/3, P4/0, P5/1.
- Il receiver aveva come finestra **0 1** 2 **3**
- Murphy distoglie il pacchetto P4/0 e lo rallenta.
- → il receiver riceve prima i pacchetti P3/3 e P5/1
- Che accetta, e quindi avanza la finestra a **0** 1 **2 3** …
- → il flusso di dati è stato P0, P1, P2, P3, P0, P5 (!!!).

Morale

- Il punto è che le due situazioni
- **0 1 2** 3

e

0 1 2 **3**

- hanno uno 0 che si sovrappone, quindi da un passaggio consecutivo all'altro, la posizione 0 resta ambigua, perchè il receiver non sa più distinguere un pacchetto da un altro in base al numero dello slot nella finestra!

Morale 2

- Conviene avere un'apertura che sia al massimo la metà della grandezza della sliding view.

Considerazioni...

- Non è solo sfiga...!
- Murphy ***si può quantificare***!!
- Pensate ai doppi errori nelle RAM, ai condici di errore, carte di credito etc...
- → impatta le scelte di design

La sfiga non esiste...?

Qui...

- Spiegazione informale/semplificata
- Il problema delle reti (informatiche), rispetto a Murphy, è la loro velocità...!
- Per quanto la probabilità di un evento isolato sia bassa (quasi 0), quindi ***Murphy-like***...
- ...quando lo stesso evento si ripete per molte volte, la probabilità sale rapidamente da 0 a 1.

Quindi...

- Visto che il progresso tecnologico ci porta a reti sempre più veloci...
- → gli eventi improbabili di Murphy sono sempre più probabili.

Altri problemi delle sliding windows

- Se vi ricordate, abbiamo detto che quando is fa piggybacking c'è un problema quando il

traffico è ***sbilanciato***, non fa in tempo a mandare l'**ack**.

Soluzione?

- Ci sono comunque delle soluzioni: ad esempio, possiamo avere un altro ***timer supplementare*** anche nel ***receiver*** quando il timer scade, rimandiamo, anche senza piggybacking.

Però...

- Occorre stare attenti, ovviamente, a settare bene la durata di questo timer: in particolare, dovrebbe essere più breve del time usato per ritrasmettere frame.

Lezione 3 (23/02/2011)

I timer del sender

- Gli altri timer da considerare sono quelli del sender: quale scelta fare?
- Se il tempo di ack è poco variabile, allora ci basta settare il timer a un po' di più del tempo medio di attesa.
- Il problema è se il tempo di ack è invece ***molto variabile***.

Che fare?

- Ci sono vari modi per scamparla, vediamone uno:
- Quando il tempo è variabile, si setta il timer del sender a un valore abbastanza largo, però si introducono altri timer nel receiver.

I timer NAK

- Questi timer, per ogni slot della sliding window, tengono conto dell'aspettativa di ricevere il pacchetto col numero corrispondente.
- Se il rischio è troppo alto, si può sollecitare il sender, inviandogli un messaggio speciale: non un **ACK**, ma un **NAK** (**N**ot **A**c**K**).

NAK

- Il NAK è un avviso che per qualche motivo, non abbiamo ricevuto il pacchetto corrispondente
- In questo modo, il sender sa che non serve aspettare col suo timeout generoso: può ritrasmettere subito!
- Un buon NAK può velocizzare la rete!

Esempi di protocolli reali:

HDLC

- HDLC sta per High-level Data Link Control
- Ha alcune varianti (LAP e LAPB)
- Ideato inizialmente dall'IBM
- Vediamo come funziona
- Usa dei frames delimitati tramite il **bit stuffing** che abbiamo visto

Vediamo meglio le componenti

- La parte **Data** è il ***payload***, i dati effettivi
- La parte **Checksum** è calcolata usando CRC

Il frame HDLC

- La parte di **Address** serve per la componente di indirizzamento, se ci sono terminali multipli dentro la stessa rete che devono essere distinti.
- La parte **Control** è quella più interessante.

Control del frame

- Essenzialmente, ci possono essere tre tipi di frame:
- **Information**
- **Supervisory**
- **Unnumbered**

- Il controllo del flusso avviene tramite una ***sliding window*** di ampiezza massima **3 bits** (ciclo di 4 bits).

Il frame Information

- **Seq** contiene il numero del controllo di flusso della ***sliding window***.
- **Next** contiene gli **ack** (in ***piggyback***)
- P/F sta per **Poll/Final**:
- Quando il bit indica **P** (Poll), si chiede al ricevente di iniziare la trasmissione.
- Quando il bit indica F (Final) si segnala che la trasmissione va conclusa.

Il frame Supervisory

- Si occupano della supervisione del flusso di dati
- Il campo **Type** indica i vari tipi di controllo di supervisione.
- **Type 0: ACK** (in questo protocollo, detto RECEIVE READY).
- Si usa quando il flusso è sbilanciato e non si può fare ACK con piggybacking (ricordate il timer extra).
- **Type 1: REJECT**
- È un NAK generalizzato, segnala che vanno ritrasmessi tutti i frame a partire da quello indicato in poi nella sliding window.
- Qui **Next** indica il primo frame.
- **Type 2: RECEIVE NOT READY**
- Questo è qualcosa di concettualmente nuovo: segnala che sci sono problemi di congestione nel receiver, e quindi la trasmissione va bloccata, finchè il receiver non rimanda un ACK.
- **Type 3: SELECTIVE REJECT**
- Questo è il classico **NAK**
- **Next** indica il singolo frame da ritrasmettere.

Il frame Unnumbered

- Usato per ulteriori comandi di controllo, non vediamo tutti idettagli perchè le implementazioni nella famiglia HDLC variano, vediamo solo qualche comando.

I comandi di Unnumbered

- **DISC**: sta per DISConnect, segnala che la macchina sta uscendo dalla rete in maniera definitiva (quindi, diverso dal frame supervisory di tipo 32, che segnalava un problema temporaneo).
- SNRM: è il comando duale, segnala che una nuova macchina è entrata nella rete.
- **SNRM** = **S**et **N**ormal **R**esponse **M**ode → indica un canale asimmetrico, dove il nuovo entrato è meno importante.
- Retaggio storico, quando c'era sempre e solo un serve centrale e terminali locali molto stupidi.
- **SABM**: Set Asynchrounous Balanced Mode
- è il comando (introdotto successivamente nello standard) che invece crea una connessione bilanciata, dove chi entra ha gli stessi diritti degli altri.
- **FRMR: FraMe Reject**
- Indica che è arrivato un frame con una sequenza di controllo non corretta/sconosciuta.
- Infine, anche i comandi di controllo unnumbered possono essere “toccati” da Mr Murphy...
- → c'è bisogno anche qui di ACK's
- → comando dedicato, **UA (Unnumbered Acknowledgement)**
- **Non è numerato perchè la sliding window per gli unnumbered è ovviamente uno.**

Vediamo ora...

- Un altro protocollo, che si usa molto di più dell'HDLC perchè è il protocollo di referenza di Internet per quanto riguarda le connessioni point-to-point.

Point-to-Point

- Sono le connessioni internet dedicate punto-a-punto, ad esempio da router a router, o da singolo utente a provider.
- In Internet si usa il protocollo **PPP (Point-to-Point Protocol)**

Cosa fa PPP?

- Ovviamente, dà un metodo di framing per impacchettare i dati
- Comandi di controllo del flusso per attivare le connessioni, test, negoziazione, chiusura
- Questa parte si chiama **LCP (Link Control Protocol)**
- Metodo per negoziare con lo strato superiore del ***network layer***
- Questa parte si chiama **NCP (Network Control Protocol).**

Vediamo come funziona

PPP

- è stato disegnato cercando di essere quanto più simile a HDLC
- Ad esempio, il delimitatore del frame è lo stesso che HDLC

PPP vs HDLC

- Differenza: PPP usa però ***byte stuffing*** invece che ***bit stuffing***.
- Notate come il protocollo più moderno (PPP) usi la tecnica meno efficiente (byte vs bit).

Il campo Address

- Ha lo stesso significato che in HDLC, e proprio per questo, non si usa: ha sempre il valore costante 11111111 (mantenuto per retro-compatibilità)

Il campo Control

- In teoria, lo standard permette a Control, come HDLC, di fare il controllo...
- In pratica però ***non si usa***, quindi tutti i frame sono di tipo non numerato
- E **Control** ha il valore fisso di 00000011

Notare...

- Dunque, PPP è un protocollo che non usa le tecniche di numbering e ACKs per creare una comunicazione affidabile.

I campi protocol

- In questo campo si specifica il ***protocollo*** che PPP sta implementando.
- Quindi in un certo senso PPP stesso è un ***meta-protocollo***.

Il campo Payload

- Nel campo Payload ci sono i dati, il cui significato dipenderà dal protocollo specificato in Protocol. (la grandezza del pacchetto è variabile)

Il campo Checksum

- C'è il ***checksum*** del frame, calcolato usando CRC (→ PPP fa error ***detection***, ma non error recovery)

Vediamo meglio ora...

- I Protocolli supportati da PPP
- Nel campo Protocol ci possono essere essenzialmente due tipi di protocolli: quelli di negoziazione (essenzialmente che restano nel data link layer), e quindi di livello più alto (***network layer***).
- I primi cominciano col bit **1**, gli altri col bit **0**.

I protocolli di negoziazione

- Ne abbiamo parlato all'inizio descrivendo PPP:

LCP

- Vediamo quali sono i tipi di frame che LCP usa:
- Sono 11:
- 4 di ***configurazione***
- 2 di ***terminazione***

- 2 di ***rifiuto***
- 2 di ***echo***
- 1 di ***test***

Configurazione LCP

- **Configure-request**
- Sender → Receiver
- Propone opzioni per la configurazione della linea
- **Configure-ack**
- Sender ← Receiver
- L'ack di configure-ack (configurazioni accettate)
- Notate come la mancanza del controllo ack nel contenitore PPP viene quindi rimpiazzata da una reimplementazione nel sottoprotocollo LCP.
- **Configure-nak**
- Sender ← Receiver
- Il NAK per configure-request: alcune opzioni non vanno bene, puoi fare di meglio.
- **Configure-reject**
- Sender ← Receiver
- Non c'è nulla da fare, cambia tono con me! (opzioni non negoziabili).

Tra le varie opzioni...

- Ce ne sono alcune degne di nota:
- Ad esempio, per settare il controllo di error-detection a 2 o 4 bytes.

E anche...

- Per settare la lunghezza del campo Protocol (1 o 2 bytes)
- Per settare la lunghezza del campo Payload (default: 1500 bytes)
- Quindi PPP è un protocollo che gestisce frame a ***lunghezza variabile***.

Infine...

- Per evitare sprechi:
- Ricordate che i campi Address e Control sono praticamente sempre fissi?
- Con una opzione, si possono togliere del tutto, risparmiando 2 byes a frame.

Terminazione LCP

- Terminate-request
- Sender → Receiver
- Chiudiamo la comunicazione
- Terminate-ack
- Sender ← Receiver
- L'ack di terminazione-request: va bene, comunicazione conclusa.

Rifiuto LCP

- Code-reject
- Sender ← Receiver
- Non ho capito cosa intendevi, richiesta sconosciuta.
- Protocol-Reject
- Sender ← Receiver
- Non capisco di che protocollo parli (o c'è stato un errore sulla linea, oppure non supporto il protocollo che mi stai chiedendo).

Echo LCP

- Echo-request
- Sender → Receiver

- Per favore rimandami il frame indietro (echo).
- Serve a controllare/misurare la qualità della linea di comunicazione.
- Echo-reply
- Sender ← Receiver
- Eco eco qui il tuo tuo frame frame

Test LCP

- Discard-request
- Sender → Receiver
- Sto testando la linea, ignora il frame per favore

Lezione 4 (24/02/2011)

Da notare...i loops...

- Come fa PPP? Pensateci...

Ciclo di connessione PPP

[IMG 3.28]

PPP e le ADSL

- PPP è anche usato per le connessioni a Internet tramite ADSL
- Le varianti sono:
  PPPoE (PPP over Ethernet) e
  PPPoA (PPP over ATM)

PPPoE: ciclo iniziale

- La connessione all'ADSL inizia così:
- Il vostro computer/modem invia un frame **PPPoE** (Active Discovery Initiation), col suo indirizzo fisico (MAC).
- Ogni servizio ADSL (in gergo, concentratore di accessi, DSL-AC) disponibile risponde con un **PADO** (PPPoE Active Discovery Offer) in cui dà il proprio indirizzo, e si "offre" per la connessione...
- Il vostro computer risponde con un **PADR** (PPPoE Active Discovery Request) in cui segnala il servizio ADSL che ha scelto.
- Il servizio fa l'ACK usando un frame PADS (PPPoE Active Discovery Session-confirmation).
- Alla fine, la connessione è terminata da un frame **PADT**, (PPPoE Active Discovery Termination).

I protocolli multiaccesso

- Finora abbiamo visto il caso point-to-point, in cui c'è uno che parla e uno che ascolta, ed un canale tutto per loro.
- Ovviamente, ci sono molti altri contesti in cui queste assunzioni non sono valide, e ci sono molte entità diverse che vogliono usare lo stesso canale per parlarsi.

Assunzioni?

- Prima di vedere quali sono i problemi del multiaccesso, occorre come al solito stabilire le regole del gioco, cioè che assunzioni facciamo (occorre sempre definire per bene il problema prima di trovare poi le soluzioni).
- **Station Model** ("modello a stazione"): sono le entità che trasmettono. Dopo che hanno iniziato la trasmissione di un frame, non fanno altro finchè non è stato trasmesso.
- **Single channel:** c'è un canale singolo disponibile per tutti.
- **Collision**: se due frame si sovrappongono, c'è una collisione e sono inutilizzabili (quindi, niente cose tipo CDMA in questo gioco).

Assunzioni: sul TEMPO

- O continuo (non c'è un orologio centrale), cioè senza sincronizzazione

- Oppure **slotted** o discreto (a intervalli)

Assunzioni

- Sul ***carrier*** (il mezzo di trasporto):
- **Carrier sense** (una stazione può vedere se il canale è in uso, prima di tentare una trasmissione).
- **No Carrier sense** (una stazione non può analizzare un canale finchè non lo usa); le stazioni non ascoltano, trasmettono senz'altro; si preoccuperanno dopo di vedere se c'è stata una collisione.

I contention systems

- Sono quei sistemi di comunicazione multipla in cui c'è un unico canale condiviso da molti, e si possono creare contenziosi (***contentions***).

Norman Abramson

- Negli anni '70...alle Hawaii c'era e c'è comunicazione via radio; tante stazioni che trasmettono sulla stessa frequenza → disastro, conflitti!
- Idea di un protocollo multiaccesso ALOHA

Aloha

- Sfruttamento del caso
- [IMG 4.1]
- Quanto grande deve essere il dado perchè tutto funziona?
- E se c'è collisione? Me ne accorgo perchè sento anche altre trasmissioni radio.

Probabilità?

- La probabilità che k frames siano generati durante un certo intervallo di tempo, se non ci sono sbilanciamenti, è di tipo **Poisson**, cioè se la media delle trasmissioni nell'unità di tempo è G, allora la probabilità che ci siano k trasmissioni è data dalla distribuzione di Poisson:

La distribuzione di Poisson

- Pr[k] = (G^k * e^-G) / k!
- [IMG Sinusoidal Poisson Distribution con linee blu]
- [IMG 4.2]

Aloha

- Dobbiamo vedere qual è la probabilità che non ci sia un altro frame Pr[0] trasmesso in un tempo doppio dell'unità.
- → la media in quel periodo è 2G
- → la probabilità è **e^-2G**

Quindi

- Quanti frame posso trasmettere nell'unità di tempo?
- → G * e^-2G

G * e^-2G

- E quindi, a quanto mi conviene settare la velocità di tentativi di accesso al canale (G) per massimizzare le prestazioni?
- Il massimo, facile da vedere, si ottiene con **G=0,5**
- **→ 1 / (2e) frames/sec → c**

0,184

- 18,4% di banda... è poco!!
- Però...il datarate è sempre 18,4% e non dipende dal numero di utenti che vogliono trasmettere contemporaneamente

Nota

- Altra variabile possibile: lunghezza del frame random
- Invece, si vede che la miglior scelta è fissare una lunghezza fissa per tutti i frame.
- Motivo intuitivo: abbastanza ovvio, perchè avere frame diversi creerebbe **rotture di**

simmetria (rifletteteci sopra).

Slotted Aloha

- Due anni dopo Aloha, Roberts trova un metodo per migliorarla:
- Assume che il tempo sia slotted, tramite una stazione principale che manda un segnale di sincronizzazione
- In tal modo, una stazione deve attendere prima di trasmettere il pacchetto, non c'è trasmissione istantanea.
- Come cambiano le prestazioni?
- Il periodo “critico” che può generare conflitti stavolta è ***dimezzato***

Quindi...

- Corrispondentemente, il numero di frame al secondo diventa:
  1 / e
- Cioè circa 36,8%

Pure e Slotted Aloha a confronto

- [IMG 4.3]

Notare...

- Abbiamo raddoppiato la velocità, alterando una delle opzioni a nostra disposizione...
- Si può fare ancora meglio?
- Le reti Aloha non erano carrier-sense.

1-persistent CSMA

- CSMA sta per ***Carrier Sense Multiple Access Protocol***
- → prima di trasmettere, controlla che non ci sia già una trasmissione (carrier sense)
- Può lo stesso succedere che ci siano collisioni (Murphy...): in questo caso?
- Si riaspetta un certo **tempo casuale**, e poi si riprova.
- Performance: aumenta a più del **50%**.

Difetto di 1-CSMA?

- Se molte stazioni stanno aspettando durante una trasmissione, alla fine della trasmissione tutte cercano di trasmettere contemporaneamente, provocando una collisione.
- Se fossero un po'meno egoiste, il mondo funzionerebbe meglio...

p-persistent CSMA

- Si ottiene rilassando l'ipotesi che uno debba sempre trasmettere (p=1) quando trova la linea libera, ma invece, trasmette con probabilità p (è più gentile :-) e non deve sempre trasmettere a tutti i costi).
- Performance? Migliora, con molte stazioni al diminuire di p.
- Al limite (vedremo), può arrivare anche al **100% di datarate**.

Nonpersistent CSMA

- Usa un metodo alternativo: invece di tenere d'occhio il canale per ricominciare la trasmissione non appena è libero, se trova il canale occupato aspetta un periodo di tempo casuale e poi riprova.
- Performance: la curva di performance si comporta differentemente da p-CSMA, e può raggiungere performance vicine al **90%**

Settimana 6

Lezione 1 (28/02/2011)

CSMA: confronto

- [IMG 4.4]

CSMA/CD

- Avendo il carrier sense, possiamo anche essere più furbi quando trasmettiamo: se ci accorgiamo di una collisione, smettiamo di trasmettere.

- Quindi, abbiamo rilassato l'operazione di atomicità della trasmissione di una frame.
- Questo tipo di trasmissione dà luogo al CSMA con Collision Detection (CSMA/CD).
- CSMA/CD tratta le collisioni come il CSMA, cioè se c'è una collusione, aspetta un certo periodo di tempo casuale prima di riprovare a trasmettere.
- In CSMA/CD, ci sono tre possibili stati della rete:
    - **trasmissione**
    - **contention**
    - **idle**
- [IMG 4.5 pag 258]

La parte importante

- E' nella parte di contention, cioè decidere quanto tempo allocare a quella zona in cui si decidono le sorti della trasmissione.
- Il tempo da allocare sarà ***due volte*** il tempo di trasmissione tra le due stazioni più distanti (→ il tempo di ***roundtrip***).

Quindi...

- Tipicamente, il periodo di contention ha durata ***il tempo massimo di roundtrip***, e si usano poi tecniche di protocollo multiaccesso (ad esempio, slotted ALOHA) per “vincere il round” ed essere sicuri che il canale è nostro.

Oltre le collisioni

- In tutti i sistemi che abbiamo visto finora, c'è sempre la possibilità che ci siano collisioni
- Le collusioni ovviamente implicano tempo sprecato e quindi inefficienza.
- Vediamo se c'è un modo migliore per ***evitare del tutto le collisioni***!

Protocolli collision-free

- Anche detti CSMA/CA (collision avoidance).
- In questa classe di protocolli, si fa un uso intelligente del contention period per evitare del tutto le collisioni.
- Nel contention period si decide tutto, e poi la trasmissione del prossimo ciclo di frames continua come stabilita dal contention period, quindi senza collisioni.

Basic bit-map

- Un modo è abbastanza ovvio: prendere il contention period, e dividerlo in intervalli uguali per ogni stazione (ricordiamo: niente collisioni!)
- Ogni stazione potrà quindi segnalare nel suo “mini-slot” se vuole trasmettere un frame (fa una prenotazione, quindi è anche un cossiddetto reservation protcol).
- Quanto grande sarà il mini-slot?
- Basta un bit per segnalare la nostra intenzione di trasmettere

Basic bit-map

[IMG 4.6 pag 259]

Efficienza?

- A pieno carico, se la taglia del frame è d, è ovvio che l'efficienza sarà **d / (d+1)**
  (c'è un solo bit “sprecato”)

Problemi

- Il problema del bit-map è che abbiamo un bit per ogni stazione, quindi con ***tante stazioni*** il contention period può diventare molto lungo.
- Ci sono tecniche per migliorare questa componente critica.

Binary count-down

- L'idea è che invece di riservare un bit separato nel contention protocol a ogni stazione (quindi, prima, per N stazioni ci serviva uno spazio di N bits), usiamo una codifica più compatta.
- Qual è la codifica più compatta per rappresentare N stazioni?

- Ovviamente, possiamo usare una rappresentazione binaria, quindi ci servono solo **log in base 2 (N)** bits.
- Pagheremo questa compattezza nel modo di gestione del contention period, ovviamente.
- Allora, ogni stazione ha il suo indirizzo in binario
- Quando vuole trasmettere, fa un OR booleano del suo indirizzo.
- Quindi il contention avrà la lista in OR di tutte le stazioni che vogliono trasmettere.
- Chi trasmette poi il prossimo frame?
- La precedenza l'avrà chi ha il numero di stazione più grande.
- [IMG 4.7 pag 262]

Efficienza

- L'efficienza del protocollo è chiaramente **d / (d + log2(N))**
- Visto che di solito il frame ha già informazione sull'indirizzo della stazione, rimescolando bene le carte si può arrivare anche al **100%** di efficienza (rispetto al datalink layer).

Problema

- Le stazioni sono in priorità, quindi si rischia lo stallo di quelle a più bassa priorità!
- Ovviamente ci sono vari modi per ovviare a questa situazione

Ad esempio

- Quello probabilistico: ad ogni round, ogni stazione si riassegna una priorità random.
- Problema: possibili collisioni, rischio in casi sfortunati di ritardi o in ogni caso di situazione non eque, e ***perdiamo l'informazione***.

Altro modo:

- Manteniamo l'informazione che abbiamo: ogni stazione sa se ha o non ha trasmesso.
- Quindi, può aumentare la sia priorità di 1 se non ha trasmesso
- chi ha trasmesso invece va a priorità 0
- → Si mantiene la separazione delle priorità e si evita lo stallo nella maniera più efficiente ed equa.

Ricapitoliamo

- Finora, abbiamo visto i protocolli tipo CSMA:
- Funzionano bene a carico basso della rete, ma male a carico alto (troppi conflitti → troppa latenza).
- Poi abbiamo visto quelli collision-free:
- Funzionano bene a carico alto, ma male a carico basso (spreco di banda → latenza!)

Dunque...

- La cosa migliore, ammesso che ci si riesca, sarebbe trovare un protocollo che ***unisca*** il meglio dei due mondi, cioè che a basso carico si comporti come un CSMA, e ad alto carico si comporti come un collision-free.

Vediamo un po'...

- Il vero problema di ogni protocollo multiaccesso è ovviamente quello della competizione
- Nel caso CSMA, non abbiamo considerato il numero di stazioni.
- Nel caso collision-free, abbiamo invece considerato il numero di stazioni
- Potremmo allora studiare cosa succede a CSMA quando fissiamo il numero di stazioni.
- Supponiamo il caso migliore, slotted time, e che ogni stazione trasmetta con probabilità p.
- Se ci sono N stazioni, qual è la probabilità che una stazione riesca a trasmettere?

Vediamo...

- Facile: **N * p * (1-p)^(N-1)**
- Avendo un certo numero di stazioni, qual è dunque la scelta di p migliore?
- → differenziamo per p e risolviamo:
- **p = 1 / N**

- Quindi date N stazioni, la miglior probabilità che le cose vadano bene è: **((N-1) / N)^(N-1)**

Vediamolo graficamente:

- [IMG 4.8 pag 263]

Allora..

- Vediamo una cosa interessante: quando ci sono poche stazioni, la probabilità di successo è buona.
- All'aumentare delle stazioni, la curva prende un valore asintotico che noi sappiamo essere **1 / e,** ALOHA!

Hmmm....

- Se la competizione non fosse determinata a priori, ma fosse invece ***dinamica***...?
- Cioè, se avessimo un protocollo che aumenta la competizione quando ci sono poche stazioni che vogliono trasmettere, e la diminuisce quando ce ne sono tante...?

Limited-Contention protocols

- E' una classe speciale, dove si rilassa un po' l'assunzione che tutte le stazioni siano sempre uguali (quindi dal caso simmetrico, passiamo ad uno asimmetrico).

Lezione 2 (01/03/2011)

Comunicazione di servizio: Eurostar 9410; giovedì h15.00 @ Padova in teoria;

Come si fa?

Il trucco: Adaptive Tree Walk Protocol

- [IMG 4.9 pag 264]

Allocazione dinamica

- Quando c'è una collisione, si ***diminuisce dinamicamente la competizione***, selezionando una sottoparte dell'albero.

Ancora meglio...

- Possiamo fare ancora meglio: analizzando il traffico recente, possiamo ad esempio renderci conto di quante sono le stazioni che cercano di trasmettere in quel lasso di tempo...

Se...

- Sappiamo quante sono circa le stazioni attive, inutile sprecare tempo a cercare proprio dalla cima dell'albero.
- Potremmo cominciare direttamente da un sottopezzo dell'albero.

Dove?

- Se siamo in un nodo a profondità **P**, i nodi sotto di lui diminuiscono all'incirca come **2^P**.
- Se le stazioni attive (**A**) sono distribuite equamente, allora ci saranno circa **A / 2^P** stazione attive nel sottoalbero.

A/2^P

- Vogliamo avere il sottoalbero più piccolo che abbia un stazione attiva.
- → A / 2^P = 1
- → **P = log2(A)**
- Quindi ad esempio, se abbiamo circa 8 stazioni attive, conviene cominciare direttamente a profondità 3.

Passiamo ora...

- Al caso wireless
- Nel wireless, ci sono ulteriori difficoltà, dovute ad un fatto fondamentale:
- **LA TOLOPOGIA DI RETE NON E' FISSA MA CAMBIA DINAMICAMENTE**

Il caso Wireless

- Il problema è che non c'è più un singolo canale per tutti, ma varie zone spaziali dove alcune stazioni interagiscono, ed altre no.

- In altri termini, il controllo da ***globale*** diventa ***locale***.

Vediamo

- [IMG 4.11a pag 268]

Problemi...

- **A** trasmette a **B**
- **C** non sente, quindi conclude che può trasmettere a **B**.
- → problema della stazione nascosta (***hidden station problem***)

D'altro canto, dualmente...

- [IMG 4.11b pag 268]
- **B** trasmette ad **A**
- **C** sente la trasmissione, e conclude che non può trasmettere a **D**, quando invece avrebbe potuto!
- E' detto il problema (duale) della stazione esposta (***exposed station problem***).

Soluzioni?

- Varie, ma vediamo una delle più semplici per avere un'idea.

MACA

- Multiple Access with Collision Avoidance
- (esteso poi a **MACAW**), W sta per Wireless
- Essenzialmente, sfrutta l'idea che chi deve trasmettere renda il suo spazio locale "conosciuto" anche agli altri.
- Questa conoscenza locale avviene tramite due comandi:
- **RTS** (Request to Send), che rende nota la volontà di inviare un messaggio, e
- **CTS** (Clear to Send), che è l'ACK

Vediamo: A manda l'RTS

- [IMG 4.12a pag 270]

B risponde col CTS...

- [IMG 4.12b pag 270]

E gli altri?

- **C** sente l'RTS di **A**, ma non il CTS di **B**
- → può trasmettere mentre il frame è inviato
- **E** sente sia l'RTS che il CTS
- → non trasmette finchè la trasmissione annunciata non è conclusa
- **D** non sente l'RTS, ma sente il CTS → anche lui sta zitto

Notare

- MACA non è collision-free, ad esempio se **B** e **D** trasmettono entrambe a **C**.
- E quindi?
- → si usa il solito trucco: aspettare un ***tempo random*** per riprovare dopo.

Passiamo ora...

- ...dopo aver visto i vari tipi possibili di protocolli multiaccesso, a vedere qualche esempio pratico di protocollo in uso.

802.3

- Il primo protocollo che vediamo è lo standard **IEEE 802.3**

Cioè, Ethernet!

- Viene in vari tipi, con nome identificativo "***XBase Y***", dove:
- X è la banda in Mbps
- "Base" indica che è una connessione baseband (a frequenza unica).
- Y è il tipo di cavo
- Abbiamo già visto come le interferenze del caov rendano necessario l'uso di ***ripetitori*** dopo

una certa lunghezza critica.
- I vari tipi di Ethernet differiscono anche, quindi, a seconda della ***lunghezza massima*** di ogni tratto senza ripetitori, caratteristica importante per cablaggio, manutenzione e costi.

Tipi di cablaggio

- A serpente, a lisca di pesce, ad albero
- [IMG 4.15a,b,c pag 274]

L'inventore: Bob Metcalfe

- Studente al **MIT**
- Fan di ARPANET (→ Internet)
- Scrive un saggio, l'AT&T lo contatta per vedere questo nuovo tipo di rete in azione...
- Mentre dà la demo, il sistema **crasha**...
- → l'AT&T si convince che ARPANET non è la via giusta e resta alla linea telefonica...
- Passa poi a fare il dottorato ad Harvard, dove comincia a sviluppare Ethernet.
- La sua tesi viene giudicata ***molto scadente*** perchè il lavoro viene giudicato carente dal punto di vista teorico...
- Passa alla XEROX, dove sviluppa Ethernet (1973)
- **ETHER**-net...! (=etere)

L'Etere luminifero (1887)

- L'etere luminifero è stata una delle più grosse invenzioni della scienza che sono state poi confutate; mentre parlo che c'è del suono che fa vibrare l'aria e ci arriva alle ns orecchie; se attivo un fascio di luce, c'è un materiale che trasmette le onde della luce; questo materiale lo chiamano Etere luminifero. Questo etere si cerca ma non si trova;

Metcalfe

- Il management gli dice che quella tecnologia è destinata all'insuccesso.
- Lascia Xerox e fonda la sua compagnia, 3Com (1979)...
- 1982: prima Ethernet card per PC...
- Il resto lo sapete...

Famoso...

- Anche per la cosiddetta ***legge di Metcalfe***:
- Il valore di una rete è proporzionale...?
- ...***al quadrato degli utenti!***
- Legge per nulla banale...
- Anche per la sua predizione sul collasso di Internet
- 1995: Internet avrà un ***catastrophic collapse*** l'anno successivo
- Promette di mangiarsi la promessa (stampata) se non succederà.

1997

- Alla conferenza annuale del Web, riconosce l'errore...
- E si mangia la predizione...!

10Base5

- [IMG 4.14a pag 273]
- E' la primissima versione, ora obsoleta
- Usava un grosso cavo coassiale
- Con tacche ogni 2 metri e mezzo per inserire altri cavi
- Il “10” significa banda fino a 10Mbps
- Il “5” significa supporta segmenti fino a 500 metri.
- Schemino riassuntivo tattico e poderoso: [IMG 4.13 pag 271]
- Ogni segmento può contenere fino a 100 stazioni

Tipo di connessione?

- La connessione fisica avviene tramite i ***vampire taps***, essenzialmente un pin veniva inserito a forza dentro il cavo coassiale.

10Base5

- Notare il design: il ***transceiver*** (che si occupa del carrier detection e collision detection) era ad ogni giunzione.
- Con le successive evoluzioni invece, il controller è stato spostato dentro al computer,

10Base2

- L'evoluzione di 10Base5
- Fatto con cavo coassiale fine
- Giuzioni a T
- [IMG 4.14b pag 273]
- Transceiver nel pc
- Economico, più affidabile di 10Base5
- 2 → cavo fino a 200m circa
- Ogni segmento può contenere fino a 30 stazioni

10Base-T

- Niente cavo condiviso, ma hubs che usano cavi dedicati per ogni pc.
- [IMG 4.14c pag 273]
- Invece del coassiale, cavo twisted pair (T).
- Poco costoso, ancora più affidabile
- → è diventato il tipo dominante
- **Svantaggi**: la massima lunghezza di ogni segmento scende ancora: 100 metri
- **Vantaggi**: il numero di stazioni per segmento cresce invece, dai 100 di 10Base5 e 30 di 10Base2, a **1024**!

Ricapitoliamo finora...

- 10base5: 500metri, 100 stazioni [200]
- 10base2: 200 metri, 30 stazioni [150]
- 10baseT: 100 metri, 1024 stazioni [10240]

Lezione 3 (03/03/2011)

10Base-F

- Usa la fibra ottica
- Grande vantaggio: permette segmenti fino a ***2km***
- Essendo anche molto fine → il tipo ideale per connessioni tra edifici
- Altro vantaggio correlato: il ***tapping*** è molto più difficile, quindi è un buon metodo per connessioni esterne all'edificio.

Codifica fisica in Ethernet

- Per codificare 0 e 1, non usa 0 volts e X volts, perchè?
- Tra le altre cose per un motivo importante
- Usare 0 volts per il simbolo 0 porterebbe a seri problemi di sincronizzazione

Esempio: inviando **00100000** dobbiamo essere esattamente sincronizzati, altrimenti potremmo confonderci con 01000000 oppure 10000000.

E allora?

- Una scelta possibile potrebbe essere quella già vista ad esempio per il CDMA:
  -X volts per 0, X volts per 1
- Per reti complesse, questa codifica ha lo svantaggio che richiede sempre una buona dose di sincronizzazione, ad esempio per dati con lunghe sequenze di 0 o di 1 (si rischia che il clock

vada fuori fase)

Il Manchester encoding (quello scelto da Metcalfe)

- [IMG 4.16 pag 275]

Svantaggi

- Il Manchester encofing risolve i problemi di sincronizzazione (→ hardware meno costoso), ma....
- Svantaggio: dimezza la banda!

Ciononostante....

- E' usato da Ethernet, anche perchè la sua affidabilità ha poi reso possibile ***incrementare la banda*** senza far esplodere i costi dell'hardware.

I frames di Ethernet

- [IMG 4.17 pag 276]
- Un preambolo...
- ...di 8 bytes!
- Con bytes 10101010...per la sincronizzazione
- L'indirizzo destinazione (6 bytes)
- Di questi, il primo bit a 1 segnala una comunicazione ***multicast*** (a un gruppo).
- Tutti i bits a 1 → ***broadcast***
- Il secondo bit invece distingue indirizzi ***locali*** da quelli ***globali***
- Spazio degli indirizzi globali: 48-2 = **46 bits**

Indirizzi globali e locali

- 46 bits l'uno....
- Un bel po'!
- Che indirizzi sono?
- Li conoscete già: i MAC ADDRESS
- MAC = Media Access Control

Problema

- Sono uno spazio di indirizzi....
- Finchè sono locali, ok, ma...
- ...quando sono globali?
- Come funziona?

I MAC Address

- MAC-48 (in futuro EUI-64): gestiti dall'**IEEE**
- I primi **3** bytes: il produttore
- **OUI** (Organizationally Unique Identifier)
- I secondi **3**: il loro spazio di indirizzi
- Vediamo...
- Durata (circa): fino al 2100...!!

- C'è poi il campo Type, che specifica il tipo di protocollo o in ogni caso l'uso del frame (analogamente a PPP)
- Poi c'è il campo dati: al massimo 1500 bytes (→ frames a lunghezza variabile...ricordate i ben 8 bytes per la sincronizzazione... e intuite come questo ha fatto la fortuna di Ethernet...)
- Il campo Checksum è il classico CRC-32
- → Ethernet fa ***error detection*** ma non ***error correction***
- Infine, il campo Pad, che abbiamo tenuto per ultimo

A cosa serve?

- A rendere efficiente la collision detection

- [IMG 4.18 pag 277]

Quindi

- Pad serve ad assicurare che la lunghezza minima del frame sia almeno il tempo di roundtrip.
- Per Ethernet a 10Mbps → il frame deve essere lungo almeno 500 bits → 512 bits (64 bytes) per sicurezza; 64 bytes quindi è la minima grandezza del frame ethernet.
- → il pad serve ad assicuare la grandezza minima

E quando c'è collusion detection?

- Si usa una tattica simile per certi versi (all'opposto) a quella della sifilide
- Se c'è collisione, aumentiamo esponenzialmente il tempo di attesa massimo, e facciamo ritrasmettere a caso, finchè non ci va liscia.
- Quello che si chiama ***binary esponential backoff***

Analogia con la teoria dei giochi e le puntate...

- Ma con meno rischi...

Backoff...truncated

- In realtà, non si rischia così tanto perchè c'è un limite: dopo 10 raddoppi (collisioni), si mantiene l'intervallo massimo a 1023 slots per altre dieci collisioni, e poi si rinuncia.
- → ***binary exponential truncated backoff***

Efficienza di Ethernet

- Se un frame ci mette in media tempo T per essere trasmesso, l'efficienza media è circa:
- **T / (T + 2* roundtrip / alpha)**
- **T** = tempo medio di trasmissione di un frame
- **roundtrip** = tempo di andata e ritorno
- **alpha** = è la probabilità di trasmissione nel canale
- Sostituendo T = lunghezza frame / bandwidth, abbiamo alla fine che...
- Efficienza = inversamente proporzionale a bandwidth * lunghezza
- Quindi, notate l'effetto perverso: quanto più aumentiamo la banda, o la lunghezza utile del cavo...tanto più cala l'efficienza!

Come se ne esce?

- Una delle due cose andrà sacrificata...
- Visto che alla banda non possiamo rinunciare → essenzialmente serve una rete che abbia lunghezza massima limitata.
- [IMG 4.20 pag 282]
- Invece degli ***hub***, si usano gli ***switch***!

Fast Ethernet

- 10Mbps erano molti, ma al solito, sono poi diventati pochi col tempo.
- → IEEE ha formato due comitati (circa 1992)

Il primo comitato

- Ha volato basso:
- Ha detto: estendiamo Ethernet mantenendo i suoi problemi

Il secondo comitato

- Ha volato alto:
- Ha detto, visto che ci siamo, facciamo un'Ethernet migliore, e con supporto di traffico ***real-time*** e ***voce***.
- Il primo comitato: → **802.3u** (Fast Ethernet), 100Mbps
- Il secondo comitato: ha volato alto... producendo lo standard **802.12**

Fast Ethernet

- L'idea di base era semplice: aumentare il ciclo di clock di un fattore 10
- Per le limitazioni che si hanno sulla lunghezza del cavo, servono però degli switch.

Settimana 7
Lezione 1 (07/03/2011)

Tipi di Fast Ethernet

- **100Base-T4**: usa cavi UTP3
- → lunghezza fino a 100m
- **100Base-TX:** usa cavi UTP-5
- → lunghezza fino a 100m
- **100Base-FX:** fibra ottica
- → lunghezza fino a 2000m :-) :-)
- [IMG 4.21 pag 285]
- Nota: ma allora UTP3 e UTP5 è lo stesso?
- Risposta: No, UTP3 richiede ***4 cavi***!

Ethernet Gigabit

- L'evoluzione 10x (**803.2z**)
- Evoluzione notevole: ora funziona in due modi, di cui uno ***point-to-point*** che sfrutta la presenza di uno switch.
- → ***niente carrier sense***, molto più veloce!

Problemi del Gigabit

- Aumentando la velocità, la lunghezza del cavo diminuisce ancora
- Soluzione 1: usare del padding (dati finti che servivano a far diventare il pacchetto sufficientemente grande) ulteriore...
- Svantaggi: l'efficienza cala fino al **9%**.
- Soluzione 2: invece di trasmettere un frame solo, si trasmettono blocchi di frame (***frame bursting***), come se fossero un unico superblocco.
- → con questa tecnica, arriviamo a **200 metri**.

L'encoding

- E' più sofisticato: per problemi di ***sincronizzazione***, si evitano le sequenze di 0 e 1 troppo lunghe, e si cerca di bilanciare il numero di 0 e 1.
- Come?

Immersione!

- Si immerge l'alfabeto in uno più grande con le proprietà volute, perdendo un po' in efficienza ma guadagnando in affidabilità.

Encoding: 8B / 10B

- Ogni **8 bits** sono codificati con **10 bits**, in modo tale che:
- Non ci siano mai 4 bits identici in una word
- Non ci siano mai più di 6 zeri o 6 uni in una word.

Problemi del Gigabit

- Altro piccolo problema: il flusso di dati puòà essere notevole:
- Se il ricevitore (o lo switch) è impegnato per **1 millisecondo**...
- ...si possono accumulare fino a quasi **2000 frames**!!!

Quindi

- Ecco (anche) spiegato perchè gli switch Gigabit costano relativamente molto di più di quelli normali Ethernet,...
- ...ed ecco anche perchè nel Gigabit Ethernet è stato introdotto un nuovo frame speciale per mandare in **pausa** la trasmissione.

Agli standard Wireless

- Il più conosciuto è **802.11**

- Usa la banda **2,4GHz** →
- ***Interferenze*** da forni a microonde, telefoni senza filo, bluetooth...!!

I vari 802.11

- **802.11a**: fino a 54Mbps
- Però, rate effettivo ovviamente minore (codifica 216 bits con 288 bits!)
- **802.11b**: fino a 11Mbps
- Ma “b” viene prima di “a”...
- Inoltre, il range è **7 volte più grande** (vediamo dopo perchè...)
- Infine, **802.11g**: va fino a 54Mbps
- Gli altri 802.11 ed altre cosette le vediamo fra poco

Trasmissione

- Avviene in 5 modi:
- [IMG 4.25 pag 292]
- L'infrarosso (non ha avuto molto successo...)
- FHSS, che vedremo in seguito
- **OFDM** (Orthogonal Frequency Division Multiplexing) è quella usata da 802.11a
- Usa **QAM**
- HR-DSSS (High Rate Direct Sequence Spread Spectrum)
- Essenzialmente, usa modulazione di fase
- E' quella usata da 802.11b
- Ricordate? 802.11b permette trasmissioni ***7 volte*** più estese di 802.11a...
- → Usa codici di ***Walsh-Hadamard*** (missioni spaziali!), ecco perchè.

Come funzionano gli 802.11 in questo layer?

- Ci sono i soliti problemi di hidden station e di exposed station.
- [IMG 4.26 pag 296]

802.11

- Ci sono due modi operativi:
- **DCF** (Distributed Coordination Function), senza controllo centrale.
- **PCF** (Point Coordination Function) con controllo centrale.

DCF

- Usa un 802.11 detto **CSMA / CA** (**CSMA** con **Collision Avoidance**), che può funzionare in due modalità diverse.
- La prima, è come un classico CSMA / CD, di tipo...?
- Ricordate... [IMG 4.4 pag 257]

Scelta che si è fatta

- Nonpersistent CSMA
- (usando il truncated binary exponential backoff)

DCF (cont.)

- Nella sua seconda modalità, CSMA/CA usa essenzialmente **MACAW**.

Problema

- Le ***interferenze*** sono un grosso problema per le reti wireless
- La tecnica usata è sfruttare la lunghezza variabile del frame, trasmettendo pacchetti più piccoli.
- Si usa un protocollo ***stop-and-wait***.

Quando si acquisisce il canale

- Si manda una sequenza di questi frammenti, detta ***fragment burst***
- [IMG 4.28 pag 298]

L'altro modo operativo: PCF

- Qui, la stazione base manda un frame “beacon” periodicamente in broadcast, per sincronizzazione e informazione sullo stato della trasmissione.
- Tutto è dunque centralizzato ed il modo di operazione diventa simile a quello usato dai telefonini.

Oltre 802.11g...?

- 802.11n è arrivato (interessante: aggiunge il **MIMO (multiple-input multiple-output)**), usando le cosiddette ***smart-antennas***.
- → range praticamente raddoppiato rispetto a 802.11g (70 metri indoor, 250 metri outdoor!)

802.11n

- Bandwidth? Dai **54Mbps** di 802.11g a **248Mbps** a **600Mbps**
- ...usa ***banda più grande*** (da canali di 20Mhz si passa a 40Mhz, restando nella banda 2,4Ghz o passando a 5Ghz)
- Usa il **MIMO** appunto (fattore **x4**!)
- Codifica: **QAM** (tipicamente, QAM-64)
- Compatibile con 802.11g (e anche con 802.11b e 802.11a): si sfrutta il...
- ***CTS protection mode*** (RTS/CTS), ed il
- ***Fragmentation treshold***
- Attenzione che molte implementazioni ***non sono standard***....
- → molti computer/laptop/schede dicono di avere 802.11n, ma in realtà implementano un DRAFT (tipicamente, Draft **v2**)
- La versione finale è l'equivalente del Draft **v11.0**

Passiamo ora...

- ...ad un altro importante argomento correlato...

Incollare più reti...!

- [IMG 4.40 pag 320]
- Repeaters
- Hubs
- Bridge
- Switch
- [IMG 4.47 pag 327]

Più reti....

- [IMG 4.46a pag 326]

Repeaters e Hubs

- Il ripetitore, come già visto, ripete (tipicamente, ***amplificandolo*** in potenza) il segnale.
- Lo Hub (che può essere anche repeater) è l'equivalente di una connessione fisica, cioè propaga il segnale da una porta a tutte le altre.

Pro e Contro dell'Hub

- Pro: poco complesso, costa poco, affidabile
- Contro: non risolve i problemi dei ***limiti della rete***.
- Contro: essendo una estensione della stessa rete, **non può unire reti diverse** (ad esempio una 10Mb con una 100Mb).

Il Bridge

- E' un dispositivo di livello più alto (data link), e quindi interagisce con la struttura dei frame, cosa che repeaters e hubs non fanno.

Gli switches

- Sono simili ai bridges, solo che di solito connettono computers o piccole LAN, invece che reti più corpose.

- Quindi, tipicamente, possono preoccuparsi meno dei cambiamenti nella topologia di rete.

Da notare....

- Che la distinzione tra bridge e switch al giorno d'oggi è abbastanza sfumata....
- Ad esempio, alcuni imponevano che il bridge avesse UNA porta in entrata e UNA in uscita, altri no....

Bridge/switch

- L'idea è di unire due o più reti, tenendole per quanto possibile distinte:
- per evitare i problemi di grandezza delle singole reti
- per poter unire reti.....
- Crea una rete più grande ma a livello logico (data link) e non fisico
- In altre parole, non abbiamo più i limiti di grandezza della rete, ed abbiamo domini di collisione diversi.
- Questo si ottiene ispezionando i pacchetti, e filtrando il traffico.

Learning

- C'è quindi una fase in cui il bridge/switch impara la configurazione di rete, e gestisce il traffico corrispondentemente.
- Essendo a livello data link, controlla mittente e destinazione dei frame (→ tipicamente, i ***MAC address***)

Hash tables

- Si usa il backward learning: le hash tables vengono costruite analizzando il flusso di dati e risposte, e si fa ***broadcasting*** nella fase intermedia
- [IMG 4.42 pag 322]

Timeouts...

- C'è un timeout di ***fading*** per ovviare ai cambiamenti nella topologia

Situazione simile...

- Ai motori di ricerca...pensate a come funziona ad esempio Google!

Problema...

- [IMG 4.43 pag 324]
- I LOOPS!!

Soluzione?

- Evitiamo i loop e facciamo solo strutture ad albero
- Ottima idea ***in teoria***, ma in pratica sarebbe troppo limitativo.
- Soluzione?
- Lo fanno i bridge/switch per noi

Quindi....

- Da ogni rete generica con magari dei loops (un grafo)....
- ...estraiamo una sottostruttura senza loops (un albero)
- → lo ***spanning tree***

Spanning trees

- [IMG 4.44 pag 325]

Come si fa?

- C'è un protocollo specifico, 802.1D, che viene gestito tra i bridge/switch, e crea lo spanning tree.

Fading...

- Anche lo spanning tree avviene in maniera distribuita tramite ***fading***, per adattarsi ai cambiamenti della topologia di rete.

Notate anche....

- Come ci voglia tempo per costruire lo spanning tree (e tenerlo aggiornato)

- Tipicamente, 30 secondi per reti normali.
- Nella nuova versione di 802.1D, si è riusciti a scendere a circa 6 secondi.
- E ci sono soluzioni ancora più avanzate (TRILL).

Nota

- Trovate spesso in giro descrizioni che dicono: nei bridge/switch c'è comunicazione istantanea e **in contemporanea** su tutte le porte.
- Commento: UN PAR DE CIUFOLI

Capiamo meglio...

- L'essenziale in un bridge/switch è fare andare le cose molto velocemente.
- Quindi all'interno si deve andare velocissimi!
- Velocissimi non è istantaneo, e soprattutto, non è sempre in contemporanea

Le magie degli switch/bridge

- E' che permettono di collegare anche reti di velocità diversa
- Quindi è possibile in effetti ottenere la magia della comunicazione quasi istantanea e parallela, in alcuni particolari...

Ergo...

- …in alcuni casi, l'uso sapiente del bridge/switch non solo permette reti più grandi, ma **velocizza la rete**, effettivamente permettendo la comunicazione quasi contemporanea.

Però....

- La magia completo non esiste: la comunicazione completamente simultanea in altri casi non può avvenire (dipende ovviamente dai flussi in entrata e in uscita)
- Quindi, si evitano i conflitti con **CSMA/CD**.

Lezione 2 (08/03/2011)

Proseguiamo più in alto...

- [IMG 4.46a pag 326]

Più in alto...

- Nella gerarchia dei “collanti” che uniscono le reti, troviamo poi i routers
- I routers fanno tipicamente parte dello strato network (network layer), che è quello che si preoccupa di portare a destinazione i pacchetti su una rete complessa.

Routing

- Il routing si divide essenzialmente in due tipi: ***adaptive*** e ***nonadaptive***, a seconda se si adattano i meno ai cambiamenti nella topologia di rete.

La metrica del routing

- Il routing deve decidere il modo migliore di gestire i pacchetti nella rete
- Deve quindi avere in mente una “metrica” di qualità per la rete
- Ovviamente varie scelte sono possibili...
- Una buona scelta è per esempio cercare di ***minimizzare la distanza*** percorsa dai singoli pacchetti.

Distanza

- In generale, la distanza si calcola in modo completamente ***modulare***, cioè si considera che la distanza complessiva di un percorso si possa ottenere dalla ***somma delle distanze dei suoi sottocpercorsi*** (per nulla ovvio...!)

Approssimazioni della distanza

- Una prima approssimazione, molto usata, è contare il numero di “**hops**” (balzi), cioè di stazioni incontrate nel cammino.
- Distanze più sofisticate sono ovviamente possibili, ***pesando*** opportunamente ogni singolo cammino tra stazioni (ad esempio, col tempo di trasmissione).

Il caso statico

- Nel caso statico in cui la rete non cambia, un modo ovvio di operare il routing è precalcolare le distanze minime tra tutti i nodi della rete (ad esempio usando l'***algoritmo di Dijkstra***).
- Ogni nodo di routing quindi avrà una tabella che gli indica dove mandare i pacchetti a seconda della destinazione.

Il caso dinamico

- L'ambito più generale è invece il caso dinamico, dove la rete può subire variazioni (anche solo, ad esempio, perchè ci sono guasti o problemi in parti della rete).

Il flooding

- Il flooding ("alluvione") è un mezzo potentissimo per fare il routing.
- L'idea: ogni pacchetto viene ***ritrasmesso a tutte le linee*** (tranne quella da cui è arrivato).
- Mezzo potentissimo; nel bene e nel male

Il flooding: il male

- Cominciamo dai lati negativi: ovviamente la rete col flooding semplice verrebbe sommersa
- Occorrono quindi dei metodi per il "controllo della acque" per così dire...

Controllo del flooding

- Una tecnica è l'***hop counting***: si associa un numero massimo di hop (→ un'età massima) ad ogni pacchetto, dopo i quali il pacchetto muore.

Il flooding: lati negativi

- Anche con queste tecniche, il flooding ha l'ovvio svantaggio che genera una quantità enorme di pacchetti.
- → il carico sulla rete aumenta enormemente

Vantaggi del flooding

- Funziona senza alcuna modifica sia per messaggi point-to-point, che per messaggi a gruppi (***multicast***) o globali (***broadcast***).
- Inoltre, il flooding ***sceglie sempre la via migliore***!
- Ed infine, uno dei più grandi vantaggi del flooding: è robustissimo rispetto alle modifiche della rete.
- In realtà, è facile dimostrare che è il più robusto sistema di routing possibile, cioè , nessun altro sistema è migliore del flooding in questo senso.

Quindi...

- ...utilissimo in tutti quei casi quando il carico di rete non è molto alto, ma o c'è topologia di rete estremamente variabile, o è critico che un messaggio arrivi sempre nel minor tempo possibile...
- Ad esempio, ambito militare...!

Caso dinamico (cont.)

- Ovviamente, ci sono altre soluzioni al caso dinamico che non il flooding
- Si basano tutte sull'idea che si raccolga informazione globale assommando varie informazioni locali, come i mattoncini Lego

Distance vector routing

- Era il routing usato dalla prima versione di Internet (**ARPANET**).
- L'idea è che ogni router ha una tabella di routing che contiene informazioni su quanto veloce è la connessione ad un altro router, e qual è la via migliore (tra i primi vicini) per raggiungerlo.

Distance value routing

- Funziona in modo molto semplice:
- Ogni router chiede ai suoi vicini la loro tabella
- Usa poi le loro tabelle, ed il tempo che c'è voluto per averle, per costruire la sua tabella selezionando i percorsi migliori.

Distance Vector Routing

- [IMG 5.9 pag 358]

Pro e contro Distance Vector Routing

- Pro: è veloce a recepire le “buone notizie”
- [IMG 5.10a pag 359]

Contro...

- Non tiene conto della capacità di banda
- Ricordate il periodo storico: circa 1970 → tutte le linee avevano 56kbps
- Successivamente, le linee hanno cominciato a differenziarsi per banda
- Certamente, si può modificare l'algoritmo per tenere conto anche della banda usando dei coefficienti a pesi.

L'altro problema

- C'è però un altro problema....
- Quanto questo routing si comportano bene rispetto alle “***buone notizie***”, tanto male, dualmente, si comporta con le “***cattive notizie***”.

Il problema del count-to-infinity

- [IMG 5.10b pag 359]

Routing fase 2: il Link State Routing

- Per questi problemi, il routing su Internet, dopo il 1979, è stato sostituito da un altro algoritmo, il cosiddetto link state routing.

Link State Routing

- All'inizio, come nell'altro tipo di routing, si costruisce informazione locale
- Ogni nodo quindi trova quali sono i suoi vicini (tramite pacchetti HELLO)
- Poi misura quanto “lontani” sono tramite dei pacchetti ECHO (similmente all'altro algoritmo di routing).
- Quando ha informazione completa sui suoi vicini, ogni nodo costruisce un pacchetto che contiene tutta questa informazione, più altra informazione che vedremo fra poco...
- ...e la manda a tutti gli altri (***broadcast***)
- L'idea è quindi che ogni nodo riceverà i mattoncini lego corrispondenti alle informazioni locali di ***ogni altro nodo***.
- In tal modo, potrà ricostruire una mappa completa della rete, e quindi calcolare i percorsi migliori.
- Restano da sistemare varie cosette
- Come si fa il broadcasting?
- Si usa essenzialmente il ***flooding***
- C'è però una differenza: questo flooding andrà ripetuto ogni tanto per fare il “refresh” della rete.
- Ma il tempo in una rete generica NON RISPETTA NESSUN PRINCIPIO DI ORDINE.
- Cioè, un pacchetto inviato prima potrebbe anche arrivare dopo (o non arrivare mai).
- Quindi, potrebbe succedere che un pacchetto con l'informazione locale, arrivi dopo un altro pacchetto con informazione locale più aggiornata
- → c'è bisogno di far funzionare una specie di orologio interno.
- Per questo motivo, ad ogni pacchetto di informazione locale si attacca un ***numero di sequenza*** (in un certo senso, l'orologio locale del nodo).
- In tal modo, chi riceve l'informazione saprà sempre qual è quella più aggiornata da tenere in considerazione.
- C'è un altro problema su una rete generica: i disastri
- I disastri possono accadere sia perchè dei pacchetti vengono corrotti, sia perchè qualche nodo per vari motivi smette di funzionare.

- Cosa succede se ad esempio un numero di sequenza viene corrotto?
- Siamo alla sequenza 2100...
- ...c'è corruzione, e la sequenza viene corrotta in 82100...
- → tutti i successivi aggiornamenti 2101....82100 verranno ignorati.
- Similmente, cosa succede se ad esempio un nodo va in crash?
- Era al numero di sequenza 920700...
- ….va in crash....
- Perde l'informazione sulla sequenza, quindi deve ricominciare da 0...
- → tutti gli aggiornamenti 0-920700 verranno ignorati!
- Come si risolve questo problema?
- Prendendo due piccioni con una fava
- Abbiamo detto che si usa il flooding per la propagazione, ma non che tipo di flooding si usa.
- Usando il flooding versione ***con età***....
- → possiamo usare il fading, cioè usare l'età non solo per controllare il flooding, ma anche internamente nella tabella
- Quindi dopo un po', l'informazione se non viene aggiornata viene fatta morire
- → ***risolviamo tutti i problemi visti precedentemente!***

Quindi riepilogando: pacchetti link-state

- [IMG 5.13 pag 363]

Link State Routing

- Notare la differenza con il Distance Vector Routing:
- Stiamo sprecando più banda (per via del broad casting), però, proprio come nel caso del flooding, quest'uso aggiuntivo della banda (a livello ***globale***) ci permette di essere molto più robusti (evitiamo il problema della ***località*** presente nell'altro algoritmo).
- Restano però altri problemi, come ad esempio il fatto che se la rete cresce, occorrono tabelle di routing sempre più grandi, così come i tempi di ***refresh del flooding*** aumentano.

Il Broadcast routing

- E' interessante notare che quando occorre fare **broadcast** su una rete che ha già in funzione il link state routing, non serve usare il ***flooding***, ma si può essere più intelligenti.
- Si usa invece un'altra tecnica più intelligente, il cosiddetto ***reverse path forwarding***.

Reverse path forwarding

- L'idea è molto semplice: funziona come il flooding, solo che vengono considerati solo i pacchetti provenienti dal cammino migliore, mentre tutti gli altri vengono ignorati.
- Essenzialmente, si forza la struttura ad albero data dall'algoritmo di routing in uso.

Reverse path forwarding

- [IMG 5.16 pag 369]

Altre soluzioni?

- Dividere la rete in zone gerarchiche (il cosiddetto hierarchical routing).
- Quindi, la rete viene suddivisa in sottoreti e così via
- Ogni router ha conoscenza completa della propria rete, e le reti esterne al suo livello vengono collassate in un solo nodo.

Routing gerarchico

- [IMG 5.15 pag 367]

Altri problemi

- La taglia non è il solo problema del routing...
- L'altro problema è il cosiddetto effetto a risonanza (conosciuto con vari nomi, ad esempio ***effetto splash***)

Anche noto...

- Come effetto see-saw (effetto altalena)

Routing ed effetto see-saw

- Nel calcolo della distanza con un vicino, abbiamo visto che si considera il tempo di trasmissione.
- Ma si considera anche il carico?
- Senza considerare il carico, rischiamo l'effetto di ***congestione***: il percorso migliore dipende solo dalla topologia, e quindi tutti sceglieranno quello.

Nota: problema non solo nelle reti di computer....

- "Partenze intelligenti"... (cambiare orario perchè so che ad una certa ora c'è congestione)

Quindi...

- Una buona strategia è tenere conto anche del carico
- (quindi ad esempio, misurare il tempo non appena si mette il pacchetto nella coda, non quando effettivamente lascia il router)

L'effetto splash/see-saw

- [IMG 5.12 pag 362]

Quindi...

- L'effetto splash / see-saw etc fa avere un comportamento ***oscillatorio*** molto brutto, che ovviamente porta problemi non da poco.

Abbiamo quindi...

- Visto come i problemi di gesione del traffico di rete sono molteplici, incluso il non facile problema del routing.

Quality of Service

- La qualità del servizio (***QoS***) è una serie di parametri che dettano appunto la qualità del servizio offerto.
- Nell'ambito delle reti, tipicamente la QoS è data da ***quattro parametri principali***.
  - **Reliability** (affidabilità)
  - **Bandwidth** (banda)
  - **Delay** (ritardo di trasmissione)
  - **Jitter** (ora vediamo)

Lezione 3 (09/03/2011)

Jitter

- Il Jitter è l'equivalente "al secondo ordine" del delay:
- Misura il ***grado di variazione*** (deviazione standard) nei ***tempi di arrivo*** dei pacchetti
- [IMG 5.30 pag 397]

Soluzioni per il QoS

- Riguardo alla reliability, abbiamo già visto soluzioni come error control ed error correction
- Le altre misure (**bandwidth, delay, jitter**), derivano da alcune ***cause***, che devono essere risolte se se vuole mantenere una buona QoS.

Causa: la congestione

- Una delle cause principali di problemi nel QoS è la **congestione**, cioè quando la capacità della linea o di qualche stazione si satura
- E' importante dunque fare attenzione al ***congestion control*** per evitare i problemi di QoS.
- [IMG 5.25 pag 385]

Attenzione

- Notate la differenza tra flow control e congestion control:
- ***Flow control*** si riferisce al traffico point-to-point, e si preoccupa che il ricevente sia in grado di assorbire il flusso
- ***Congestion control*** invece deve garantire che la rete, globalmente, sia in grado di gestire il

traffico richiesto.

Connessioni

- Ricordate i due tipi principali di connessioni che abbiamo visto: ***connectionless*** e ***connection-oriented***.
- Le prime vengono anche dette ***datagram networks*** (ed i loro pacchetti ***datagrammi***), le seconde invece ***virtual-circuits (VC)***.

Congestion nei varitual circuit

- Qui le cose sono più facili:
- Si può fare ***admission control***, cioè se c'è congestione si proibisce la creazione di nuovi circuiti virtuali.
- Inoltre, si possono evitare i nodi congestionati, quando si crea un circuito virtuale.

Congestion nei VCs

- [IMG 5.27 pag 390]
- In generale dunque, nei VCs le cose sono più facili perchè la situazione è abbastanza statica, e si può scegliere la via migliore in anticipo.
- Il prezzo da pagare è che spesse volte si ***sprecherà banda***, essendo ogni connessione indipendente dalle altre.

Congestione nelle reti datagram

- Il caso più delicato è invece quello dinamico, cioè nelle reti a datagrammi.
- Varie strategie sono possibili, vediamone qualcuna!

La tecnica dei choke packet

- Come nel flow control, alle volte il ricevente può segnalare che le cose non vanno bene: anche qui l'idea è che se un router si accorge che c'è congestione, può inviare un pacchetto speciale (***choke packet***) a chi sta inviando dati, dicendogli di rallentare.

I choke packets

- Tipicamente dunque, un host ad esempio dimezza il suo data rate non appena riceve un choke packet.

Gestione del choke

- Ad esempio, si usa il ***fading*** come modo per uscire dal choke:
- Se non riceviamo choke packets dopo un certo periodo di tempo, torniamo ad incrementare la nostra velocità di trasmissione.
- [IMG 5.10 pag 359]
- C'è congestione sulla linea tra A ed E
- → B, C e D inviano dei choke ad A
- A riceve il primo choke da B: -50% (0,5)
- Riceve il secondo da C: -50% (0,25)
- Riceve il terzo da D: -50% (0,125)
- Quando invece bastava ridurre il flusso solo del 50%

Soluzione

- Si fa fading alla rovescia, nel senso che non appena si riceve un choke, per un certo periodo di tempo si ***ignorano*** le altre richieste di choke che vengono dalla stessa destinazione.

Gestione del choke

- Se la rete è grande (pensate ad Internet), una richiesta di choke può metterci troppo tempo per sistemare le cose.
- Una soluzione è una variante del choke, il choke **hop-by-hop**.

Hop-by-hop choke

- Il choke hop-by-hop non agisce solo su chi sta inviando dati, ma agisce anche mentre passa, rallentando i routers.
- [IMG 5.28b pag 393]

Altra tecnica....Load Shedding

- Un'altra tecnica è quella derivata dalle reti elettriche, estrema ma molto utile:
- Il load shedding fa sì che in caso di sovraccarico alcuni pacchetti semplicemente ***vengano buttati via***.
- Ovviamente, si può fare in modo intelligente, non buttando via pacchetti a caso, ma con raziocinio.

Wine

- "Old is better than new"

Wine e Load Shedding

- Wine: ad esempio, se ho un remote login, conviene buttar via i pacchetti più nuovi rispetto ai vecchi.

Qual è l'altra tecnica? Milk

- "New is better than old"

Milk e Load Shedding

- Invece, ad esempio in un file multimediale conviene buttar via un pacchetto vecchio: anche se c'è qualche salto nella trasmissione, l'importante è che "***the show must go on***"

Buffering

- Un'altra tecnica molto usata è il buffering nella stazione ricevente
- Rispetto al QoS, il buffering non risolve la congestione e quindi il **delay**, ma riduce il **jitter**
- Ovviamente può essere gestito dinamicamente (vedi multimedia sul web)

Traffic Shaping

- Un altro modo di limitare la congestione è quello sempre di non lasciare l'onere ai router, ma a chi trasmette e riceve dati:
- Mettendosi d'accordo su che rate di trasmissione avere (***service level agreement, SLA***)
- Anche qui ovviamente il livello può essere cambiato dinamicamente.

Altro modo: i Buckets

- I "buckets" (secchi) sono essenzialmente una specie di filtri, che servono a garantire buone proprietà (in termini di QoS)
- Tipicamente, sono gestiti da chi invia i dati, anche se (al solito) ci può essere un adattamento dinamico.

Leaky Buckets

- Letteralmente, il "secchio che perde", è un tipo di filtro che garantisce un data rate ridotto costante (quindi, evita i ***burst*** che possono sovraccaricare la rete).
- [IMG 5.32 pag 401]

Oltre il leaky bucket

- Il leaky bucket ha il vantaggio/svantaggio che il data rate è sempre costante.
- Alle volte, ad esempio in presenza di traffico più sostenuto, converrebbe magari aumentare un po' il data rate.

Il Token Bucket

- Il token bucket genera ogni certo intervallo di tempo un ***token***
- I pacchetti in arrivo possono uscire solo se "bruciano" un token disponibile
- Quindi, se il traffico per un certo periodo è lento, ma poi c'è un burst, si riesce a gestirlo meglio consumando i **token** che si erano accumulati.

Token Bucket

- [IMG 5.34 pag 404]

Combinazioni

- I buckets possono essere ***combinati***
- Ad esempio, possiamo avere un leaky bucket dopo un token bucket, ottenendo una via di mezzo fra un token e un leaky bucket.

Ed ora...lo strato di rete di Internet

- Dopo aver visto varie tecniche per il routing ed il QoS, è arrivato il momentoi di vedere più nel dettagio quale protocollo usa Internet per trasmettere dati a livello di rete
- **IP** = **I**nternet **P**rotocol
- Che avrete sentito senz'altro di solito dentro a “TCP/IP”

“TCP/IP” o “TCP” / “IP”?

- 1969: ARPANET
- Usava il Network Control Protocol (NCP)
- → ***inadatto***
- 1974: Transmission Control Protocol (TCP)
- 1978: Divisione TCP e IP

OSI e TCP/IP

- [IMG 1.22 pag 43]

Quindi...

- IP ~ “network level”
- TCP ~ “transport level”
- E poi gli “strati alti”, che includono ad esempio FTP, Telnet, SMTP, SNMP, NFS, XDR, RPC, X Windows, SFTP, ecc....

IP e TCP

- IP: il postino “pigro”, disattento
- TCP: il postino “pignolo”

Il protocollo IP

- Sposta pacchetti chiamati “datagrammi”
- Servizio di posta base “inaffidabile” (unreliable):
    - Pochissimi controlli
    - Si possono perdere pacchetti
    - I pacchetti possono essere consegnati mischiati
    - Oppure in copie multiple
    - Oppure con grave ritardo

L'header di IP

- Ha una parte fissa a 20 bytes, ed una a lunghezza variabile.
- [IMG 5.53 pag 434]
- La prima parte è la versione di IP usata (permette quindi il ***versioning***)
- La seconda è la lunghezza dell'header (**I**P **H**eader **L**ength)
- La terza è il tipo di servizio (campo adatto per la selezione della QoS), anche se spesso ignorato.
- C'è poi la lunghezza totale del datagramma IP
- L'***identification*** serve a identificare se c'è stata frammentazione dei dati in più datagrammi.
- Il Campo DF invece (Don't Fragment) impone la non-frammentazione dei dati
- Il campo **MF** (More Fragment) segnala l'ultimo frammento.
- Infine il fragment offset è il numero d'ordine del frammento.
- Il Time to Live (TTL) è l'età massima del pacchetto
- Il campo **protocol** indica il protcollo di più alto livello che sta usando IP
- C'è poi un checksum ***per l'header***, calcolato banalmente tramite somme in complemento a uno.
- Poi, gli indirizzi IP di chi invia e di chi deve ricevere il datagramma.
- E infine, la parte variabile per opzioni extra, di cui riparliamo dopo

Analisi critica

- IP... Lo strato fondamentale di Internet...
- Tutto qui?

DoD (Dipartimento della Difesa)

- TCP/IP è stato inizialmente commissionato dal DoD (il Diparimento della Difesa statunitense).

Indietro alla storia...

- **1969**: ARPANET
- Usava il Network Control Protocol (NCP)
- ***Realiable, flow-controlled***
- **1974**: Transmission Control Protocol (TCP) → ***ancora troppo per il DoD***
- **1978**: Divisione TCP e IP
- (**1983**: DoD impone il TCP/IP)

DoD e l'uso militare

- ARPANET → Internet e... MILNET!

Design: portabilità

- Nell'ambiente militare, “media” tra i più disparati: sistemi diversi, e committenti diversi per ragioni di legge (competizione e libero appalto)
- → necessità di avere in IP una struttura di comunicazione con elevata astrazione (layer), cioè che possa essere **portabile** sui media più disparati.

Design: no overload

- La rete non deve essere sottoposta al pericolo di sovraccarichi.
- → niente traccia in IP degli errori intermedi durante una comunicazione
- → similmente per TCP (l'errore è “end to end”), senza sovraccarichi
- Ed inoltre, dunque, niente connessione persistente (→ IP “connection-less”)
- E dunque, necessità di ***ritrasmissione*** nel caso di failure.

Design: robustezza

- In battaglia, la perdita di un nodo (attack) è prassi comune: le perdite possono essere rendicontate successivamente, l'importante è che la rete resti operativa (“Errors are normal and can be largely ignored”)
- → la comunicazione IP si riconfigura in caso di attacco alla rete
- → con sufficiente ***ridondanza*** si può controbattere un attacco.

Modelli Open e Closed World

- Nonostante gli sforzi per rendere TCP/IP “robusto ad attacchi”, il sistema è stato pensato per il modello closed-world (tipo MILNET) → attacchi esterni
- Quando l'attacco è interno (open-world model), ci sono molte vulnerabilità (ad esempio, IP gestisce male la ***congestione del traffico***)
- Riflessione: questo spiega molti dei problemi di Internet (e i non-problemi di MILNET, da quanto si sa)

I limiti di estendibilità

- Uno dei più grossi problemi dell'informatica è l'estendibilità di un servizio.

Esempi

- “640k ought to be enough for everybody”

Altro esempio:

- 32Mb, 4Gb, 32Gb

I limiti di estendibilità di IP

- Vediamo meglio i limiti intrinseci di IP a livello di estendibilità

- IHL è di 4 bit, quindi la lunghezza massima dell'header è di 60 bytes.
- → meno i 20 bytes dei campi fissi visti prima, fanno 40 bytes per le parti di estensione opzionale
- → FORTE LIMITE ALL'ESTENDIBILITÀ

Limiti di IP (cont.)

- TTL (Time to Live): 255 hops massimi
- Può essere un problema se la rete è mal disegnata e non gerarchica
- Comunque, il router può intervenire e riaumentare il TTL (o, non diminuirlo), quindi non è un grosso problema.
- Gli indirizzi (source e destination): **32 bits**
- Questo si è rivelato essere un ***grandioso problema***, perchè lo sapzio degli indirizzi globale è fisso e non estendibile.

Indirizzi IP

- Essendo di 4 bytes, vengono al solito scritti con la cosiddetta **dot notation**, ogni byte in decimale separati dal punto (es. 192.168.2.1, 127.0.0.1 etc)
- Ovviamente, ci deve essere una ***autorità centralizzata*** che assegna gli indirizzi IP alle entità in Internet.
- Potremmo assegnare gli indirizzi ***uno a uno***, ma in questo modo, i router dovrebbero mantenere delle tabella mostruosamente grandi.
- La soluzione è quella di avere anche negli indirizzi IP una struttura gerarchica; invece di assegnare un solo indirizzo alla volta, assegniamo interi ***blocchi di indirizzi*** alle varie organizzazioni (→ tante reti).
- Ogni rete poi provvederà alla gestione del routing tra i vari nodi.

Indirizzi IP inizialmente

- Quindi, all'inizio di Internet si è pensato di avere vari tipi di ***blocchi (classi)***, a seconda della grandezza della rete richiesta da un'organizzazione
- → metodo del ***classful addressing***
- [IMG 5.55 pag 437]

Le classi principali di indirizzi IP

- **Classe A**: (7+24)
  128 reti possibili, con indirizzi di 24 bits (circa 16,7 milioni)
- **Classe B**: (14+16)
  16384 reti possibili, con indirizzi di 16 bits (65536)
- **Classe C**: (21+8)
  circa 2 milioni di reti, con indirizzi di 8 bits (256)

Notare...

- Le taglie possibili delle reti:
- **1 byte – 2 bytes – 3 bytes**
- **256 – 65536 – 16,7 milioni**

256 – 65536 – 16,7 milioni

- Manca una taglia intermedia (ad esempio, un migliaio)
- → la maggioranza ha chiesto una classe B (65536), sprecando spazio!!!
- Confrontate con Ethernet e Metcalfe...
  (3 bytes + 3 bytes)

Lezione 4 (10/03/2011)

La soluzione?

- **CIDR: Classless InterDomain Routing**
- Classes: si va oltre le vecchie classi di Internet

- Ora i blocchi sono di lunghezza variabile

Lunghezza variabile?

- Occorre informazione aggiuntiva su quanto grande è il blocco di indirizzi
- Questa informazione è contenuta in una ***maschera*** (**mask**) di bits
- Questa maschera di 32 bits ha tanti 0 quanto è largo il blocco di indirizzi, mentre gli altri bits sono 1

Indirizzi CIDR

- Gli indirizzi CIDR di solito si scrivono in maniera abbreviata usando la ***CIDR notation***, che è la dotted notation seguita da uno slash (“/”) e poi dal numero di 1 nella maschera.

Esempio

- La vostra organizzazione chiede un blocco di ***2048 indirizzi (11 bits)***
- Gli viene assegnato ad esempio come primo indirizzo 194.24.0.0, con maschera composta da 11 bits a 0, e 32-11 = 21 bits a 1
- La maschera di solito si scrive mettendo gli 1 come bits più significativi:

11111111111111111111100000000000
(in tal modo con un AND binario separate l'indirizzo della rete dall'indirizzo nella rete)

- In CIDR notation, la vostra organizzazione ha dunque come indirizzi **194.24.0.0/21 (a quanti 1 corrisponde la maschera di bit)**

Aggregazione

- Il CIDR permette lo stesso ai router una gestione abbastanza efficiente, per via della cosiddetta aggregazione:
- Ricordiamo che dopotutto i router devono solo indirizzare i pacchetti al router successivo.
- Quindi, nelle loro tabelle possono usare le cosiddette **aggregate entries**:

Aggregate entries

- Se ci sono varie classi di indirizzi che devono venire indirizzati allo stesso router, possono essere combinate in un'unica entrata se hanno un prefisso comune.

Esempio

- Un router vede che il router successivo a cui inviare pacchetti è lo stesso per:
- La vostra organizzazione 194.24.0.0/**21**:
  194.24.[**00000**000].0
- Ed un'altra 194.24.16.0/**20**:
  194.24.[**0001**0000].0
  Ho due reti diverse però hanno la proprietà di avere un prefisso simile.
- Può aggregare le due entrate in una singola entrata 194.24.0.0/**19**:
  194.24.[**000**00000].0

Però

- Cosa succede se c'è un'altra organizzazione dentro a 194.24.0.0/19
  194.24.[00000000].0 che invece deve essere mandata ad un altro router?
- Ad esempio 194.24.8.0/22
  194.24.[00001000].0
- Si usa la regola del ***longest matching***: cioè l'entrata con il prefisso di rete più lungo ha la priorità.

Subnet masks

- La stessa tenica del CIDR usata per creare sottoreti nella rete Internet viene anche usata dentro alle singole reti, sempre usando maschere di bits per creare sottoreti (subnets) dentro alle organizzazioni
- [IMG 5.58 pag 440]

Forma di Internet

- Che forma ha lo spazio degli indirizzi di Internet?

- [IMG mappa di internet]

Come si va oltre il CIDR?

- Il CIDR ha servito molto bene il suo scopo, ma lo spazio degli indirizzi sta diminuendo: ovviamente, quando non ci saranno più indirizzi sarà un disastro perchè la rete non potrà più crescere.
- Allora, come si va oltre?

In realtà...

- Il vero salvatore dello spazio IP non è stato tanto il CIDR quanto un'altra tecnologia, la tecnologia NAT

NAT

- **NAT** sta per **Network Address Translation**
- L'idea (ingegnosa) è di simulare una interna sottorete usando un solo indirizzo IP.

Come funziona il NAT

- Internamente, la rete funziona con degli indirizzi IP interni, che sono invisibili all'esterno.
- Quindi dentro alla rete, tutto funziona in maniera trasparente.
- All'esterno invece, la rete appare come un unico indirizzo IP.

Il NAT box

- [IMG 5.60 pag 446]

NAT nel dettaglio

- Alcuni indirizzi sono riservati per il NAT e non possono essere usati come normali indirizzi Internet:
- **10.0.0.0/8**
  (rete da 16777216 hosts)
- **172.16.0.0/12**
  (rete da 1048576 hosts)
- **192.168.0.0/16**
  (rete da 65536 hosts)

E l'altro verso?

- Il vero punto del NAT è però un altro: l'altro verso dei messaggi
- Quando un messaggio è inviato a un computer della rete interna, come fa il NAT box a mandarlo al computer giusto?
- Per sapere la risposta occorre aspettare un po'

Uso di NAT

- NAT è molto usato dentro ad organizzazioni, ma anche ad esempio:
- nelle ADSL (in tal modo un provider può moltiplicare gli indirizzi IP a sua disposizione)
- nelle reti locali casalinghe, dove un utente da un singolo indirizzo IP può creare una sottorete di macchine diverse

Il resto del network layer

- Come abbiamo visto, **IP** è il protocollo di base che serve a traserire dati nello strato network.
- Ci sono altri protocolli che servono a far funzionare lo strato network:
  tipicamente, **ICMP**, **ARP** e **DHCP**

ICMP

- **ICMP (Internet Control Message Protocol)** è il protocollo di controllo di Internet per gestire eventi inaspettati.
- I messaggi ICMP vengono incapsulati dentro IP
- [IMG 5.61 pag 449]

ARP

- **ARP** (**A**ddress **R**esolution **P**rotocol) risolve il problema di far parlare gli indirizzi IP con gli altri tipi di indirizzi presenti nel data link layer

- Supponiamo di dover inviare un pacchetto IP: l'indirizzo del destinatario è nello spazio IP
- Il data link layer però può usare un altro spazio di indirizzi.
- Ad esempio, Ethernet, se ricordate, usa uno spazio di indirizzi di 48 bits (MAC)
- Quindi, ad esempio, come avviene dentro ad una LAN la traduzione da indirizzi IP a indirizzi Ethernet?
- Una soluzione potrebbe essere avere da qualche parte un server o simil-router a cui rivolgersi, con la mappa delle corrispondenze IP → Ethernet
- Questo però introdurrebbe un collo di bottiglia, rendendo le LAN inutilmente più lente.
- La soluzione adottata è invece diversa: si gestisce la corrispondenza in modo totalmente distribuito

L'idea

- Tipicamente nelle LAN il costo di una broadcast è simile all'invio point-to-point di un pacchetto.
- Quindi invece di avere una server che gestisce la corrispondenza IP → strato data-link, facciamolo fare indipendentemente a ciascuna macchina.
- Ogni macchina quindi fa correre il protocollo ARP, mantenendo una tabella delle corrispondenze IP → data-link
- Quando vuole inviare un messaggio ad un certo indirizzo IP ***X***, manda un messaggio ARP (che va in broadcast) chiedendo chi ha l'indirizzo ***X***
- Solo la macchina con indirizzo X risponde con un messaggio ARP di ACK, includendo il suo indirizzo data-layer

ARP e le tabelle locali

- Possiamo ottimizzare il processo, visto che come detto, quello che conta sono le tabelle locali ARP di corrispondenza
- Ad esempio, tipicamente quando una macchina A comunica con una macchina B, è molto probabile che anche B poi comunichi con A.
- Quindi se in ogni pacchetto ARP che una macchina invia c'è la corrispondenza IP → data-layer della macchina stessa....
- ...quando A chiede tramite ARP chi ha l'indirizzo IP X, e B risponde...
- ...anche B sa già a quel punto la corrispondenza IP → data-layer di A
- Quindi quando B dovrà magari rispondere ad A tramite IP, non dovrà fare un'altra richiesta ARP.

ARP e tabelle locali

- Un'altra ottimizzazione possibile è per ogni macchina, quando vede un pacchetto ARP, memorizzare in ogni caso le informazioni contenute sulla corrispondenza IP → data-layer
- In tal modo ogni volta c'è un comando ARP tra due macchine, anche le altre hanno informazione sulla loro corrispondenza utile per future comunicazioni.
- Una conseguenza di questa politica è che, tipicamente, quando una macchina entra in una rete (ad esempio, viene accesa) invia un pacchetto ARP chiedendo il suo stesso indirizzo IP.
- Nessuno risponderà, ma tutti hanno la possibilità di memorizzare la nuova corrispondenza IP –> data-layer
- Inoltre, avviene un controllo sulla gestione degli indirizzi IP nella LAN, perchè se invece qualcuno risponde, significa che due macchine nella rete hanno lo stesso indirizzo IP → c'è stato un errore di gestione

ARP e fading

- Anche ARP funzioina tramite fading, cioè le corrispondenze IP → data-layer invecchiano e dopo un po' muoiono, permettendo quindi una gestione più flessibile della LAN

ARP

- Inoltre se c'è nuova informazione su altri assegnamenti IP → indirizzi MAC, questi hanno

priorità sui vecchi
- Questo permette la gestione dinamica della rete

L'opposto di ARP
- ARP serve in una LAN a dare la corrispondenza
  indirizzo IP → indirizzo data-layer
- Alle volte però serve fare l'opposto, cioè avere la corrispondenza
  indirizzo data-layer → indirizzo IP
- Ad esempio, quando l'indirizzo IP di una macchina non è fisso, ma viene gestito dinamicamente dalla rete.

DHCP
- La soluzione è data dal protocollo **DHCP**, che sta per
  Dynamic Host Configuration Protocol
- Quando una macchina vuole sapere il suo indirizzo IP, manda un pacchetto DHCP di tipo DISCOVERY
- Il gestore DHCP della rete (stavolta, ***centralizzato*** e non distribuito come in ARP) risponde con l'indirizzo corrispondente.
- Ci possono essere vari gestori DHCP che offrono un indirizzo IP: il richiedente ne sceglie uno, e invia un messaggio di acquisizione.
- Il gestore IP corrispondente risponde con un ACK, e a quel punto l'indirizzo IP è stato assegnato.

DHCP e fading
- Anche DHCP ha il problema del refresh dei dati (ad esempio, una macchina potrebbe poi uscire dalla rete)....
- …e per risolverlo usa il fading, che qui è chiamato ***leasing***
- La differenza col fading classico è che nel leasing chi ha acquisito l'informazione sa della scadenza, e quindi pià agire in tempo per rinnovarla

La struttura gerarchica di Internet
- Abbiamo visto la gestione di Internet a livello LAN
- Ovviamente poi, come detto, Internet è costruita gerarchicamente a vari livelli.

Il livello AS
- I livelli sono bene o male
  semplicemente reti sempre più grandi, ma c'è un livello particolare, quello AS
- Il livello **AS** è quello degli ***autonomous systems***.
- Ricordiamo che Internet è presente quasi in ogni parte del mondo...
- ...ma DI CHI è?
- Molte parti sono in mano ai governi locali, molte altre invece sono in mano ai privati
- Come interagiscono queste parti? Ognuna di queste reti è pianamente integrata in Internet?
- La risposta è NO.
- Un **Autonomous System** è appunto una rete sotto il controllo di un singolo gestore:
- Ogni gestore ha liberà di decidere la sua politica di integrazione con il resto della rete.

Quindi...
- “gestire la politica” significa essenzialmente definire come funziona il ***routing*** di ingresso e di uscita tra i vari AS

La struttura di Internet
- Vediamola! [IMG]

Routing tra AS
- Il routing tra AS viene effettuato da un protocollo speciale: **BGP**
- ***Border Gateway Protocol***
- Molto complesso, essenzialmente tramite questo protocollo un gestore di AS può decidere

come la sua rete si integra in Internet

Ad esempio...

- Si possono specificare regole come:
- Ogni pacchetto che proviene dal governo USA non deve mai passare per l'Iraq
- Ogni pacchetto da un sito interno a Google non deve mai passare per la rete Microsoft

BGP

- Funziona essenzialmente col ***distance vector***, con la differenza che invece di tenere le **distanze** in tabella, tiene gli ***interi cammini***

Problema della performance

- Supponiamo di avere pacchetti di 40-64 bytes
- a 100Gbit/s quanto tempo ha il router per decidere?
- **3-5ns di tempo**
- Superiore ai tempi di accesso della DRAM
- Si usa SRAM, e l'algoritmo di ***lookup*** dev'essere velocissimo (“Lulea scheme” per comprimere)

Lezione 5 (11/03/2011)
recupero mortale – maratona 4h with special guest MM DJ in da mix,
free drink se entri con lista SailorStur

Oltre IP?

- Abbiamo visto la scarsità di indirizzi dello spazio IP
- Anche se CIDR e NAT hanno dato fiato, la situazione alla fine collasserà
- Occorre una soluzione definitiva
- → una nuova versione di IP: **IPv6**

IPv6: l'header

- [IMG 5.68 pag 467]

Quindi...

- Alcune differenze saltano subito all'occhio:
- Gli indirizzi sono ora di **16 bytes** invece che di **4 bytes**
- Alcuni volevano meno bytes, altri di più, altri un numero variabile
- Siamo sicuri che con un numero fisso di bytes non limiteremo ancora l'estendibilità?

16 bytes

- Cerchiamo di capire.....
- Se abbiamo esaurito lo spazio dei 16 bytes, e distribuissimo i computer sulla superficie terrestre, quanti ne avremmo per chilometro quadrato?
- Circa **700000000000000000000000000000**
- Cioè a dire: **700000000000000000 ogni millimetro quadrato.**

Torniamo all'header IPv6

- C'è stata una grande semplificazione: solo **7** campi invece dei **13** di IPv4
- C'è il campo version (sempre 4 bit), ora contiene 6 invece che 4
- C'è traffic class, circa equivalente al type of service di IPv4, ma stavolta con 8 bits invece di 6 bits.
- Anche se finora non usato in Ipv4, si pensa sarà importante per ***real-time e multimedia***.
- C'è poi un nuovo campo, flow label (etichetta del flusso)
- IPv6 ha in più la gestione dei ***flussi***: l'idea è di poter creare flussi separati anche partendo dalla stessa macchina, flussi che magari hanno esigenze diverse (banda, real-time, etc)
- Simile alla creazione di ***circuiti virtuali***

IPv6

- Il ***Payload length*** è simile al campo Total Length di IPv4, solo che ora dà la lunghezza del payload, escludendo l'header.
- ***Hop Limit*** è come il TTL di IPv4, solo che ora è stato ribattezzato più correttamente con “hop” invece che con “time”
- Notare che ***è rimasto 8 bits***, come in IPv4...
- Infine, c'è il campo “***next header***”, che serve a gestire le ***estensioni*** in IPv6
- Non commettendo più l'errore di IPv4, ora non c'è un limite alle estensioni: le estensioni stanno in un altro header, il cui tipo è appunto presente nel campo ***next header***.

Ultima nota:

- IPv6 ***non ha il campo checksum***
- → è un protocollo totalmente ***unreliable***
- L'idea è che il controllo dell'errore, se serve, verrà fatto dai procolli degli.....

Lo strato di trasporto

- IP è principalmente lo ***strato network*** di Internet
- Vediamo ora come viene usato nello strato di trasporto
- Ha una differenza con lo strato network: siccome è una astrazione ulteriore, fa multiplexing delle singole risorse network (le macchine, identificate da indirizzi IP) tramite le ***porte*** (ports)
- IP + porta = ***socket***, cioè la divisione in “rete logica”, a livello transport, di una singola entità in rete.

UDP

- Cominciamo quindi con l'applicazione più semplice, UDP
- UDP sta per **User Datagram Protocol**
- Essenzialmente da poco di più che non usare IP nello strato di trasporto.

L'header UDP

- [IMG 6.23 pag 526]

Quindi...

- UDP aggiunge l'informazione minimale per lo strato transport: la ***porta*** di chi ***manda***, e la ***porta*** di chi ***riceve***.
- Inoltre fa il controllo dell'errore tramite ***checksum*** (che tra l'altro è opzionale).

Uso di UDP

- UDP è dunque sempre un protocollo ***connection-less***, e si usa in tutte quelle situazione dove serve inviare messaggi brevi o dove non serve avere controllo dei dati ma si preferisce la velocità

Esempio d'uso:

- Il **DNS**, cioè il **D**omain **N**ame **S**ystem
- Quella parte di Internet che associa dei nomi agli indirizzi IP

Il DNS Name Space

- [IMG 7.1 pag 581]

Il DNS

- Funziona ***gerarchicamente***, usando i resolver dei TLD quando non c'è informazione disponibile
- [IMG 7.5 pag 587]

L'informazione di un DNS

- E' costituita da quintuple:
- **Domain_name** (il nome del dominio)
- **Time_to_live** (il TTL)
- **Class** (sempre **IN** per InternNet)

- **Type** e **Value**
- [IMG 7.2 pag 583]

Vediamo ora...

- L'altro protocollo principale di Internet a livello transport: **TCP**

TCP

- **Transmission Control Protocol**
- E' in un certo senso il duale di UDP:
- Mentre UDP è ***connection-less***, TCP è ***connection-oriented***
- Mentre UDP è ***unreliable***, TCP è ***reliable***

Le connessioni TCP

- Sono ***full-duplex***
- E ***point-to-point***
- → ma **non** multicasting o broadcasting

L'header TCP

- [IMG 6.29 pag 537]

TCP

- I primi due campi sono quelli necessari anche in UDP: la porta di origine (***source***) e la porta destinazione (***destination***).
- C'è poi un campo TCP header length che appunto detta la lunghezza dell'header (variabile, visto che ci possono essere, come in IP, opzioni)
- Cosa interessante: questo campo è il ***numero di words*** (32 bits) dell'header → si è aumentato per 4 la taglia massima dell'header (cosa che non ha fatto ad esempio IP...)
- Successivamente, ci sono 6 bits vuoti, in teoria per estensioni
- Notare come a tutt'oggi sono rimasti vuoti → indice che TCP ha funzionato bene!
- C'è poi una serie di 6 flags
- **ACK** indica che c'è un acknowledgment (notare dunque come si faccia piggybacking).
- **RST** indica che c'è un problema nella connessione e serve a fare il reset (serve anche a rifiutare, per vari motivi, l'apertura di una nuova connessione).
- **FIN** serve a finire una connessione
- La terminazione avviene tramite handshacking (con ACK, anche in piggyback).
- In realtà, termina solo ***uno dei due versi*** della connessione: chi lo invia, segnala che non ha più dati da trasmettere (ma l'altro potrebbe continuare a trasmettere dati).
- Ci sono poi due flags, **URG** e **PSH**, che servono al controllo della trasmissione
- Tipicamente, il protocollo TCP non indica quando deve avvenire la trasmissione.
- Può quindi succedere che i dati restino nel buffer e non vengano spediti sulla rete, mentre magari la nostra applicazione ha bisogno di inviarli.

TCP: PSH e URG

- Ad esempio, se stiamo scrivendo su un terminale remoto, vorremmo che i dati partissero ogni volta che abbiamo finito una riga di comando
- Per far questo c'è il flag **PSH**, che segnala appunto che va fatto il PUSH sui dati, che quindi vanno inviati prima possibile.
- L'altro flag, URG, è un PUSH di livello più elevato: segnala che i dati sono veramente URGenti e prendono il sopravvento sul flusso normale di trasmissione
- Tipicamente, chi riceve un URG blocca il processo e controlla come mai un URG è stato inviato (ad esempio, CTRL-C...)

TCP

- C'è poi un campo **Window size** che gestisce la taglia della sliding window: quindi, TCP usa le ***sliding windows*** ad ***ampiezza variabile*** (max 2^16-1)

- [IMG mike]
- Di default, usa sliding windows con **goback-N**, e c'è un'opzione per usare il **selective repeat** (con i NAK).
- Notare dunque come TCP sia appunto un protocollo ***reliable***
- Infine, c'è un campo di checksum per il controllo dell'errore
- Interessante notare come si calcola il checksum (che tra l'altro è lo stesso modo di UDP)

TCP: il checksum

- Il checksum viene calcolato con semplici somme in complement a 1, e poi si fa il complemento a 1 del risultato.
- Quiz: perchè alla fine si fa un ultimo complemento a 1?
- Da notare: il calcolo del checksum non viene fatto solo sul pacchetto (header + dati), ma viene anche incluso nel calcolo un ***pseudoheader***.

Lo pseudoheader di TCP

- Essenzialmente, informazione aggiuntiva che entra a far parte del calcolo del checksum
- [IMG 6.30 pag 539]

Checksum e pseudoheader

- Notate come ci sia un mescolamento di livelli: informazione che stava nello strato ***sottostante*** IP (come gli indirizzi IP source e destination) viene estratta dal pacchetto IP e resa visibiile nel protocollo ***superiore*** TCP.

TCP

- Ci sono poi due campi: ***sequence number*** e ***acknowledgment number*** (entrambi di 32 bits) che servono a gestire le connessioni....
- ...assieme ai flag SYN e ACK

La connessione TCP: 3-way handshake

- [IMG 6.31 pag 540]

Ultima cosa da notare...

- Tutto in TCP (inizio connessione, ritrasmissione, fine connessione) è regolato da timeouts, per evitare che pacchetti persi mandino in stallo la comunicazione.

Infine...

- Ora che abbiamo visto TCP (e UDP), possiamo capire meglio come funziona la magia del NAT
- Come fa a moltiplicare un indirizzo IP in più indirizzi interni?
- La risposta: usa le ***porte*** come non si dovrebbe...

Il trucco del NAT

- Ricordate che le porte essenzialmente fanno multiplexing ***a livello transport***
- NAT usa le porte per fare multiplexing ***a livello network***: quindi abusa il concetto di porta e fa sì che uno spazio delle porte (1-65536) possa essere associato a molti indirizzi IP invece che uno solo.

NAT

- Quello che fa il NAT è quindi: crea una mappa delle corrispondenze tra porte dlel'unico indirizzo IP visibile all'esterno, e le varie macchine presenti all'interno.
- In tal modo, l'informazione su come distinguere le macchine interne viene trasportata esternamente come porta.

Notare:

- Anzitutto, che si ***mescolano i livelli*** (blub)
- E poi, peggio, che trasforma Internet in una rete ***connection-oriented***, con tutti gli svantaggi conseguenti: se il NAT box crasha, a differenza di un router, tutte le sue connessioni saranno ***perse***!

Le basi della crittografia

- C'è informazione da trasmettere (il plaintext)
- Il canale informativo non è sicuro (quindi, qualcuno può ascoltare il messaggio ad esempio).
- Vogliamo trasmettere senza che altri capiscono.
- Quindi, chi manda usa una funzione di ***encrypting*** (E), e chi riceve un'altra funzione di decrypting (D), inversa dell'altra:
- Per ogni plaintext P, **D(E(P)) = P**

Che tipi di sicurezza?

- Ci sono vari tipi di sicurezza, a seconda essenzialmente.....

Ciphertext-only

- In questo livello, l'attaccante conosce solo il testo criptato (E(P))

Known plaintext

- In questo livello, l'attaccante ha anche qualche esempio di plaintext, ed il suo corrispondente criptato (quindi ha qualche P ed il suo E(P))

Chosen plaintext

- In questo livello, l'attaccante può avere E(P) per ogni P di sua scelta.

Chiaramente...

- Quando si disegna un sistema, conviene per quanto possibile alzare il livello di sicurezza, chosen plaintext è il preferito, altrimenti known plaintext.

ATTENZIONE

- Al ciphertext-only... (es. remote login etc)

Auguste Kerckhoff

- Nel **1883**, scrive ***La Cryphtographie Militaire***
- E dà i ***sei principi della crittografia***

Le regole di Kerckhoff

- 1) Il sistema dovrebbe essere in pratica anche se non in teoria, inviolabile.
- 3) La chiave dovrebbe essere memorizzabile senza dover essere scritta, e facilmente intercambiabile
- 4) I crittogrammi dovrebbero essere trasmissibili tramite telegrafo
- 5) L'apparato crittografico e i documenti dovrebbero essere portatili e gestibili da una sola persona.
- 6) Il sistema dovrebbe essere facile, non richiedendo conoscenza di un gran numero di regole né tantomeno eccessive fatiche mentali.
- 2) Il design di un sistema non dovrebbe richiedere segretezza, e compromettere il sistema non dovrebbe creare problemi ai corrispondenti.
  E' anche detta il ***principio di Kerckhoff***

Il principio di Kerckhoff

- → l'algoritmo di encrypting deve essere pubblico (l'unica parte segreta sono le chiavi)

Quindi...

- La parte segreta è relegata alle **chiavi (keys)**
- In questo modo, le funzioni E e D dipendono dalle chiavi:
  Ek1 e Dk2
- Le chiavi possono ovviamente anche essere una sola (K1 = K2 = K)
- Mentre E e D sono pubbliche

L'opposto del principio di Kerckhoff

- Si chiama ***security by obscurity***

Vediamo ora qualche metodo: Cyphers e codes

- **Cypher:** una trasformazione carattere per carattere o bit per bit
- **Code:** una sostituzione parola per parola

Esempio di code

- film Windtalkers con Nicholas Cage, si parla di crittografia con i code
- Choctaws (1a guerra mondiale)
- Comanche (2a guerra mondiale)

Esempi di cipher

- Caesar cipher: il cifrario di Cesare
- Dyh Fhvduh!

Caesar cipher

- Agisce sulle singole lettere : ogni lettera viene spostata di un certo numero di posizioni nell'alfabeto
- Cesare usava 3 posizioni:
- a → D, b → E, c → F
- Dyh Fhvduh! = Ave Cesare!

Oltre Cesare?

- Chiaramente il modo di generalizzare quest'idea è di usare una corrispondenza qualsiasi, invece che il solo spostamento
- → una qualsiasi prefissata permutazione dell'alfabeto
- → si chiama ***monoalphabetic substitution***

Quanto efficace?

- Ci sono **26! -1** possibili combinazioni di encrypting
- → circa 403291461126605635583999999999
- Ad 1 ms per combinazione, ci vorrebbero **1000000000000 anni...**

Però...

- Quel calcolo si basa su un attacco a forza bruta
- Quando abbiamo ulteriore informazione sul sistema, possiamo fare molto meglio
- Che informazione abbiamo?
- Tipicamente, il fatto che il plaintext è in una certa ***lingua***, e parla di certi ***argomenti***

Lingua

- Significa che lettere e parole non sono casuali, ma hanno determinate distribuzioni

Etaoin...

- Ad esempio, per quanto riguarda le singole lettere, in Inglese la più frequente è "**e**", seguita da "**t**", poi "**a**", "**o**", "**i**", "**n**" etc etc...
- [IMG]

Th-in-er-an...

- Si possono poi analizzare i ***digrammi***, cioè le combinazioni di due lettere
- E oltre, i ***trigrammi***: the, ing, and, ion, etc, etc...

Inoltre...

- Sapendo più o meno gli argomenti, il vocabolario si può restringere di molto, e quindi dare ancora più struttura.
- → quando si fa un attacco crittografico si sfrutta anche tutta questa informazione.
- Quella che si chiama **frequency analysis**

Infatti...

- Durante la seconda guerra mondiale, si recrutavano le persone migliori tramite ***parole crociate*** particolarmente difficili sui giornali.

Contromisure? Non facili....

- Una serie di contromisure cerca di cambiare il testo
- Già da tempo, ad esempio con i **lipogrammi** (lèipo = lascio, gramma = lettera)

Esempi

- **Ou**vrir de **Li**tterature **Po**tentielle in Francia

- L'Opificio di Letteratura Potenziale in Italia
- Sperimentano con la cosiddetta ***letteratura vincolata***

Esempio

- **Geroges Perec** (scrittore francese), ha scritto un intero romanzo come lipogramma: La Disparition (senza la lettera “e”), e poi un altro romanzo dove invece c’è una sola vocale.

Paradossalmente...

- Alcune forme di modernità ora danno nuova linfa a questo tipo di contromisure…

In ogni caso...

- Il vero problema è che queste contromisure tendono sempre a mantenere ***principi statistici***, e quindi possono solo lenire il problema, non risolverlo.

I Transposition Cipher

- [IMG 8.3 pag 729]

Nota

- I transposition cipher, così come i substitution che abbiamo visto prima, sono sistemi a chiave simmetrica (symmetric key), cioè c'è la stessa chiave (informazione segreta sulla permutazione, o ordine delle colonne) in chi parla e chi riceve.

Domanda

- Ma c'è un modo per costruire un sistema crittografico totalmente sicuro...?
- Risposta: SI!

One-Time Pad

- E' un sistema crittograficamente sicuro, e molto semplice tra l'altro
- L'idea: si usa una chiave della grandezza del messaggio
- E si fa lo **XOR** (o addizione modulare) col messaggio stesso

Flash paper!

- Gli agenti segreti scrivono la chiave su flash paper!

Problemi?

- Poco pratica: “One-Time!”
- LA SEQUENZA DEVE ESSERE SCELTA BENE.....!!!!!
- Ad esempio, se scegliete I Promessi Sposi come chiave....
- ...finite male!

Passiamo adesso...

- A sistemi più sofisticati
- I cosiddetti sistemi **block cipher**
- Sono cifrari “a blocchi” perchè agiscono appunto su blocchi di bit (prendono n bit alla volta di plaintext e li traducono in n bit di messaggio criptato)

I block cipher

- Si compongono essenzialmente con due elementi, che possono essere composti come i mattoncini Lego

I P-box

- **P**ermutation box: fanno una permutazione
- [IMG 8.6a pag 737]

Gli S-Box

- Substitution Box (sostituiscono certi bit con altri)
- [IMG 8.6b pag 737]

I product cipher

- Sono le varie combinazioni dei due box
- Tramite vari S e P si possono fare un sacco di cose
- [IMG 8.6c pag 737]

Esempio: DES

- Uno dei primi block cipher è il DES
- DES = Data Encryption Standard
- Molto famoso: adottato nel 1977 come standard dagli Stati Uniti

Come funziona?

- E' un block cipher, con:
- Taglia del blocco: 64 bits
- Chiave: 56 bits
- 19 passaggi
- [IMG 8.7 pag 739]

DES ed il contesto storico....

- Inventato dall'IBM (Lucifer)
- Aveva chiavi di **128** bits
- Adottato come standard dall'NSA (National Security Agency)
- → chiavi di **56** bits

DES e la sicurezza...

- Chiave di 56 bits (dai 128 iniziali)
- → presto resi conto che era troppo poco
- Pochi anni dopo quindi, 1979, si cerca di migliorare la sicurezza, senza dover cambiare del tutto algoritmo

Triple DES

- Si usa tre volte DES in sequenza
- Con due chiavi (–> 112 bits)
- [IMG 8.8 pag 741]

Domanda?

- Perchè non usare tre volte encription?
- Risposta: perchè così se usate due ***chiavi uguali*** ritornate ad avere DES

Il successore di DES

- Essendo lo standard del governo USA, si è ovviamente poi cercato di alzare ancora di più il livello di sicurezza dopo che erano passati quasi vent'anni.

Cosa si è fatto?

- Invece di riprendere una soluzione proprietaria, si è scelto la strada opposta:
- Una gara aperta ed internazionale (notare: internazionale...!)

Nel 1998

- Vengono selezionati cinque finalisti
- E vince uno: Rijndael
- Fatto da due giovani ricercatori belgi (Joan Daemen e Vincent Rijmen)
- Ovviamente, Rijndael è un nome perfetto per uno standard
- → il nome viene cambiato subito!

AES

- Advanced Encryption Standard
- Cipher block con blocchi e chiavi che possono essere scelte da 128 bits a 256 bits (ricordate DES: blocchi da 64 bits, e chiavi da.....)

Un altro cipher block: Blowfish (pesce palla)

- Inventato nel 1993 (prima di AES quindi) da **Bruce Schneier**
- “un codice davanti a lui si autodecripta” (cit. Marchiori)
- E' open, e quindi ha avuto (e tuttora ha) un certo successo.
- Visto anche che è ancora oggi considerato sicuro
- Blocchi di 64 bits, e chiavi da 32 a 448 bits

- Più lento di AES

Twofish

- Versione più avanzata di Blowfish
- Cifrario con blocchi di 128 bits, e chiavi da 128 o 256 bits
- Più veloce di AES per chiavi di **256** bits.

Ultima settimana

Lezione 1 (14/03/2011)

Vediamo ora...

- Un'altra classe di sistemi crittografici, ancora più sorprendente per certi versi
- Nei sistemi a chiave unica c'è una parte di informazione segreta che viene condivisa (la chiave)
- Questo implica quindi che chi si parla deve ***prima*** in qualche modo sicuro scambiarsi questa informazione
- Come? Fuori dalla rete...!

Grosso problema...

- Non possiamo lavorare solo dentro la rete, ma abbiamo bisogno di altri canali di comunicazione ***esterni***, e ***sicuri***!
- Quindi due livelli di sicurezza...e come garantire poi la sicurezza dell'altro livello???
- Problema fondamentale: si può rimediare?
- La risposta, sorprendente per l'epoca (1976), è stata: **SI**

1976

- Diffie e Hellman scoprono un nuovo tipo di sistemi crittografici, quelli a ***chiave pubblica*** (***public-key***)
- In questi sistemi chi si parla non deve scambiarsi segretamente nessuna informazione!

Vediamo come funziona....!

- Essenzialmente, dobbiamo riuscire a scambiarci una chiave segreta, usando un mezzo insicuro, senza avere nessun segreto condiviso precedente
- (alla faccia!)
- Se ci riusciamo, possiamo poi usare uno qualsiasi degli algoritmi a chiave condivisa!
- Alice e Bob

Diffie-Hellman

- Scegliamo due numeri, **p** e **g**
- **p** è un primo, **g** è un numero minore di p con certe proprietà (non difficile da trovare)
- Ci si scambia **p** e **g** (in chiaro)
- Alice sceglie un numero a caso, **a**
- **Bob fa lo stesso, sceglie b**
- Alice calcola **g^a mod p**, è la sua chiave pubblica, che manda a Bob
- Bob fa lo stesso: calcola g^b mod p, e la manda ad Alice
- Bob riceve da Alice g^a mod p
- Calcola ((g ^ a mod p) ^ b mod p
- Per proprietà dell'aritmetica modulare, questo numero risulta essere:
- **((g^a)^b) mod p**
- La stessa cosa fa Alice, ottenendo ((g^b)^a) mod p
- Che è la loro **CHIAVE SEGRETA**!!
- Chi osserva da fuori ha visto:
  p, g, (g^b) mod p, (g^a) mod p
- Dovrebbe derivare **(g^a^b) mod p**

- Se **p** è molto grande, questo è un problema intrattabile, equivalente al ***logaritmo discreto***, che non ha soluzione conosciuta se non calcoli a forza bruta...!

RSA

- Due anni dopo la grande scoperta di Diffie-Hellman, **R**iver, **S**hamir ed **A**dleman costruiscono l'algoritmo **RSA** basandosi su idee simili.
- Solo che ora, invece di usare come "problema equivalente" il logaritmo discreto, usano un altro famoso problema.
- Vediamo come funziona RSA:
- Ad ***alto livello***, si comincia scegliendo due primi, **p** e **q**, molto grandi
- Alice calcola p*q, e lo manda a Bob
- Bob usa p*q per codificare il messaggio, che manda ad Alice
- Alice usa **p** e **q** per decodificare il messaggio
- Quindi, il problema è che per decodificare servirebbero **p** e **q**, i fattori primi di **p*q**
- → per chi attacca, il problema è equivalente a ***trovare i fattori primi*** di un numero
- Problema che ha sconfitto generazioni e generazioni di matematici, tuttora insoluto
- Più nel dettaglio:
- In RSA si scelgono due primi p e q (abbastanza grandi, almeno 1024 bits)
- Prodotto: *prod* = p*q
- Prodottino: ino = (p-1) * (q-1)
- numero primo con ino: dec
- *enc* tale che *enc* * *dec* = 1 modulo *ino*
- L'encrypting si fa con:
- **^enc (mod prod)**
- Il decrypting invece:
- **^dec (mod prod)**
- **(P ^ enc (mod prod)) ^ dec (mod prod)**
  → **P^enc * dec (mod prod)**
  → **…**
  → **P**
- Quindi, la chiave pubblica è ***(enc, prod)***
- E la chiave privata è ***(dec, prod)***
- Conoscendo i fattori di prod, saremmo in grado di calcolarci *ino* e quindi di ricavare *dec*.

$$$$$$$$$$$$$$$$$

- RSA ha per un certo tempo fatto il ***Factoring Challenge***, offrendo soldoni a chi riuscisse a fattorizzare certi dati numeri

Esempio 1

- **RSA-155** = 109417386415705274218097073220403576120037329454492059909138421314763499842889347847179972578912673324976257528997818337970765372440271467435315933543338 97
- Fattorizzazione: 102639592829741105772054196573991675900716567808038066803341933521790711307779 × 106603488380168454820927220360012878679207958575989291522270608237193062808643
- Tempo impiegato: 8000 MIPS-anni

Esempio 2

- **RSA-170** = 260626236841398449215298792666744321970859253804864064161647851918599996285420693614502839319145146186835121981648059198820530572229741164780650958098323773365107115457 59
- Fattorizzazione: 358642073042850148679980458726852042329145968105997816114023186063394845085804059396 3 ×

726702906410701907886379776392394626413613780385699667031370893600228158224958749449 3
- Fattorizzato il 29 dicembre 2009!!

Esempio 3

- **RSA-180** = 19114792771898660968922946663145464981298624627666735486418850363880726070343679905877620136513516127813425829612810920004670291298456875280033022177775277395740454049570785142104 1
- Fattorizzato l'8 Maggio 2010

Esempio 4

- **RSA-210** = 24524664490027821197651766357308801846702678767833275974341445171506160083003858721695220839933207154910362682719167986407977672324300560059203563124656121846581790410013185929961993381701214933503487587055106 7
- IRRISOLTO!

Vediamo ora...

- Qualche problema delle cifrature a blocchi
- Anzitutto, ricordiamo che sono “block”, quindi prendono blocchi e li traducono
- Inoltre, se usiamo la stessa chiave il risultato è lo stesso per ogni blocco

ECB

- Questa caratteristica è anche detta **ECB: Electronic Code Book**
- Essenzialmente, ogni blocco codificato è una pagina di un libro
- Stiamo quindi trasmettendo un libro con pagine criptate

Dov'è il problema?

- Introduciamo un altro personaggio
- Trudy / Crudelia

Vediamo...

- [IMG 8.11 pag 746]

Come si risolve?

- Usando altri modi (***cipher modes***) per fare la codifica di encrypting
- Vediamone qualcuno...
- **IV** = **I**nitialization **V**ector, trasmesso in chiaro nel flusso criptato.
- [IMG 8.12 pag 747]

Feedback Mode

- [IMG 8.13 pag 748]
- [IMG 8.14 pag 749]

Counter

- [IMG 8.15 pag 750]

Caratteristica essenziale

- Per funzionare al meglio, questi modes hanno bisogno che IV cambi ogni volta che si trasmette.
- Se IV non cambia o viene ripetuto, cosa succede?
- Vediamo...

Keystream reuse attack

- Prendiamo lo stream cipher mode
- Usiamo due volte lo stesso IV
- Abbiamo quindi due messaggi codificati fatti così:
  P1 xor K
  e
  P2 xor K
  (dove K è il ***keystream*** appunto)

- Crudelia cosa fa...?
- Semplicemente fa lo xor di questi due messaggi:
- **(P1 xor K) xor (P2 xor K) = ?**
- **P1 xor P2**
- Quindi, è sparita la parte crypto (k), e sono rimasti, combinati, i due messaggi in chiaro.
- → attacchi possibili usando frequence analysis di secondo ordine!

Solo lo stream?

- No, ovviamente anche il counter mode.

E solo quelli hanno problemi?

- Vediamo cosa succede con ogni altro modo se riusciamo lo stesso IV:
- Alice manda un messaggio criptato a Bob: M
- IV e key restano le stesse, quindi lei può mandare un altro messaggio, N

Guardiamo ora cosa succede...

- La malefica Crudelia interviene, e fa la seguente cosa: rimanda uno dei messaggi (M o N o quant'altro)
- Quindi un messaggio, diciamo M, arriva due volte a Bob
- In certi contesti non cambia nulla, semplicemente abbiamo ripetuto quello che avevamo già detto.

Replay attack!!

- In altri contesti invece, questo dà luogo al cosiddetto replay attack!
- Più precisamente, nei contesti dove i messaggi hanno effetti collaterali sullo stato di Bob
- Esempio: dove ci sono transazioni finanziarie
- Immaginate di comprare qualcosa su un e-shop, ad esempio un frigorifero
- Qualcuno vi fa un reply attack → invece di averne comprato uno, ne avrete comprati trenta!

Vediamo un altro esempio di replay attack...dove Alice e Crudelia coincidono...

- MMORPG → Phantasy Star Online

Passiamo ora...

- Dalla crittografia pura ad un altro concetto correlato
- Le firme digitali

Firme Digitali

- Il problema lo conoscete: dare l'equivalente delle proprietà della firma in formato digitale
- → si può **verificare** l'identità di chi ha firmato
- → la firma è ***vincolante (nonrepudiation)***
- → la firma è difficilissima da rifare per altre persone

Soluzioni?

- Possiamo usare le tecniche di crittografia viste finora
- La soluzione più potente è quella a chiave pubblica

Firme digitali a chiave pubblica

- Il modo è semplice, se scegliamo un modello di crittografia a chiave pubblica in cui E e D siano ***totalmente inverse***, cioè non solo
- D(E(P)) = P (com'è ovvio), ma anche
- E(D(P)) = P
- Ad esempio, RSA è totalmente inversa
- Alice prende il suo messaggio P
- Lo decripta con la sua chiave privata
- E poi lo cripta con la chiave pubblica di Bob! (e poi manda il messaggio)
- Bob può decrittare il messaggio usando prima la sua chiave privata...

- [IMG 8.19 pag 758]
- E poi la chiave pubblica di Alice
- La firma digitale di Alice è la coppia
  (P, Da(P))

Altri concetti di firma

- La firma digitale vista finora racchiude due concetti: ***autenticazione*** e ***segretezza***
- Togliendo l'autenticazione, abbiamo la segretezza, già vista con la crittografia a chiavi
- E togliendo la segretezza?

Autenticazione (senza segretezza)

- Si può fare in vari modi, ad esempio usando il concetto di hash (anche detto digest)
- Gli hash sono funzioni particolari che godono delle seguenti proprietà

Hash crittografico

- Dato un plaintext P, è relativamente facile calcolare l'hash H(P)
- Tornare indietro (da H(P)) a P) è difficilissimo
- Dato un P, è difficilissimo trovare un altro P che abbia lo stesso valore di hash
- Anche piccole modifiche del testo P producono grandi cambiamenti di hash

Torniamo all'autenticazione

- E vediamo come usando gli hash possiamo migliorare l'efficienza

Con chiave pubblica

- [IMG 8.20 pag 760]

Esempi

- **MD5** (Message-Digest algorithm 5), inventato da Rivest (R...SA!) nel 1991
- 128 bits
- **SHA-1** (Secure Hash Algorithm), dalla NSA, nel 1995
- 160 bits, blocchi di 512 bits, 80 strati
- **SHA-2** (SHA-224, SHA-256, SHA-384, SHA-512), nel 2002

Quanto sono sicuri gli hash?

- Una modalità di attacco è il cosiddetto ***preimage attack***:
- Dato un hash, trovare il messaggio corrispondente
- Oppure:
- Dato un messaggio, trovare un altro con lo stesso hash

Quindi?

- Se un hash genera **n** bits, ed è sicuro, ci vogliono circa **2^n** tentativi (forza bruta) per fare il preimage attack.

Altri attacchi...

- Si può non fare ***preimage***, se Alice coincide con Crudelia
- Supponiamo di voler fare un contratto con qualcuno usando gli hash, e fargli vedere abbiamo firmato un contratto, mentre ne abbiamo in realtà firmato un altro.
- Viceversa, fargli firmare un contratto e poi cambiargli contratto mantenendo la firma.

Come si fa?

- Dovremmo trovare due testi, P1 e P2, che hanno lo stesso hash, solo che P1 è il contratto “apparente”, mentre P2 è quello che sostituiremo dopo
- Quanti tentativi ci vogliono?

Hash = compleanno

- Hash = 1...365
- → per trovare due hash uguali, quanti tentativi dobbiamo fare?
- Se i tentativi sono 365/2, intuitivamente avremo ½ di probabilità

Quindi...

- In generale, se abbiamo un hash di n bits, quindi 2^n valori, per avere provabilità di successo ½ ci vorranno 2^n / 2 tentativi
- → **2 ^ (n-1)** tentativi
- Col birthday attack ce le hanno suonate...!
- Ci vogliono circa **2^(n/2)** tentativi

Morale

- La sicurezza di un hash ad n bits equivale a **2^(n/2)** tentativi: **come se l'hash quindi fosse lungo la metà!!**

Ad esempio...

- Usando tecniche appropriate, **MD5** è stato craccato, e quindi non è ora considerato più crittograficamente sicuro

Passiamo ora ad Internet

- E vediamo come si è posto rimedio ai problemi di sicurezza del suo protocollo fondamentale, IP

Da IP a qualcosa di sicuro...

- → **IPSec** (IP Security)
- Essenzialmente, si aggiunge ad IP informazione sulla sicurezza della transazione
- In questo modo, notare, IP passa da ***connection-less*** a ***connection-oriented*** (ma solo per la sicurezza)

Le connessioni IPSec

- Sono dette **SA (Security Association)**
- Essenzialmente, identificatori di sicurezza, che identificano quindi una connessione IP point-to-point (simplex)

Due modi di estendere IP

- O in modalità trasporto (transport mode) o in modalità tunnel (tunnel mode).

Lezione 2 (15/03/2011)

Le connessioni IPSec

- Sono dette **SA (Security Association)**
- Essenzialmente, identificatori di sicurezza, che identificano quindi una connessione IP point-to-point (simplex)

Due modi di estendere IP

O in modalità trasporto (transport mode) o in modalità tunnel (tunnel mode).

In transport mode si inserisce informazione dopo l'header IP (ora vediamo come)
In tunnel mode, si incapsula l'informazione dentro IP

IPSec in transport mode

- Tramite l'AH (Authentication Header)
- [IMG 8.27 pag 774]
- HMAC = Hashed Message Authentication Code
- → l'hash include la ***chiave simmetrica***
- Svantaggio: per calcolare l'HMAC, devo prima ricevere tutto il pacchetto

L'altro modo: ESP

- **ESP =** Encapsulation Security Payload
- [IMG 8.28 pag 775]

Passiamo ora ad un altro tipo di attacco: l'attacco MTM

- **MTM** = **M**an in **T**he **M**iddle
- [IMG MTM.jpg]
- E' un problema fondamentale!
- Fondamentalmente, rende ***impossibile*** la gestione della sicurezza senza informazione segreta

pre-condivisa...

Esempio: ricordate ARP

- Corrispondenza indirizzi IP e MAC address

Attacchi interni alla rete

- Sono i più insidiosi...
- Nel caso specifico, attacco MTM chiamato ***ARP sproofing*** (o ***ARP poisoning***)

Quindi...

- Tramite l'ARP poisoning.....

Come si fa a superare l'MTM in generale?

- In generale, come detto, impossibile, però ci si può basare sul contesto e sull'altra probabilità
- Cioè, se la maggioranza della rete funziona bene, e certe parti sono relativamente sicure, si può cercare di costruire una infrastruttura che ci protegga.

Esempio: le chiavi pubbliche

- Sono suscettibili di MTM, ovviamente
- [IMG 8.23 pag 765]

Servirebbero dei...certificati!

- [IMG 8.24 pag 766]

Serve qualcuno di affidabile...!

- Una **CA** (Certification Authority)

Collo di bottiglia o...?

- **PKA** (Public Key Infrastructure)
- [IMG 8.26 pag 769]

X.509

- [IMG 8.25 pag 768]

Autenticazione a chiave pubblica con certificazione

- [IMG 8.43 pag 798]

Senza certificati?

- [IMG 8.37 pag 792]
- Servirebbe un modo per fare autenticazione, riusando le tecniche crittografiche che conosciamo.

MTM?

- [IMG 8.38 pag 792]

E a chiave condivisa?

- La situazione ora è diversa: abbiamo informazione segreta pre-condivisa
- Quindi, almeno in teoria, potrebe.....

Vediamo un possibile protocollo di autenticazione

- [IMG 8.32 pag 787]

Possiamo fare di meglio?

- [IMG 8.33 pag 787]

Quindi, immune agli attacchi? Vediamo...

- [IMG 8.34 pag 788]

Attacco particolare

- Sfrutta il fatto che possiamo ribaltare le funzionalità di un protocollo, come allo specchio
- → ***reflection attack***

Ergo, dobbiamo tornare a quello precedente

- [IMG 8.32 pag 787]

Ma siamo veramente sicuri che vada tutto bene...?

- [IMG 8.35 pag 789]

Quindi, ancora vittime del reflection attack!

E allora come facciamo?

- [IMG 8.36 pag 790]

Torniamo ora ad internet, per vedere....

- I classici ***firewall***
- [IMG firewall]

Tra i problemi che i frewall devono affrontare...

- Atacchi come lo ***smurf***. Ad esempio, qualcuno manda dei pacchetti ICMP chiedendo l'echo
- Anche detto il ***ping of death***
- ***Se ne mando tanti.....***

In generale...

- ...questo tipo di attacchi rientra nella categoria del ***DoS (denial of service attack)***
- Cioè, grosso modo, quando chi attacca supera le risorse disponibili di chi riceve

Altro esempio

- Il SYN attack (o SYN flood)
- Iniziare connessioni TCP e lasciarle aperte contemporaneamente
- [IMG syn attack]

Da DoS a DDoS

- La cosa brutta è che ogni attacco di tipo Dos si può tramutare nella versione distribuita, quando cioè qualcuno si impossessa di molte macchine, e lancia l'attacco contemporaneamente.

Notare...

- Il DDoS sembra comunque difficile da fare, perchè richiede appunto di impossessarsi delle macchine zombie (→ difficoltà proporzionale al n. di macchine)
- Fra un po' ci torniamo....

Protezione: i firewalls

- Tipicamente fanno packet filter, cioè filtrano i pacchetti di rete
- Due grandi categorie:
    - ***Stateless***
    - ***Stateful***

Stateless

- Filtrano i pacchetti in base a come sono fatti, localmente (cioè, senza stato)
- Ad esempio, bloccare i pacchetti che vengono da un certo IP, che vanno a certe porte, etc...
- Vantaggi: il filtraggio è veloce

Stateful

- Usano anche l'informazione sullo stato (tipicamente, le sessioni attive), per decidere quali pacchetti accettare
- Quindi, possono usare informazione pertinente a dove ci si trova nella sessione
- Più oneroso della stateless, ma più efficace

Oltre lo stato network

- Il **DPI** (Deep Packet Inspection)
- Può entrare dentro ai pacchetti, e controllare ogni loro parte
- Eccettuato lo strato fisico (ovviamente), ***ad ogni livello nella scala dei layers***

Ergo...

- Un buon firewall basta a proteggerci dagli attaccanti: se qualcuno ci invia traffico anomalelo possiamo bloccarlo.
- Giusto?

Quasi...

- Torniamo al DDoS

E usiamolo in un altro modo...

- L'attacco DDoS può essere più sottile di quanto si pensi, usando la modalità ***reverse***.

Modalità reverse: l'attacco IP spoofing

- L'IP spoofing si basa sul fatto che IP non ha alcuna sicurezza
- Quindi quello che possiamo fare è ad esempio ***scrivere un altro mittente***
- Perchè mai dovremmo farlo?

Per fare DDoS...

- Ricordate il ***ping of death***... vi basta ora inviare una richiesta ICMP di echp a tantissime macchine, usando come mittente la macchina da attacare...
- Non avete dovuto prendere il controllo di nessuno zombie!

Vediamo ora...

- [IMG 8.46a pag 807]
- I DNS...!

I DNS sono un ottimo bersaglio...

- ...per una varietà di motivi:
- Fanno parte dell'***infrastruttura critica***
- Offrono informazione di ***livello superiore***
- Sono relativamente ***indifesi***

DNS attacks

- Due grandi classi di attacchi ad un DNS:
- Ai suoi ***dati***
- Al ***server*** stesso

Attacco ai suoi dati: DNS Spoofing

- [IMG 8.46b pag 807]

Però....

- Se non facciamo ***tapping*** fisico....risponde anche il DNS vero
- E quindi? Mettetevi nei panni di un DNS...cosa può fare?

Ad esempio...

- Può usare un numero di sequenza
- E con quale politica?
- Coda univoca!

Allora...

- [IMG 8.47 pag 808]

Attacco al server DNS

- Potete fare DDoS contro un server DNS
- In tal modo buttate giù quel server e quindi di lì a poco (quando i TTL finiscono) tutta una rete.

E ancora altri attacchi...

- Non ***contro***, ma ***con*** i server DNS...
- Fate DDoS contro una macchina, usando invece di ICMP il protocollo DNS...
- Attacco chiamato ***DNS amplification***

Il brutto...

- E' che il protocollo DNS tipicamente passa ***oltre i firewall*** (porta 53)!

Lezione 3 (16/03/2011) – last day with MM :''(

Una parziale soluzione...

- **DNSsec** (DNS Sicuro)
- Essenzialmente, si usa crittografia a chiave pubblica (RSA)

- E a cascata si propaga l'informazione firmata
- Anche qui, solito problema dell'informazione iniziale...

Inizio?

- I root servers (che ci sono già ora...)
- Quanti? **13** principali (distribuiti: in realtà **241**)
- [IMG mappa root servers]

Torniamo ora...a 802.11

- Ricordate, avevamo detto che usa la banda a **2.4GHz** (quindi, interferenza con 802.11b ed altri, banda ISM, senza licenza...!)
- E avevamo detto che un tipo di trasmissione che si usa è FHSS... che spieghiamo ora.

FHSS

- Frequency Hopping Spread Spectrum

Hedy Lamarr (1913-2000)

- Definita “la donna più bella del cinema”
- Famosa anche per essere la prima donna mai apparsa nuda sullo schermo (*Extase*, 1933, Mostra del Cinema di Venezia)
- E per i suoi baci da 25 mila dollari...(!)

1939-1945

- La Seconda Guerra Mondiale
- Heidi Lamarr è una donna impegnata, e fa di tutto per poter aiutare nella guerra
- Tra le varie cose, i famosi **baci da 25mila dollari**
- Gossip dell'epoca: 7 milioni di dollari raccolti in una sola serata!
- Nel 1941, a Hedy Lamarr viene un'idea....
- ***Brevettata*** nel 1942...

L'idea...

- Cambio frequenza mentre sto trasmettendo, così se un tedesco è in ascolto, è costretto a cercare una nuova frequenza. L'idea è di usare gli spartiti musicali. Durata di una nota = tempo di attività su una data frequenza.
- Frequency hopping...!
- …viene scartata da Washington (perchè lei era una donna!), ma poi, dal 1962 in poi (blocco di Cuba) entra a pieno regime nelle telecomunicazioni wireless, usando sequence one-time o sequenze pseudo-random!!!

Sicurezza di 802.11?

- 802.11, oltre ad FHSS, usa anche altre tecniche di sicurezza: vediamo le principali.

WEP

- **W**ired **E**quivalent **P**rivacy
- Funziona usando encrypting a ***chiave simmetrica***
- Usa un block cipher chiamato RC4 (Dove la R qui sta sempre per Rivest)
- Chiaramente deve proteggersi dal problema dell'Electronic Book Mode → usa una tecnica di chaining

Lo stream cipher!

- [IMG 8.14 pag 749]
- [IMG 8.31 pag 782]

WEP

- Il primo standard WEP (WEP-40), ha una chiave di 40 bits.
- Che viene aggiunta ad un IV lungo 24 bits, e quindi processata in uno stream cipher mode
- Sono state poi introdotte versioni con chiavi

I problemi del WEP...

- L'IV è di 24 bits
- → ce ne sono 2^24 = 16777216
- → dopo al più 16777216 pacchetti, siamo costretti a riusare lo stesso IV (e quindi, la stessa chiave di crypting...!)
- A 3000 pacchetti al secondo, significa circa ***un'ora e mezzo***

Però...

- Un'ora e mezzo per avere probabilità **1**
- In realtà, siamo nella classica situazione da birthday attack...
- → si ripete una chiave ***ogni secondo*** (!!)

Nel 2001...

- Si scopre l'attacco “FMS” (Fluhrer, Mantin and Shamir)
- Essenzialmente, si scopre come craccare RC4: per molte chiavi, dato il keystream si possono ricavare dei bits del plaintext!
- Potenza dell'attacco: ci vogliono ***dai 4 ai 6 milioni di pacchetti*** per impadronirsi della chiave.
- → a 5000 pacchetti/sec, circa ***15 minuti***

Nel 2004...

- L'attacco “KoreK”
- Un hacker che posta il suo attacco su Internet
- Sfrutta sempre le debolezze di RC4
- Potenza dell'attacco: 700000 pacchetti per impadronirsi della chiave
- → a 5000 pacchetti/sec, fanno poco meno di ***due minuti e mezzo***

Nel 2007...

- L'attacco “PTW” (Pyshkin, Tews e Weinmann)
- Potenza dell'attacco: 35000-40000 pacchetti per impadronirsi della chiave
- → a 5000 pacchetti al secondo, fa ***meno di un secondo***!

L'8 novembre 2008...

- L'attacco “BT” (Beck e Tews)
- Potenza: 24000 pacchetti per impadronirsi della chiave
- → ***meno di mezzo secondo***

Solo teoria?

- Andiamo in un'altra parte del mondo...Miami

E andiamo da Marshalls (catena di supermercati molto famosa)

- Durante il pagamento con carta di credito, i lettori di carta di credito, passavano i dati via wireless con protezione WEP

Risultato?

- **45.7 MILIONI** di account di carte di credito rubati! (Guinness World Record)
- (e innumerevoli altri dati sensibili)
- La più grave violazione si sicurezza mai registrata al mondo

Si è quindi dovuto cambiare...

- Fine 2002 → **WPA** (Wi-Fi Protected Access)
- Usa **TKIP** (Temporal Key Integrity Protocol), basato su RC4
- Chiave simmetrica, di 128 bit (**WPA-PSK**, Pre Shared Key)
- Chiave anche inserita come ***passphrase*** (8-63 caratteri ASCII)

Problema...

- Sociale: passphrase da 8-32 caratteri ASCII → moltissimi usano passphrases corte e simili a parole di dizionario, mai veramente casuali.
- Quindi, chiavi suscettibili di attacchi di tipo dictionary

8 novembre 2008...

- Martin Beck e Erik Tews:
  “Pratical attacks against WEP and WPA”
  (stesso articolo dell'attacco BT visto prima per il WEP...)
- Mostrano come si possano iniettare fino a 7 pacchetti ARP dentro ad una trasmissione WPA

WPA2

- Derivato dallo standard **802.11i**
- Usa **CCMP (Counter Mode with Cipher Block Chaining Message Authentication Code Protocol)**
- Chiave simmetrica da 128 bits, e blocchi sempre di 128 bits.

Ottobre 2008...

- Elcomsoft annuncia di avere velocizzato di un **fattore 100** gli attacchi contro **WEP**, **WPA** ed anche **WPA2**
- Usando tecniche di GPGPU

GPGPU

- GPGPU = General Purpose computation on GPU
- L'idea è interessante: si usa la GPU per fare calcoli di un certo tipo, veloci e paralleli (pensate agli shader models...)

CUDA

- L'architettura GPGU di NVIDIA
- Presente praticamente su tutte le lultime schede GeForce, Tesla, Quadro

Elcomsoft e CUDA

- [IMG password recovery speed]

“Return of the caveman”...Potenza di Calcolo?

- Anche se non uso un algoritmo efficientissimo,

Jaguar!

- Il più potente supercomputer al mondo!
- Della Cray Corporation (il forza agli Oak Ridge National Laboratory nel Tennesse)
- **224256** processori X86 Opteron
- Potenza: **1,75 petaflops** (1750000000000000 FLOPS)

Ottobre 2010...

- Tianhe-1a
- In forza al National University of Defense Tenchology (NUDT), a Tianjin
- Potenza: **2,507 petaflops**
- Come hanno fatto?

Jaguar contro Tiahne-1a

- Jaguar: **224256** processori Opteron
- Tiahne-1a: **7168** “unità”
- Che razza di unità sono...???
- Dotazione di ogni unità:
- **Un** processore Xeon X5670
- ...e due...

GPU NVIDIA Tesla M2050 “Fermi”

- Simile alla linea “consumer” della Geforce GTX470
- **3 GB** di memoria RAM
- **448 cores CUDA**

Capite quindi...

- Come l'attacco “return of the caveman” sia solo questione di tempo, quando le schede

grafiche scenderanno di prezzo...

Gennaio 2011...

- Thomas Roth usa il servizio di Amazon (**Elastic Computer Cloud (EC2)**)

Configurazione macchina EC2

- 67 processori quad-core Xeon X5570
- 22GB di RAM
- 1,7 Terabyte spazio disco
- ...e due GPU NVIDIA Tesla M2050 “Fermi”!

Risultato?

- Potenza dell'attacco:
- 400000 passwords al secondo
- → Riesce a **craccare WPA** in circa **20 minuti** (e promette di farlo in **6 minuti** migliorando il codice)
- Costo dell'affitto? Circa **$1.68**

La soluzione **Definitiva**??

- **Security Mantra**: *Il solo modo per avere veramente sicurezza è di non essere connessi alla rete!!!*

Per finire...

- Vediamo un filmato.

Morale:

- Un punto debole si trova sempre prima o poi
- → Il solo modo per avere veramente sicurezza è pianificare sempre per il caso peggiore...!