VPN/tunnels (e altro...)

Otacon22

~~VPN~~/tunnels (e altro...)

Reti: episodio 2 La vendetta!

Giacomo

Netcat?

- È il comando base da bash per fare dei test di connessione/etc...

nc <hostname> <port>

Avvia una connessione TCP verso l'host e porta scelti e ci permette di inviare uno stream di caratteri.

- Con **-v** otteniamo lo stato della connessione
- Con **-n** evitiamo la risoluzione dell'RDNS
- Con **-l** possiamo avviare un server
- Con **-u** utilizziamo UDP invece di TCP

Vediamo su netkit due macchine che si parlano

Netcat?

- Inviare un file da un host ad un altro:
- Sull'host che invia:

  # nc -l 8000 < /tmp/file_da_inviare
- Sul secondo host:

  # nc host1 8000 > /tmp/file_ricevuto

Esempio GET da google

Tcpdump

tcpdump -i eth0 -n *“[espressione bpf]”*

Non risolvere i reverse DNS (rallenta l'output)

- Esempi di espressioni:
  - *tcp port 80*
  - *udp*
  - *icmp or udp*
  - *tcp port 80 and src 10.0.0.2*
  - *not (udp port 1900 and dst 10.0.0.1)*
  - *tcp[tcpflags] & (tcp-syn|tcp-fin) != 0 and not src and dst net localnet*
  - *tcp and src net 131.175*
- *-w crea un file pcap con un log esatto del traffico di rete:*

tcpdump -i eth0 -n -w /tmp/very_secret_log.pcap

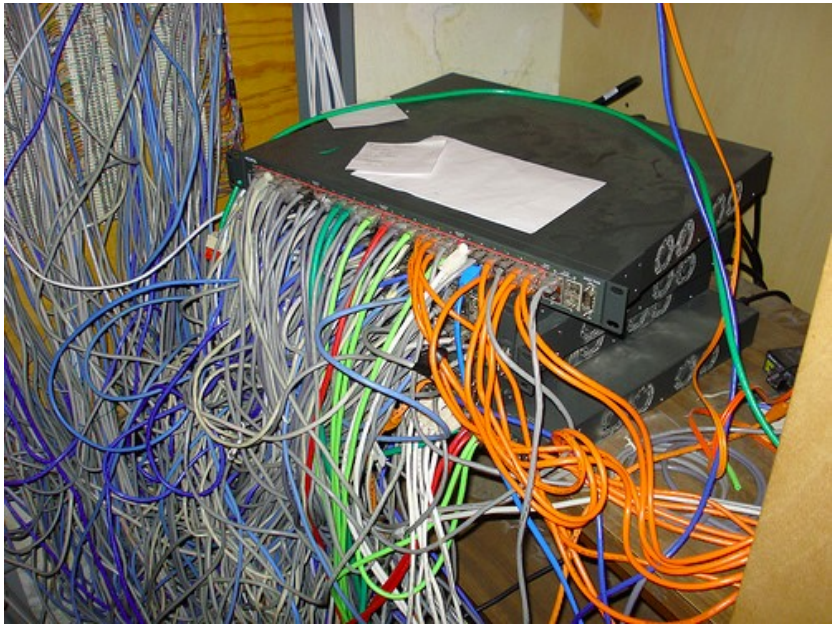
ngrep

- Network grep

ngrep -d eth0 -W byline “<match expression>” “<bpf filter>”

- Possiamo filtrare il contenuto di tutti i pacchetti che corrispondono ad una espressione regolare
- Possiamo killare connessioni tcp...
- Attenzione alle stringhe spezzate tra due pacchetti

Esempi netkit



- Fa comodo per qualche simulazione senza stare a tirar fuori troppi cavi
- Un tool piuttosto comodo è **netkit**, che potete scaricare da http://netkit.org

Secure Shell (**SSH**)

SSH

- Ci permette di accedere alla shell di un server remota, in modo autenticato e sicuro.

ssh username@host

- Normalmente possiamo fare login con password
- Al primo login dobbiamo salvare il fingerprint del server per verificarne l'identità
- Se ci sono problemi dovremo cancellare il fingerprint salvato alla prima connessione (in ~/.ssh/known_hosts)

SSH: utilizzare chiavi

- Per accedere in modo più sicuro possiamo generare delle chiavi RSA:
  - Chiave pubblica (ad es: ~/.ssh/id_rsa.pub)
    - È quella che possiamo dare in giro, la mettiamo all'interno del server su cui vogliamo autenticarci
  - Chiave privata (ad es: ~/.ssh/id_rsa)
    - È quella che teniamo segreta, la usiamo nel momento in cui vogliamo entrare in un server su cui abbiamo la chiave pubblica
- La chiave ssh può essere senza password (basta avere il file per usarla), oppure protetta da password (serve una password per poterla utilizzare)

SSH: utilizzare chiavi

- Per creare la nostra coppia di chiavi:

  ssh-keygen

- Se lasciamo le nostre chiavi in ~/.ssh/ saranno utilizzate di default per tentare l'accesso al server, prima di rinunciare e usare la password
- Per specificare una chiave specifica da usare in una connessione si usa **-i** (identity):

  ssh -i /home/antani/key root@remotehost

- Nota: se avete troppe chiavi in ~/.ssh/ dovete usare -o PubkeyAuthentication=no

SSH: utilizzare chiavi

- Per poter accedere ad un server utilizzando la nostra chiave privata, dovremo copiare la nostra chiave pubblica nel file **~/.ssh/authorized_keys** dell'host remoto
- Attenzione ai permessi!
  - Se non riuscite a loggarvi date un'occhiata a **/var/log/auth.log**
  - chown -R utente:utente ~/.ssh/
  - chmod -R 700 ~/.ssh/

SSH

- Con **-v** abilitate il verbose e potete subito capire se ci sono problemi con le chiavi che sta provando
- Se vi serve utilizzare SSH su una porta diversa dalla TCP 22, **-p <porta>** farà al caso vostro

  ssh -p 1234 user@host -v
- Con **-C** possiamo anche abilitare la compressione di tutti i dati inviati in rete!
- Possiamo verificare i fingerprint iniziali con i DNS! Per approfondire: http://goo.gl/ClE7l

SCP

scp /tmp/file1 user@hostname:/mnt/

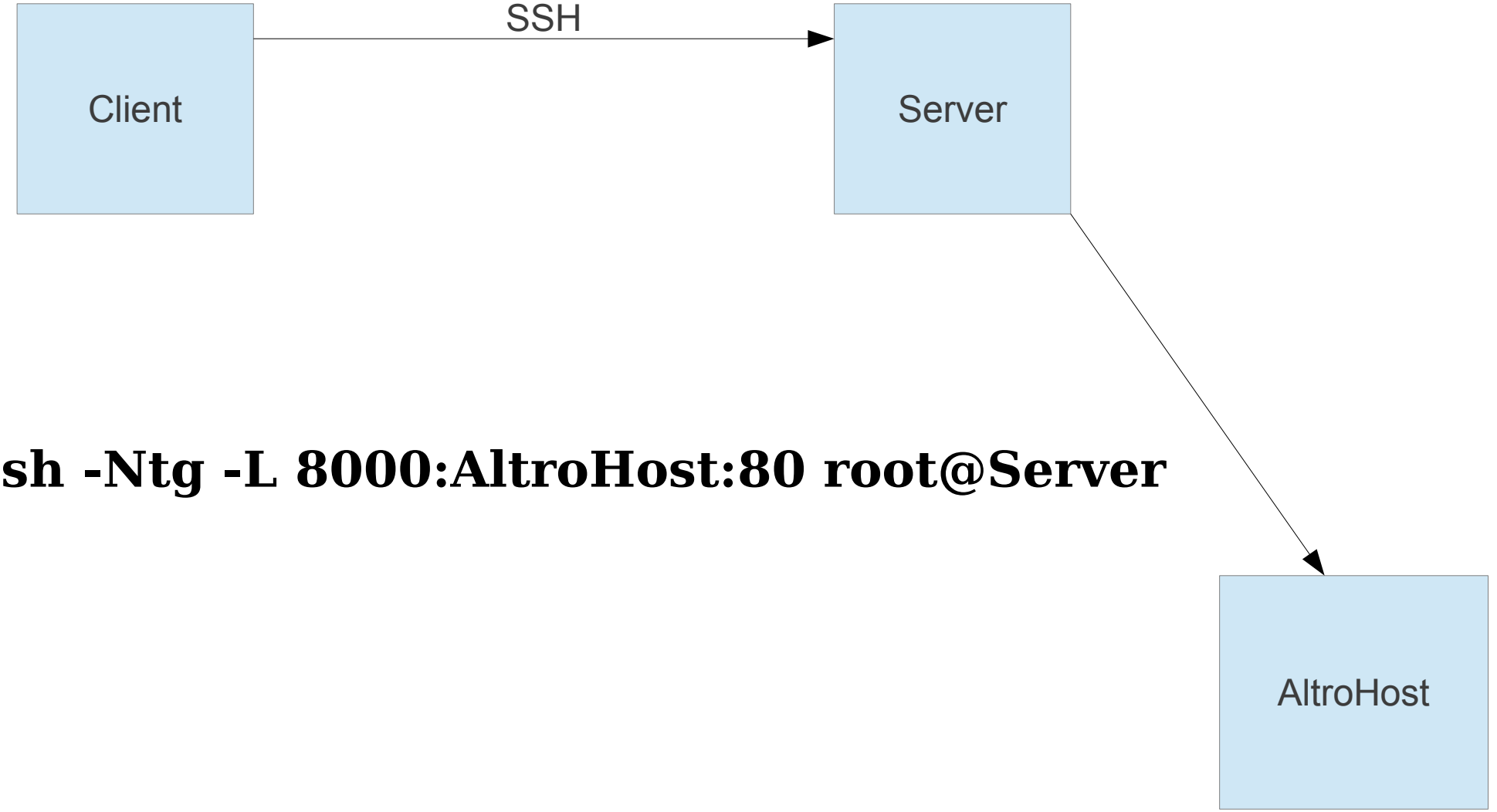
scp user@hostname:/mnt/file1 /tmp/

scp -r user@hostname:/mnt/ /tmp/

SSH tunneling

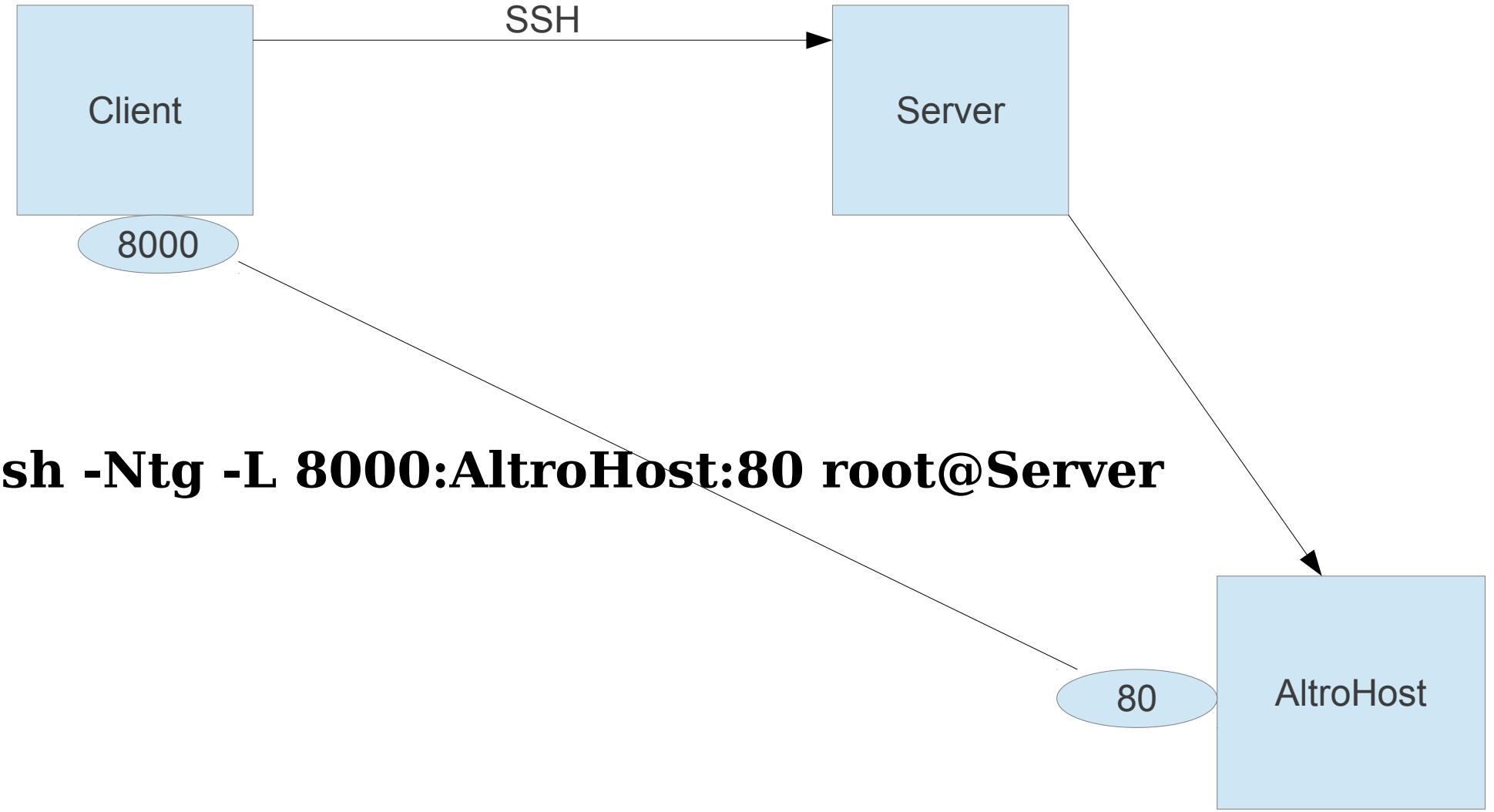
- SSH non è **solo** un programma per accedere ad una shell remota. Ha una serie di funzioni aggiuntive che sfruttano il “canale cifrato” che viene creato in ssh tra il client e il server.
- Tra queste c'è SSH tunneling.
- Vediamo un esempio per capire a cosa serve

SSH tunneling



ssh -Ntg -L 8000:AltroHost:80 root@Server

SSH tunneling



ssh -Ntg -L 8000:AltroHost:80 root@Server

SSH tunneling

- Spesso è comodo se abbiamo un database MySQL (in ascolto su TCP 3306) e vogliamo accederci da remoto:
  - senza “esporlo” su Internet
  - Accedendoci “fingendoci” per il server (quindi 127.0.0.1)

ssh -Ntg -L 3306:localhost:3306 root@server

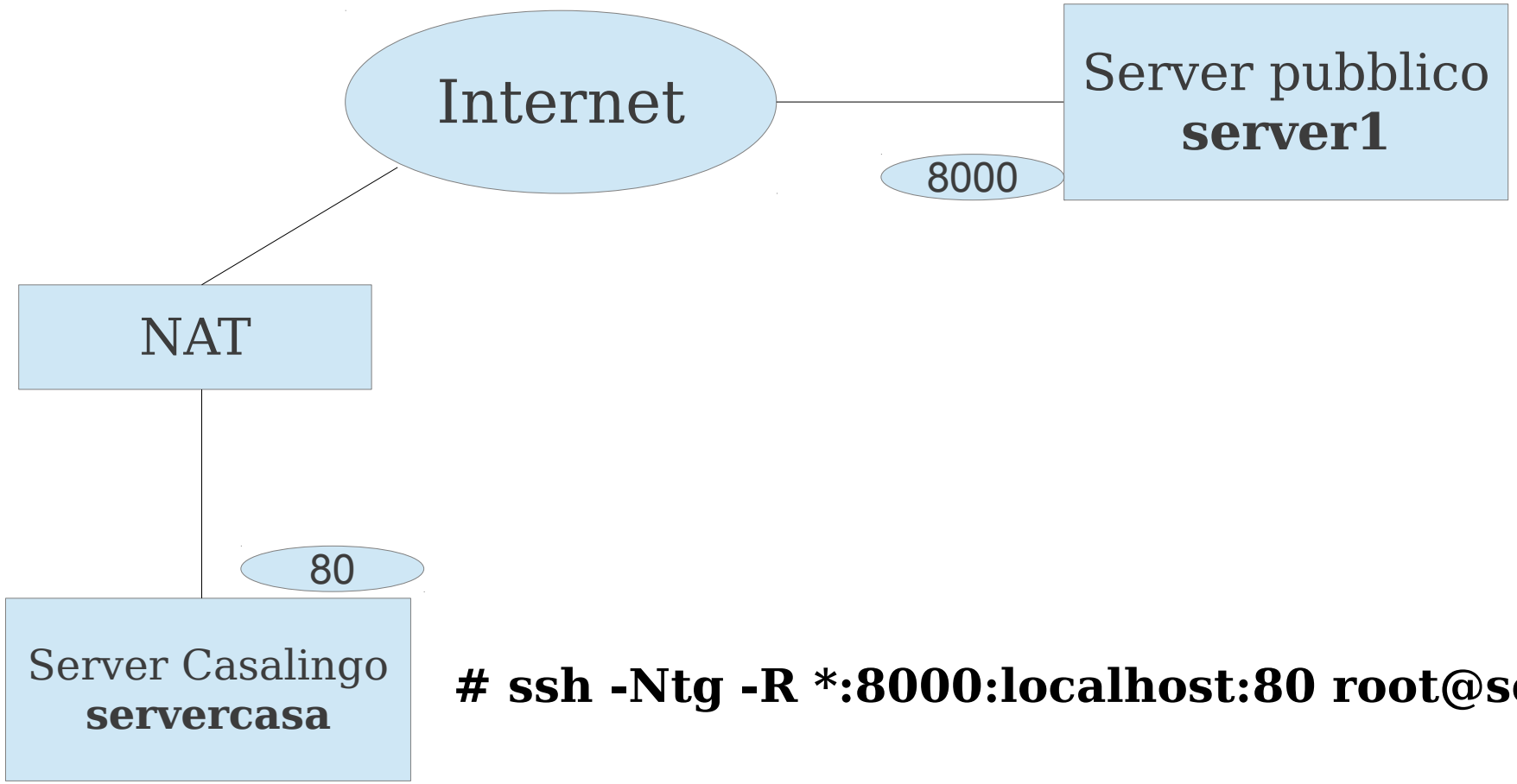
SSH reverse tunneling

- Funziona al contrario: diciamo al server “se ti arriva una connessione TCP su questa porta A, rigiramela a me. Poi io inoltro la connessione all'host2 sulla porta B.

ssh -Ntg -R *:8000:192.168.0.1:80 user@server

SSH reverse tunneling

- Esempio: ho un server con IP pubblico e ssh, voglio esporre su internet il mio server web di casa dietro NAT.



 # ssh -Ntg -R *:8000:localhost:80 root@server1

SSH reverse tunneling

- Ammettiamo che vogliamo dare un utente per accedere in ssh ad un amico/collega. Non vogliamo che possa usare il server ssh per fare tunneling/reverse tunneling.
- Si può risolvere modificando la config di ssh:

  /etc/ssh/sshd_config
- E aggiungendo opzioni specifiche per l'utente

- P.S: SSH tunnel/reverse tunnel non funziona con UDP, sorry

SSH SOCKS proxy

- Non abbiamo voglia di tirare su una VPN, ci serve qualcosa di veloce per uscire su Internet spacciandoci per il server ssh a cui ci colleghiamo (qualcuno ha detto vedere siti bloccati dalla rete dove sono?)

  ssh -D 9053 user@server

- E impostiamo sul brower un proxy SOCKS4 o SOCKS5
- **Fine, non serve altro!**

OpenVPN

- La guida sul mio blog come reference:

http://goo.gl/pMXkW

OpenVPN

- Se possibile evitate di usare TCP per la VPN.
- Se usate TCP trasformate il traffico real-time/datagram (UDP) in traffico reliable, perdendo tutti i suoi vantaggi

Server

```
port 1194
proto udp
dev tun
ca /etc/openvpn/keys/ca.crt
cert /etc/openvpn/keys/VPN_Server.crt
key /etc/openvpn/keys/VPN_Server.key
dh /etc/openvpn/keys/dh2048.pem
tls-auth /etc/openvpn/keys/ta.key 0
server 10.0.0.0 255.255.255.0
max-clients 10
ifconfig-pool-persist
/etc/openvpn/ipp.txt
comp-lzo
keepalive 10 60
client-to-client
log-append /var/log/openvpn.log
status /var/log/openvpn-status.log
verb 4
```

Client

```
client
dev tun
proto udp
remote SERVER_HOSTNAME 1194
resolv-retry infinite
nobind
persist-key
persist-tun
ca /etc/openvpn/ca.crt
cert /etc/openvpn/client1.crt
key /etc/openvpn/client1.key
tls-auth /etc/openvpn/ta.key 1
keepalive 8 40
comp-lzo
verb 4
ns-cert-type server
```

OpenVPN

- Differenza tun/tap
- Due tipi generali di VPN: collegamento siti remoti, routing su internet geolocalizzato
-

VPN con SSH

- Si può fare una semplice VPN anche con SSH!

  ssh -w 0:0 user@server

- Crea una interfaccia tun0 sul nostro sistema e sul server remoto
- Reference:

  https://help.ubuntu.com/community/SSH_VPN

IPsec

- http://ipsec-howto.org/
- Lavora a livello 3 invece di livello 4
- Se usate racoon:

  /etc/racoon/racoon.conf

  setkey -f config.conf

  racoon -f /etc/racoon/racoon.conf
- Tutta la config si basa sugli indirizzi IP sorgente e destinazione per decidere cosa cifrare

Tunnel IP-IP

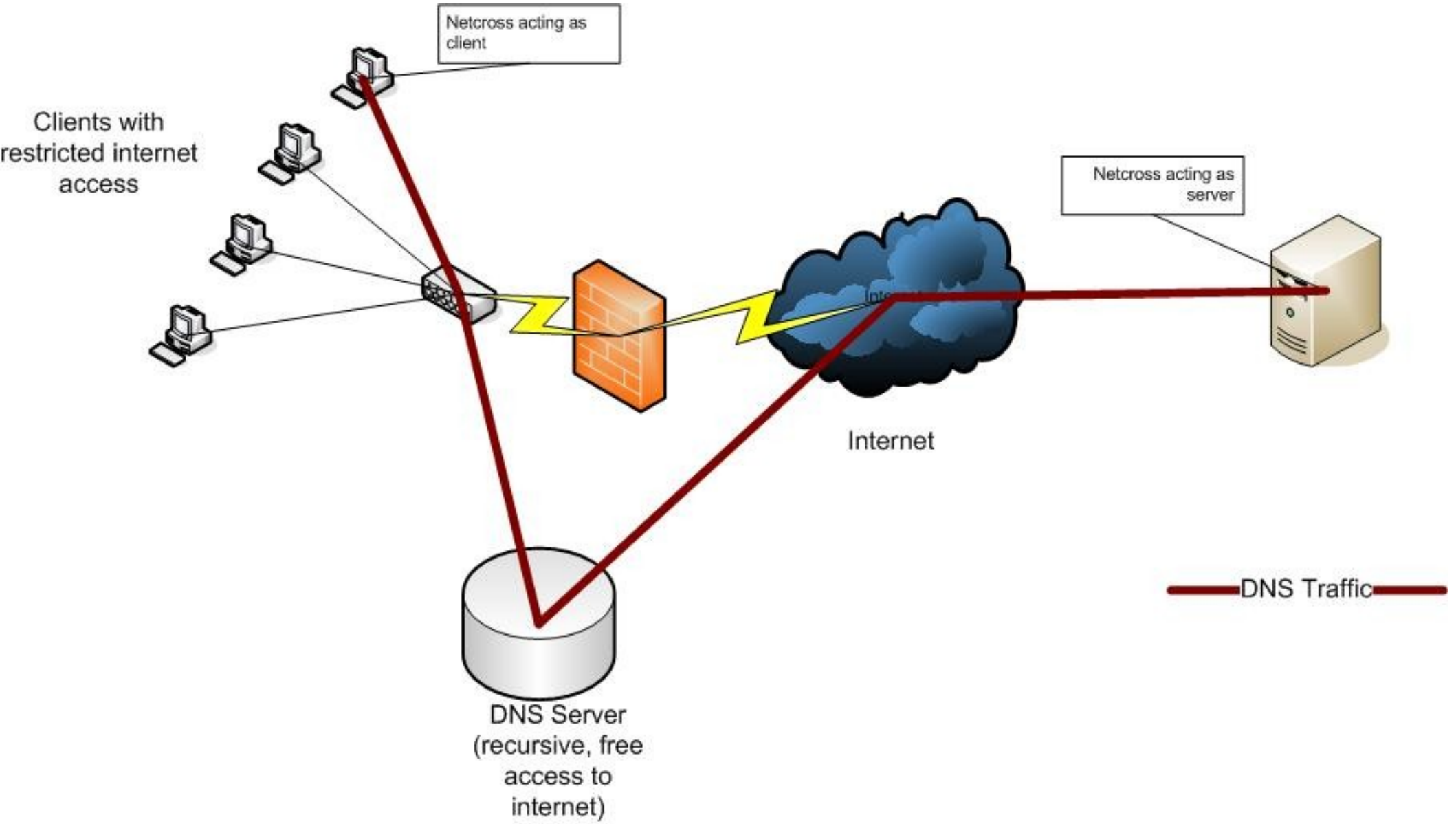
- Router rete A:
  - ifconfig tunl0 10.0.1.1 pointopoint 172.19.20.21
  - route add -net 10.0.2.0 netmask 255.255.255.0 dev tunl0

- Router rete B:
  - ifconfig tunl0 10.0.2.1 pointopoint 172.16.17.18
  - route add -net 10.0.1.0 netmask 255.255.255.0 dev tunl0

Iodine

- Tenta prima di collegarsi alla UDP 53 e fare tunneling su quella.
- Se non riesce, tenta di fare tunneling su vari tipi di record DNS
  - D:Chi è abcwieuhw.iodine.domain.com?
  - R:4.5.6.7
- Tenta con i record NULL,TXT,A,AAA....
- Calcola in automatico la dimensione ottimale delle richieste
- Abbiamo un lag pauroso

Iodine


Netcross acting as client
Clients with restricted internet access
Netcross acting as server
Internet
DNS Server (recursive, free access to internet)
DNS Traffic

Iodine

- Ammettiamo di possede **example.com**
- Sul server DNS:

  example.com A 1.2.3.4

  iodinedomain NS example.com

  - In questo modo:

    (qualunquecosa).iodinedomain.example.com

    Genererà una richiesta che sarà inoltrata verso il server DNS example.com

Iodine

- Sul server:

  iodined -P password 172.16.0.1

- Sul client:

  iodine -f iodinedomain.example.com

- Vengono create delle interfacce dns0 (di tipo tun) su entrambi gli host.

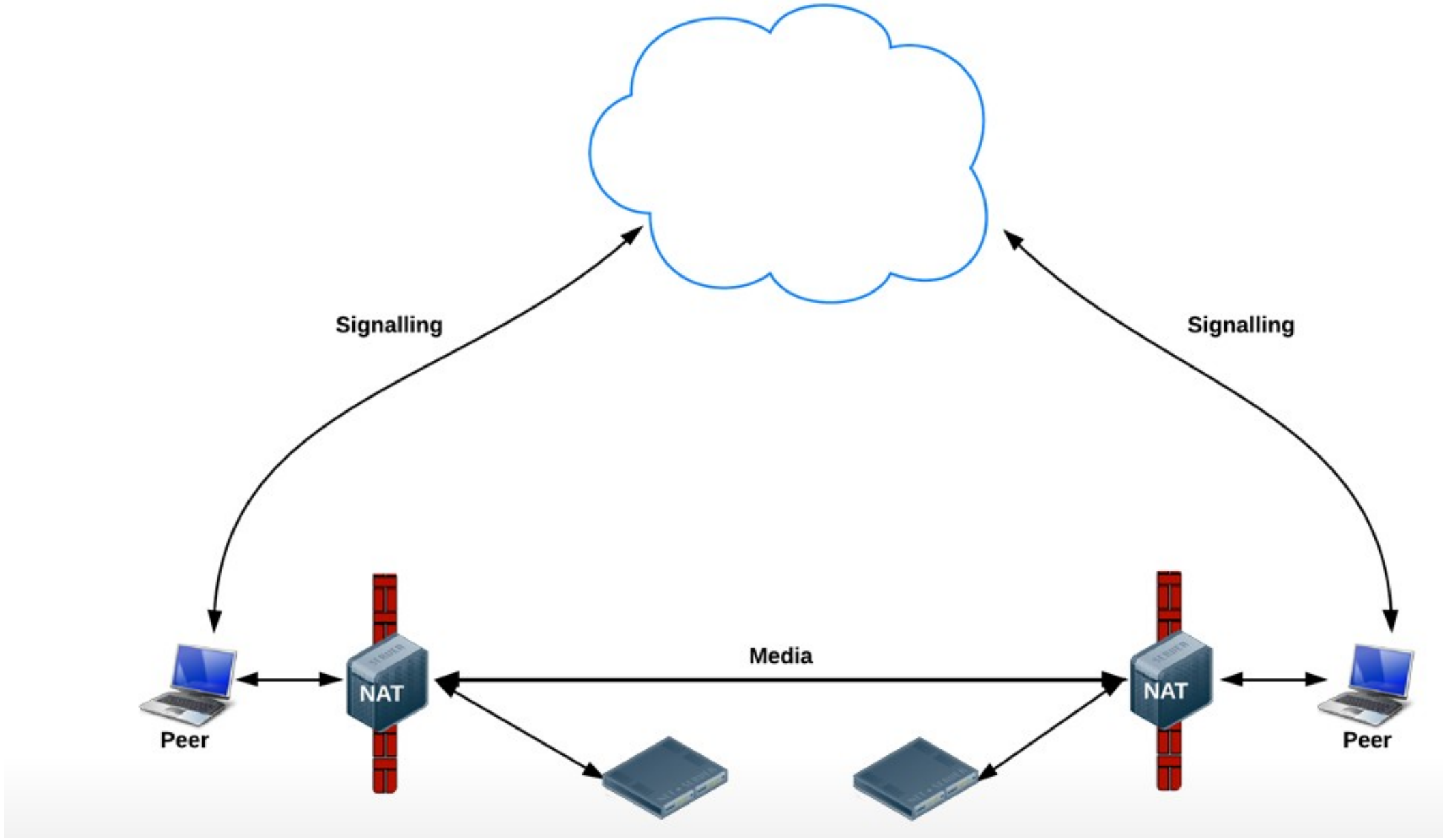
ptunnel

- Se volete fare ICMP tunneling...

ICEtunnel

- https://github.com/Otacon22/icetunnel
- Non è un vero e proprio tunnel (non crea una interfaccia virtuale)
- WebRTC funziona in modo molto simile (ma è anche cifrato!)

STUN



Domande?

Grazie!

http://www.otacon22.it/