# NAT

Internet nel contesto del livello di rete si intende insieme delle entità del livello di rete che stanno nei router e negli host. Essere connessi vuol dire che si forma un'unica rete. Un router in internet ha più indirizzi IP, uno per ciascuna interfaccia. Un host ha un unico indirizzo IP.

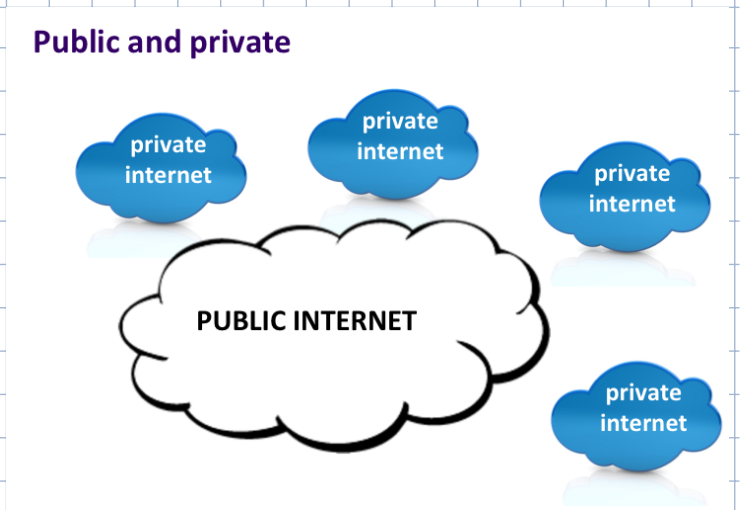
Internet è uno solo, ogni entità ha un indirizzo univoco.

Nulla vieta che le aziende creino una propria rete privata, completamente sconnessa dalla rete pubblica.

Quanto vale per internet pubblico, vale anche per le reti private di internet. Sono come delle isole autonome.

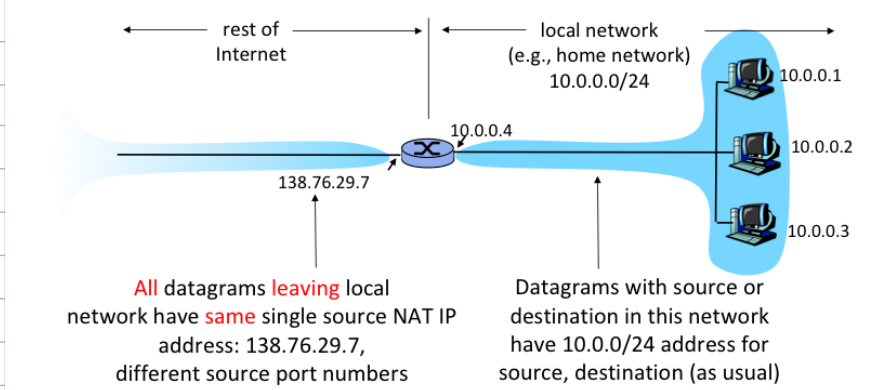
Gli indirizzi IP sono univoci nella rete di internet privata.

Tecnologicamente le reti private sono identiche, si differenziano in ambito amministrativo e giuridico.




Quindi gli indirizzi sono univoci nell'ambito di una rete, che sia pubblica o privata.

Il problema nasce quando si è voluto connettere la rete privata alla rete pubblica.

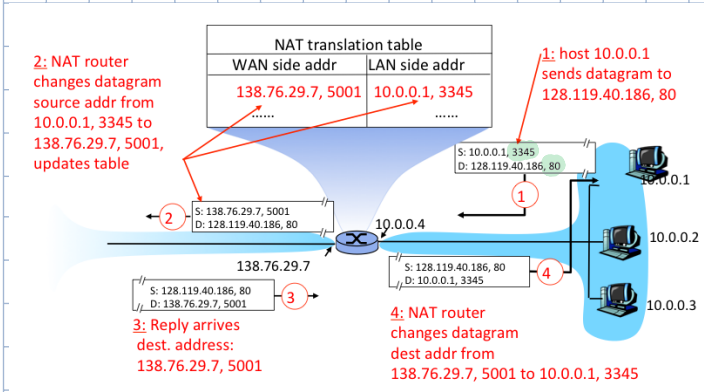



Se volessi interconnettere la rete privata con quella pubblica lo faccio attraverso un router.

Dal punto di vista della rete pubblica il router è visto come un host in quanto è come se avesse una sola interfaccia (non importa quello che c'è dietro), in quanto è una sorgente di pacchetti.

Il problema nasce quando un host vuole scaricare dei file da internet. Ad esempio se il serve deve mandare un pacchetto destinato all'host 10.0.0.1 questo non può farlo perché è un indirizzo non univoco perché potrebbe appartenere a diverse reti private contemporaneamente.

Processo software per risolvere questo problema: NAT. Fa una traduzione di indirizzi a livello di rete.




Il NAT ha una tabella divisa in WAN (rete pubblica) e LAN (rete privata). Inizialmente questa tabella è vuota. Ogni volta che si scrive nella tabella c'è sempre un azione che cancella anche. Poichè non c'è connessione per ogni riga che si scrive c'è un tempo di vita, se la riga non viene mai utilizzata ad un certo punto viene eliminata.L'host emette un pacchetto che parte da lui destinato a un host

della rete pubblica. In questo momento arriva una violazione, i numeri 3345 e 80 sono dei campi che stanno nell'intestazione della PDU nel payload del pacchetto  e servono ad indentificare il codice del processo applicativo a cui si vuole mandare l'informazione.
Questi numeri sono porta sorgente e porta destinazione e stanno nell'intestazione di UDP e TCP che si trovano nel carico pagante del pacchetto.
Quindi il NAT prende delle informazioni che non potrebbe avere.
Il router preleva i campi porta sorgente e porta destinazione e scrive i campi nella parte di tabella della rete privata la porta destinazione mentre nel lato pubblico ci mette la porta sorgente ma si inventa un nuovo numero come per dire che la risposta l'ha generata lui.
Questo pacchetto modificato arriva alla destinazione in quanto la rete pubblica ha emesso un indirizzo della rete pubblica e quindi sa dove consegnarla.
Quando questo pacchetto arriva al router  e si crea l'ambiguità: il pacchetto è destinato al router o deve rilanciarlo? Il router consulta prima la tabella NAT e controlla se c'è una riga con l'indirizzo che ha ricevuto, quindi in quel caso capisce che non è destinato a lui e risostiuisce il il codice identificativo e lo manda nella rete privata.

Quindi il NAT permette  alle reti pubbliche e le reti private  di connettersi.

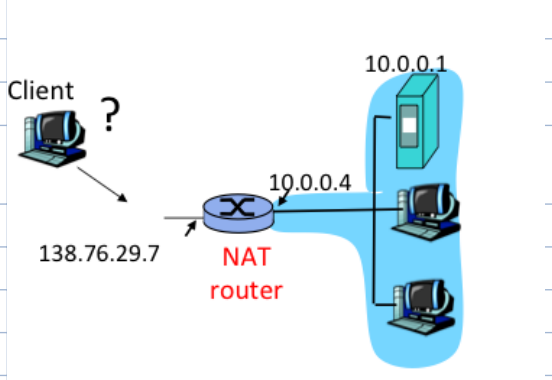
L'host non si rende conto  di cosa succede nella rete pubblica, non sa che c'è il NAT.

Questo processo ha consentito nel tempo di espandere internet.

- il NAT è controverso: viola i sacri principi dell'architettura protocollare, usa i numeri di porta e non protrerebbe.

Inoltre i numeri di porta stanno solo in UDP e TCP, cosa succede se questi ultimi due non ci sono?
Ogni volta che ci sono dei pacchetti senza questi bisogna pensare ad una variante ad hoc per far funzionare il NAT. Questa variante deve essere ogni volta specifica proprio perché si stanno violando i principi dell'architettura protocollare.

Altra problematica: **NAT traversal problem**

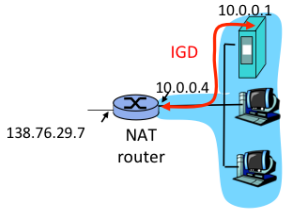



Tutto funziona quando il client si trova nella rete privata. Se invece il client è al di fuori.
Si risolve con la

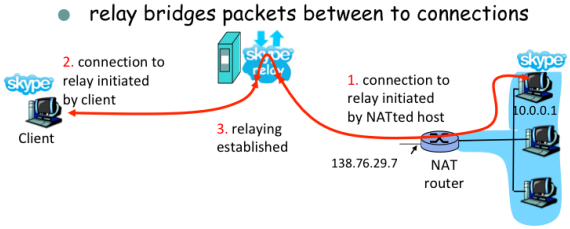
- solution 1: statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (138.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000

- solution 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol.  Allows NATted host to:
  - learn public IP address (138.76.29.7)
  - add/remove port mappings (with lease times)

  i.e., automate static NAT port map configuration



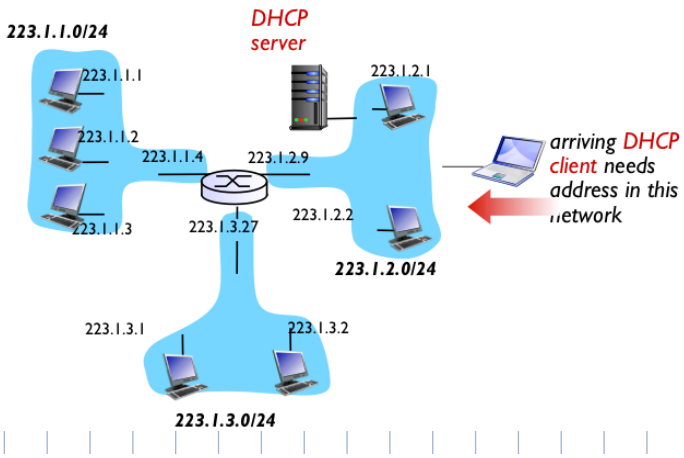

- solution 3: relaying (used in Skype)
  - NATed client establishes connection to relay
  - External client connects to relay
  - relay bridges packets between to connections



## Come si ottiene un indirizzo IP

Ci sono due modalità: statica (configurata a mano, si inserisce da amministratore) e dinamica ( si effettua un protocollo in modo automatico).
In ambedue i casi configurare l'indirizzo vuol dire attribuire all'indirizzo IP di ogni interfaccia dei parametri specifici.

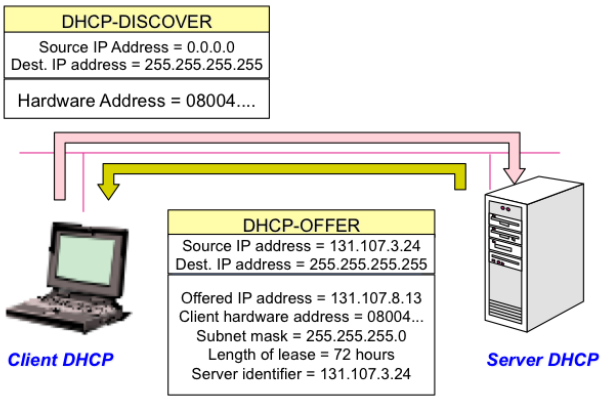
## DHCP

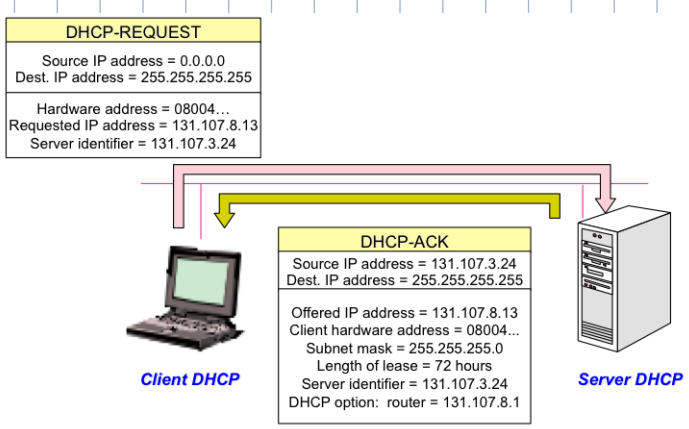



È come una transazione commerciale. Per avviare questo protocollo sono necessari due presupposti : la connessione alla sottorete per connettersi alla rete locale (in quel momenti si attiva il DHCP), non ho l'indirizzo IP.


Il primo messaggio viene dall'host (discovering), chiede se c'è un server DHCP. La risposta, se c'è un server DHCP è l'offerta. Allora il client fa una richiesta (vuole un indirizzo) e l'ultimo messaggio è il riscontro da parte del server.

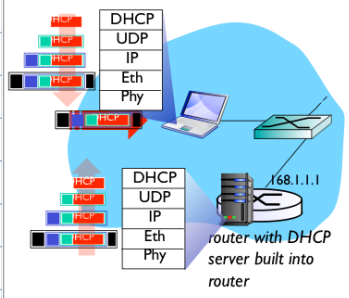
Questo programma viene eseguito al livello applicativo. Quindi chiede al livello di trasporto e poi al livello di rete. Il livello di rete deve creare un pacchetto dove inserisce l'indirizzo IP, tuttavia non ha nulla da mettere nel source address (perché l'indirizzo IP lo sta chiedendo) quindi mette tutti 0. Nel destination address invece tutti 1, quindi lo manda a tutti. Tutti gli host quando aprono i pacchetti poi lo buttano perché non ha l'applicazione, mentre se lo apre DHCP prende l'indirizzo che aveva richiesto.



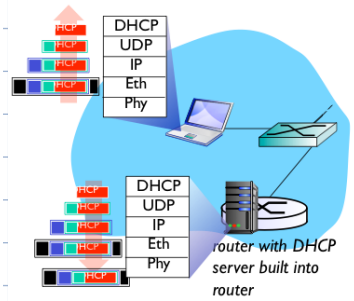




Il server DHCP si installa nel router.

## Esempio:

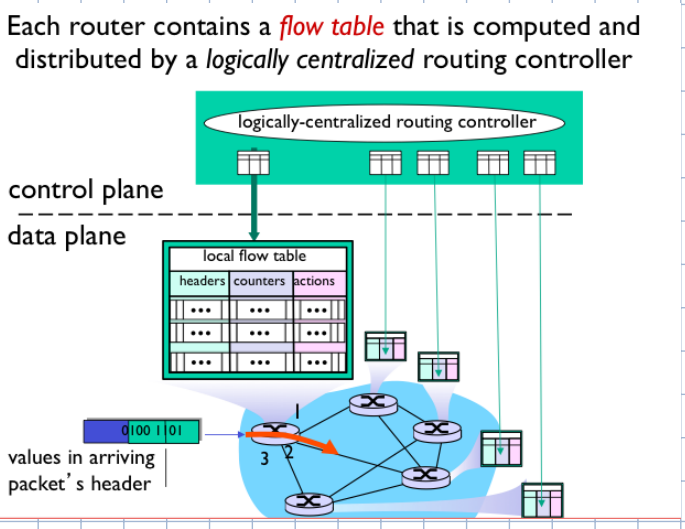



- connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP




- DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulation of DHCP server, frame forwarded to client, demuxing up to DHCP at client
- client now knows its IP address, name and IP address of DSN server, IP address of its first-hop router

## Inoltro generalizzato e SDN




Se il pacchetto non è destinato a un router della sottorete, devo trovare un router che si avvicina alla destinazione. Questo si fa consultando una tabella, si preleva il destination address dal pacchetto utilizzo la chiave sulla tabella e trovo la riga che fa match che mi dice il router con il suo indirizzo IP a cui devo dare il pacchetto.

Se ci sono più righe che concordano con la chiave, si sceglie la riga che fa match più lungo (longest prefix matching).

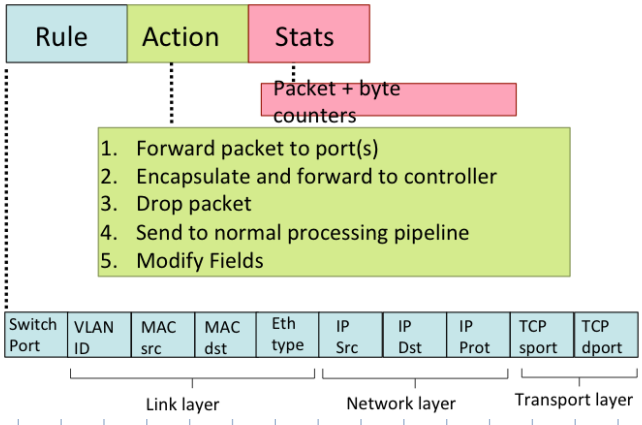
La tabella di inoltro del switch viene riempita con l'algoritmo di auto apprendimento, quindi non è certo di quello che fa quindi non butta la trama se non coincide con nessuna riga, quindi la manda a tutti.

La tabella di inoltro invece il router l'ha ottenuta attraverso la comunicazione con tutti i router, quindi se dopo tutto ciò non c'è match con nessuna sottorete allora si butta.

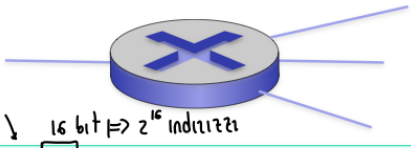
I router nelle loro tabelle hanno elencate tutte le sottoreti o solo una parte + un otherwise, ossia una riga di tutti asterischi che fa match con qualsiasi indirizzo.

L'inoltro generalizzato...
Ogni riga della tabella contiene : regola, azione e statistica.

L'ordine nelle tabelle è importante e le regole vengono scandite uno per uno.



- generalized forwarding: simple packet-handling rules
  - *Pattern:* match values in packet header fields
  - *Actions: for matched packet:* drop, forward, modify, matched packet or send matched packet to controller
  - *Priority*: disambiguate overlapping patterns
  - *Counters:* #bytes and #packets



16 bit ⇒ $2^{16}$ indirizzi

* : wildcard

1. src=1.2.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*, dest=3.4.*.* → forward(2)
3. src=10.1.2.3, dest=*.*.*.* → send to controller

# Examples

Destination-based forwarding:

| Switch Port | MAC src | MAC dst | Eth type | VLAN ID | IP Src | IP Dst | IP Prot | TCP sport | TCP dport | Action |
|---|---|---|---|---|---|---|---|---|---|---|
| * | * | * | * | * | * | 51.6.0.8 | * | * | * | port6 |

*IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6*

Firewall:

| Switch Port | MAC src | MAC dst | Eth type | VLAN ID | IP Src | IP Dst | IP Prot | TCP sport | TCP dport | Forward |
|---|---|---|---|---|---|---|---|---|---|---|
| * | * | * | * | * | * | * | * | * | 22 | drop |

*do not forward (block) all datagrams destined to TCP port 22*

| Switch Port | MAC src | MAC dst | Eth type | VLAN ID | IP Src | IP Dst | IP Prot | TCP sport | TCP dport | Forward |
|---|---|---|---|---|---|---|---|---|---|---|
| * | * | * | * | * | 128.119.1.1 | * | * | * | * | drop |

*do not forward (block) all datagrams sent by host 128.119.1.1*