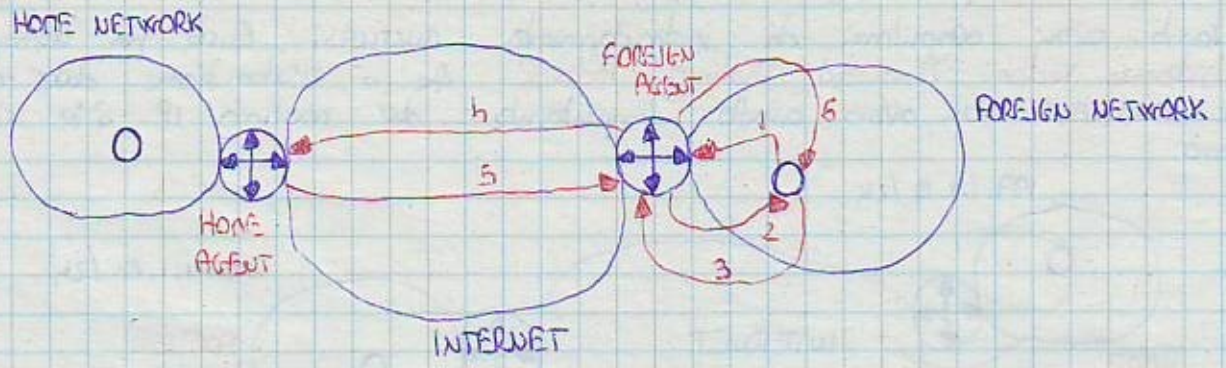


Quindi è il Foreign Agent che fa arrivare l'indirizzo care-of-address al mobile host. Così non vengono consumati inutilmente indirizzi IP nella rete straniera. Una volta scoperto il Foreign Agent:

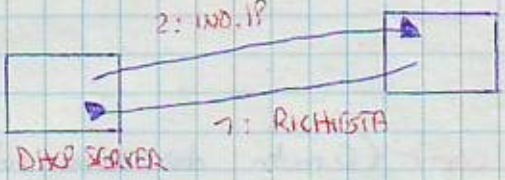
1) passo mandare un ICMP ROUTE SOLICITATION

2) passo imbustarlo in un pacchetto IP con D.A pari a 224.0.0.11 che è l'indirizzo MULTICAST speciale chiamato ALL AGENTS GROUP.

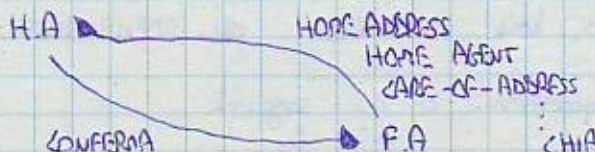
Quindi il Foreign Agent risponde con un ROUTER DISCOVERY ICMP. Il messaggio di risposta contiene l'indirizzo IP di quel router. Dopo fa conferma da parte dell'HOME AGENT, il Foreign Agent invia all'HOST MOBILE un care-of-Address fatto in questa maniera:

TYPE	---	---
LIFE TIME		---
CARE_OF_ADDRESS		

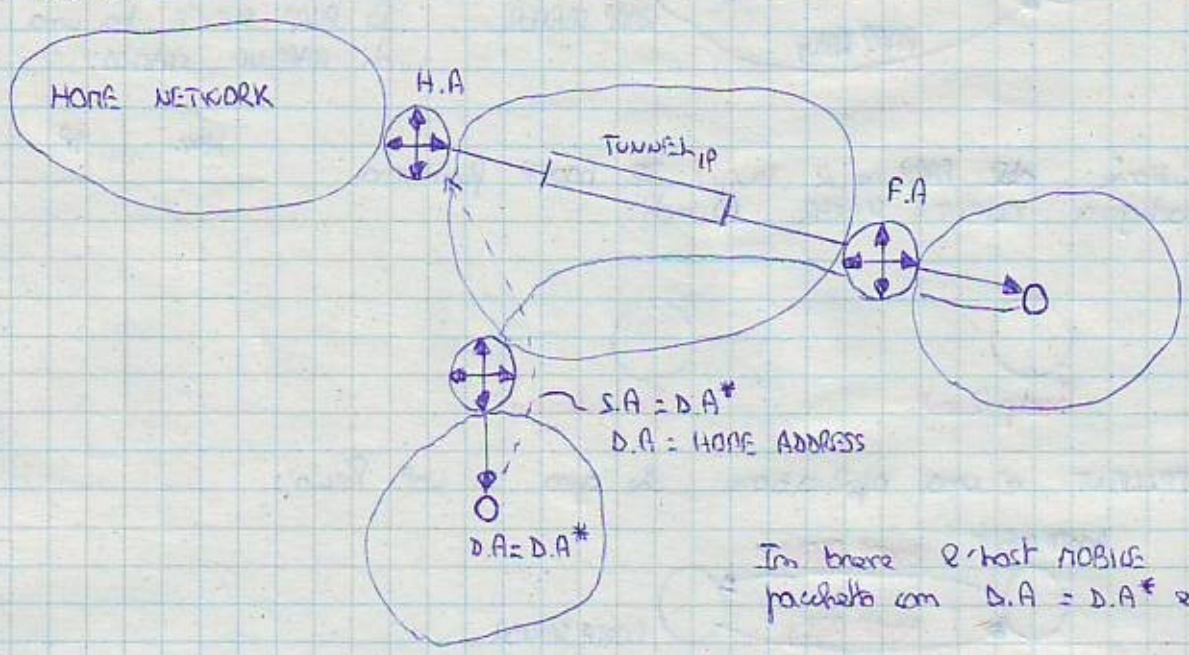
Esiste un care-of-Address di tipo CO-LOCATED dove non è più necessario il Foreign Agent, per assegnare il care-of-Address al MOBILE HOST, ma quest'ultimo può rivolgersi al server DHCP. In breve il MOBILE HOST si comporta come una macchina con gli stessi diritti delle altre macchine della rete straniera.



mentre con il Foreign Agent si avrà:

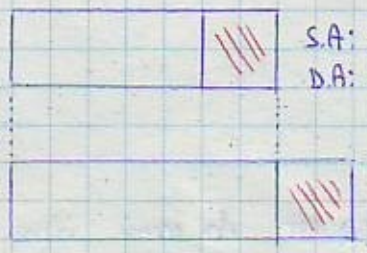


Puntiamo il care-of-address è un indirizzo del Foreign Agent. Quando giunge un pacchetto al Foreign Agent, come fa quest'ultimo a capire a quale host mobile mandarlo? È necessaria una procedura ARP evoluta e specifica. Come comunica l'HOST MOBILE?

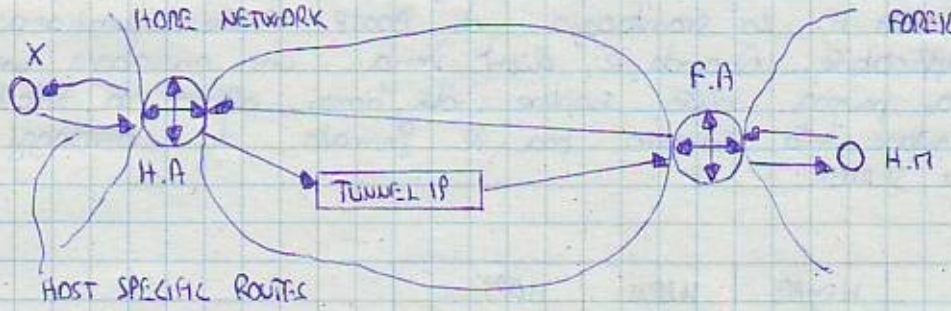


In breve l'host MOBILE genera un pacchetto con D.A = D.A* e S.A = HOME ADDRESS.

Quindi:



Il Foreign Agent la sbusta e la inoltra all'host MOBILE. Supponiamo ora di avere la seguente situazione:



Ipotizziamo che 'X' voglia parlare direttamente con l'host MOBILE e spedisce un pacchetto con D.A = HOME_ADDRESS. Questo pacchetto può anche non riuscire a uscire dalla HOME NETWORK. Questa cosa si risolve usando procedure esplicite come PROXY ARP. L'Home Agent si pubblicizza come ARP SERVER cercando di catturare tutte le richieste di ARP. E' quindi l'HOME AGENT con l'HOST SPECIFIC ROUTES che specifica il fatto che quell'indirizzo lo raggiunge lui. Parliamo ora della configurazione degli host. Abbiamo già visto una procedura per la configurazione degli host.

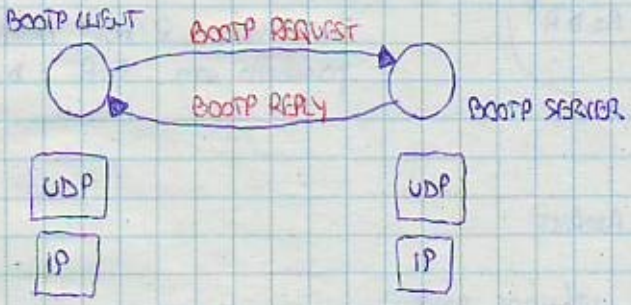
- RARP che è limitato. È basilare e radicale ma è in quasi completo disuso.

Il RARP SERVER ha una tabella di MAPPING STATICO.

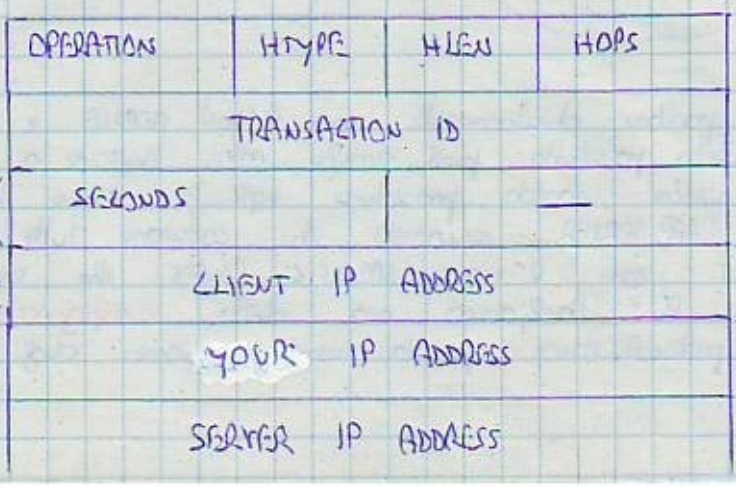
HW	IP
...	...
...	...
...	...

- Un'evoluzione del RARP è il BOOTP. Il BOOTP funziona sul paradigma CLIENT-SERVER. Quindi:

Il BOOTPCLIENT è una applicazione che gira su UDP. Perciò:



Il BOOTPCLIENT, da quando nasce, invia il BOOTP REQUEST. Ma come viene inviato? Lo si invia in broadcast limitato. BOOTP tende a funzionare ad alto livello e quindi si sogancia dal livello hardware per avere così una funzionalità più generale. Invece RARP è legato agli indirizzi hardware e quindi è limitato. BOOTP è più evoluto. Anche BOOTP REPLY viene spedito in broadcast limitato. BOOTP funziona a livello IP. Lo svantaggio di BOOTP è che funziona su UDP che è leggero e inaffidabile. Quando il client invia un messaggio un timer. Se la risposta arriva prima dello scadere del timer, allora va tutto bene, altrimenti bisogna rifare tutto. Vediamo ora il formato dei messaggi BOOTP trasmessi tra CLIENT e SERVER.



→ identifica la transazione.

specifica il numero di secondi che sono passati da quando il client ha iniziato la procedura di BOOTSTRAP.

→ è significativo per una risposta

→ dipende dal contesto

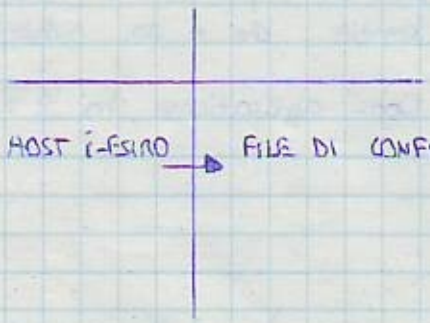
→ è strettamente collegato al SERVER HOST NAME.

ROUTER IP ADDRESS	→ indirizzo di default gateway.
CLIENT HARDWARE ADDRESS	
CLIENT HOST NAME	
SERVER HOST NAME	→ 64 byte
BOOT FILENAME	→ 128 byte
VENDOR SPECIFIC AREA	→ 64 byte

Il client ha memorizzato su HD il nome del server.

Il campo OPERATION ci dice se si tratta di REQUEST o REPLY. Il campo HOPS ci da un'idea del fatto che il client inizia mettendo HOPS = 0. HTYPE e HLEN ci fanno sapere la tipologia di hardware che c'è sotto.
Se siamo in presenza di grandi reti, un BOOTP SERVER può iterare il tutto ad altri BOOTP SERVER. In questo caso incremento il campo HOPS per ogni BOOTP SERVER attraversato. Così sappiamo chi ci ha risposto. Il campo BOOT FILENAME viene utilizzato per caricare da remoto un file di configurazione per esempio delle host anche se non è collegato alla rete. I limiti principali del BOOTP sono:

1) Non è dinamico nel senso che essenzialmente il BOOTP SERVER deve avere una tabella che è equivalente della ARP CACHE TABLE.

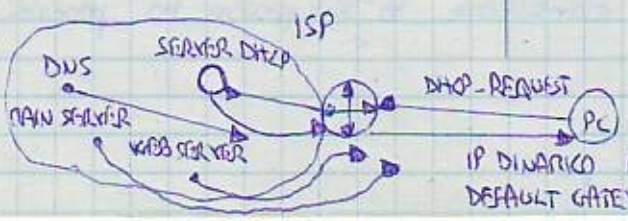


La tabella di configurazione è statica e va configurata a mano. Così facendo però nasce una difficoltà nella gestione.

BOOTP è stato poi evoluto nel DHCP.

- DHCP = Dynamic Host Configuration Protocol ha le tabelle di configurazione gestibili in modo dinamico.

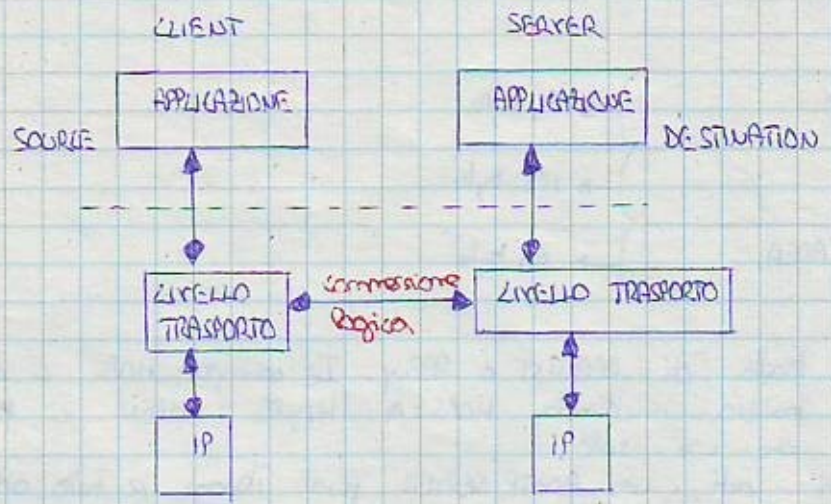
Quindi il nostro ISP sarà:



IP DINAMICO rilasciato da DHCP SERVER. È un DEFAULT GATEWAY.

(7h)

Vediamo ora UDP e TCP. Analizziamo la seguente situazione:

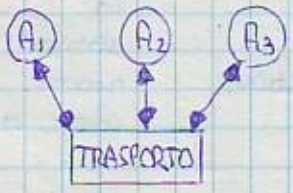


I livelli di trasporto per prima cosa creano una connessione logica.

Il socket viene identificato dalla coppia (PORTA, IP) → socket source. Il destination viene identificato dalla coppia (PORTA, IP) → SOCKET DESTINATION. Una connessione logica viene dunque identificata da:

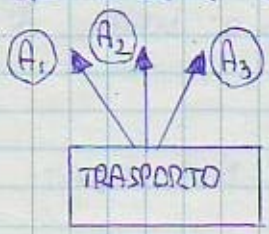
(SOCKET (S), SOCKET (D)).

In un pacchetto IP è presente tutta una serie di informazioni sulla coppia di socket. La porta di destinazione identifica il servizio che si sta richiedendo. Per esempio:



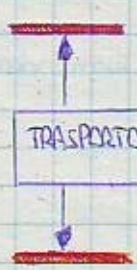
Ogni applicazione ha il suo numero di porta.

All'interno delle richieste ci sono tutta una serie di informazioni sulla porta di destinazione. Tale porta viene creata dal livello trasporto per decidere a chi mandare le informazioni. Nel caso ricevente ci sarà un'operazione chiamata demultiplazione del seguente tipo:



I numeri di porta standardizzati di destinazione vanno da 0 a 1024. I numeri maggiori di 1024, nel caso serva, possono essere usati se serve la mia applicazione specifica.

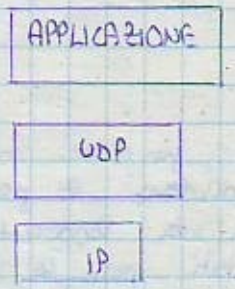
Si noti che la socket PORT mi identifica una singola connessione. L'interfaccia tra l'applicazione e il livello trasporto si chiama, come abbiamo già visto, SOCKET INTERFACE. Sostanzialmente è un'API che è dotata di una serie di chiamate e primitive. Per aprire e gestire una connessione lo sviluppatore ha parecchi strumenti a sua disposizione.



interfaccia superiore di tipo stream. Il flusso di dati non ha alcuna struttura.

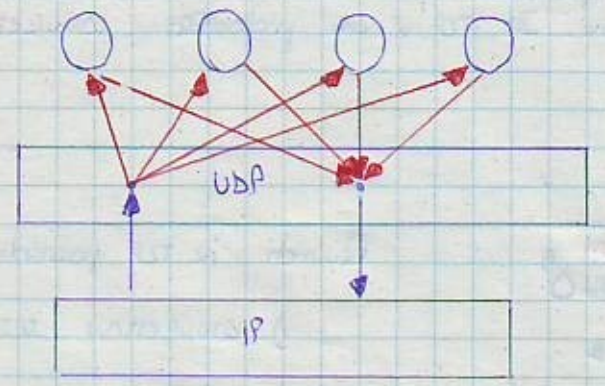
interfaccia inferiore di tipo a blocchi perché la comunicazione avviene tramite segmenti, e non avviene byte per byte.

Vediamo ora UDP. UDP = User Datagram Protocol ed è il livello di trasporto più semplice.



I pregi del UDP sono la semplicità la leggerezza e la velocità. UDP è unreliable cioè non effettua la ritrasmissione di segmenti che sono andati persi. Qui è l'applicazione deve garantire l'affidabilità del trasporto.

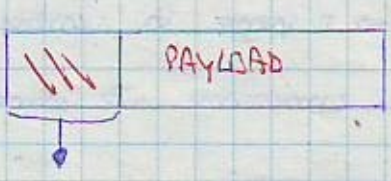
Ma vediamo cosa fa UDP! UDP esegue la segmentazione del flusso di byte che arrivano dall'applicazione e poi esegue le operazioni di MULTIPLAZIONE / DEMULTIPLAZIONE.

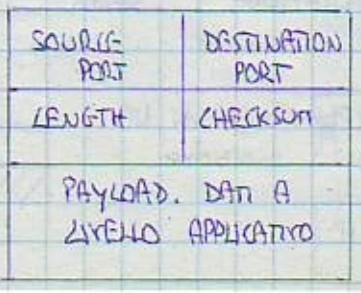


Quindi si ha:

- MULTIPLAZIONE in trasmissione
- DEMULTIPLAZIONE in ricezione.

L'UDP però non gestisce le perdite di informazioni, le duplicazioni, e non esegue il controllo di sequenza. Non esegue neanche un controllo del flusso, un controllo della congestione, e tante altre cose che il TCP gestisce. Vediamo come è fatto il segmento UDP:

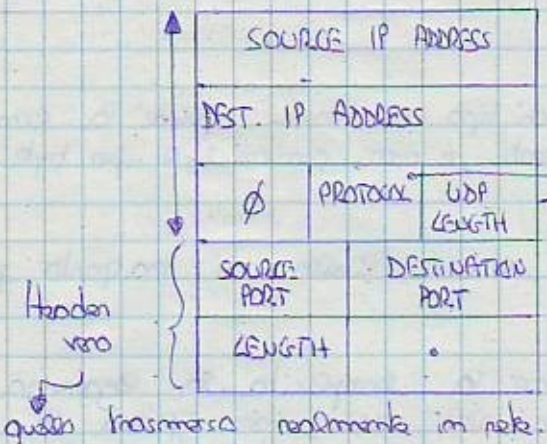




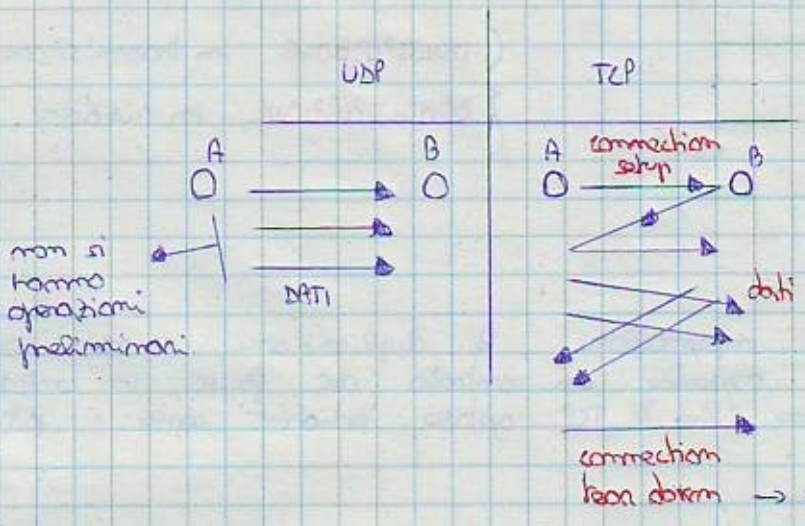
Header di 8 byte.

Il comp. checksum può non essere usato. Per quanto riguarda il suo calcolo andrebbe calcolato sulla SOURCE PORT, DESTINATION PORT, LENGTH.

Im UDP però tale calcolo avviene in modo strano. Il checksum viene calcolato sullo pseudo-header.



Questo pseudo-header serve per un ulteriore grado di sicurezza, per errori nel routing o malfunzionamenti dell'IP. Nel caso di VATBOX, esso deve ricalcolare lo pseudo-header il checksum e deve ricalcolare e/IP. Quindi è UDP, data la sua leggerezza e semplicità, va bene per applicazioni REAL-TIME, o per applicazioni dati per le quali la perdita di qualche informazione non è critica. Vediamo ora il TCP. Il TCP è molto più complesso dell'UDP ma molto più affidabile. Questo perché garantisce la correttezza delle informazioni trasportate. Inoltre garantisce la corretta sequenza per cui i byte da 1000 a 2000 arrivano dopo dei byte da 1 a 1000. Il TCP segue il controllo del flusso e della congestione. Il TCP è un protocollo CONNECTION ORIENTED mentre UDP è CONNECTION-LESS.



Quindi il TCP prevede 3 fasi:

1) CONNECTION SETUP
2) DATI
3) CONNECTION TEAR DOWN

→ si spegne la connessione logica

Lo scopo di tutto ciò è il controllo della connessione. Il connection oriented abilita il controllo.