NAT issues, contd. - Server behind NAT

“Port forwarding” or “port mapping”

- Statically configure public port # on NAT device
- To forward to particular internal ⟨ip, port⟩
- Limitation:
  - cannot support two internal servers on port 80 with only 1 public IP address + port 80.

You basically map a specific port for that server (usually 80) on the NAT.

NAT editor, SPNAT

- NAT editor:
  - Some application-layer protocol carry lower-layer information
    - E.g., ftp traditionally uses multiple connections: control and data
    - Data in control connection communicates info such as port # about data connection
  - A “NAT editor” rewrites application-layer data as well.

- SPNAT = “Service-Provider NAT” = NAT by ISP to mitigate address-depletion
  - Reduces customer-control over filtering policy
  - E.g., cannot run a server any longer as a customer of ISP

MISSED THIS, GOOGLE IT

More broadly - STUN and TURN

- STUN = Session Traversal Utilities for NAT
  - RFC 5389
- STUN server helps a client that is behind NAT.
  - "...a protocol that serves as a tool for other protocols in dealing with NAT traversal..."
  - "...used by an endpoint to determine the IP address and port allocated to it by a NAT."
  - "...used to check connectivity between two endpoints, and as a keep-alive protocol to maintain NAT bindings."
  - "...works with many existing NATs, and does not require any special behavior from them."

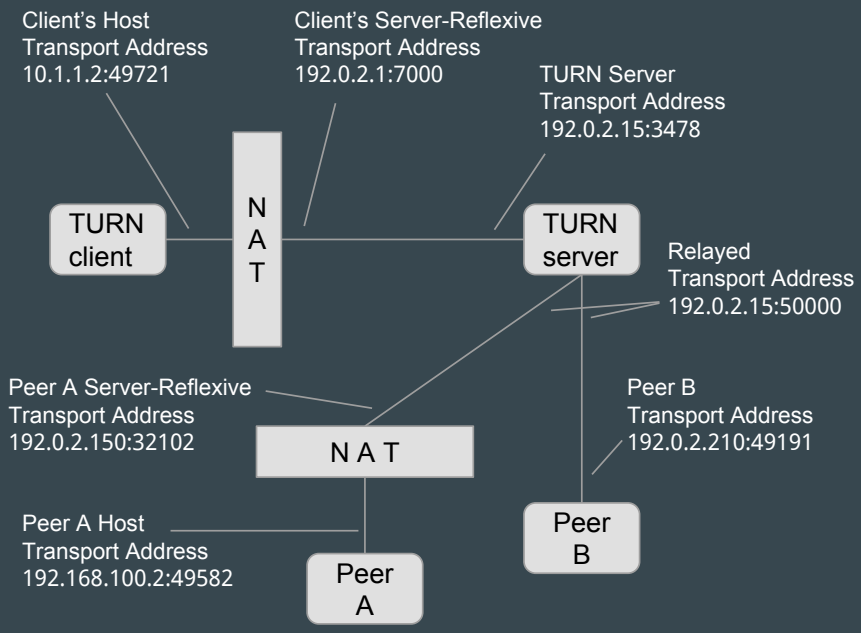
Basically you have a server that sits outside and helps clients deal with NAT problems and such. Helps to determin the IP and port is allocated. Can also be used as a heartbeat type thing.

TURN

- Traversal Using Relays around NAT
- RFC 5766
- “Last resort for communication around uncooperative NATs.”
  - “Uncooperative NATs” are also called “Bad NATs.”
  - E.g., one that performs address- and port-dependent mapping.

Traversal Using Relays around NAT. This is kind of an alternative to STUM. It is meant to help you deal with troublesome NATS that perform address/port dependent mapping. This makes it hard to work with.

TURN, contd.



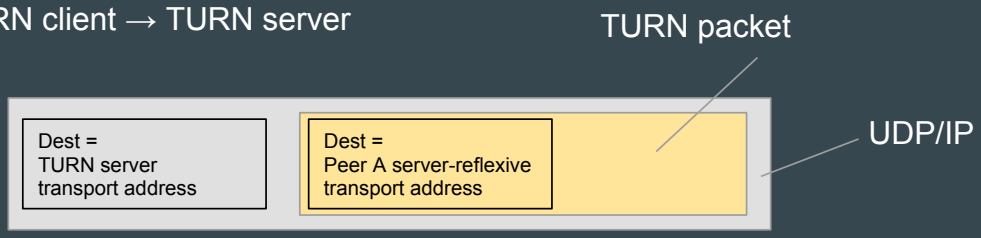
- Focus in figure is on TURN client to the far left, 10.1.1.2:49721
- It has reserved a 'relayed transport address,' at the TURN server, 192.0.2.15:50000. Other peers send UDP packets to this as destination so it is forwarded to the client.
- Client sends TURN packets to TURN server at the 'TURN server transport address,' 192.0.2.15:3478. Server extracts and sends UDP packets to peers.
- Client addresses another peer using that peer's 'server-reflexive transport address.' E. g., 192.0.2.150:32102

Warning: Fig. 7-11 in your textbook appears to be erroneous.

There are 2 NAT devices and a TURN client. There is a TURN server that is helping route between two peers. The TURN client has a private ip address that is NATed by the device, The client reserves an allocation at the TURN server for a relayed transport address. The TURN client is enabling other peers to send it packets on this new public address. This address is owned by the server and allocated to the client and packets will be forwarded there. If the TURN client wants to send packets to a peer it encapsulates the packets with a server transport address and sends them to the server for routing. The address of the peer the client is sending to is called the server reflexive transport address. This address is the address that the server would see (if the device is NATed the NATed address is used, else its standard ip address).

TURN + UDP

- TURN client → TURN server

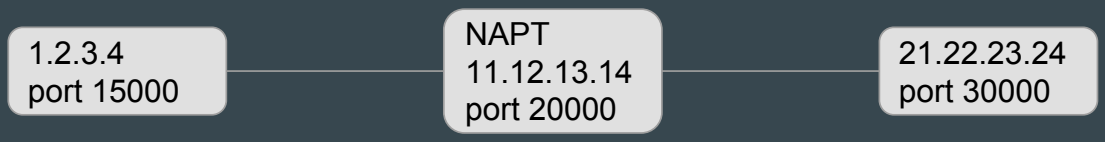


- TURN server → peer

The destination on the packet should first be the address and port of the TURN server. Once it gets there it removes a layer and looks at the TURN packet with now has its destination as the server reflexive address of the peer it actually wants to go to. The source address is the relayed transport address of the server.

Quiz 1

- On translation behaviour. Suppose:



1. Does a packet ⟨src-ip: 31.32.33.34, src-port 30000, dst-ip: 11.12.13.14, dst-port 20000⟩ on the Internet, reach the host 1.2.3.4?
2. Does a packet ⟨src-ip: 21.22.23.24, src-port 40000, dst-ip: 11.12.13.14, dst-port 20000⟩ on the Internet, reach the host 1.2.3.4?

Answers:

1. *Yes, under endpoint-independent. No, under address- and/or port-dependent.*
2. *Yes, under endpoint-independent and address-dependent. No, under address- and port-dependent.*

This shows the reason that endpoint independent behavior is favorable so that people can send data to your server easily.

Quiz 2

- Suppose one of those packets does indeed reach the host 1.2.3.4. What is the host's behaviour that we expect?
  - After translation in the reverse direction, we expect the packet that arrives at 1.2.3.4 to be:

    1. ⟨src-ip: 31.32.33.34, src-port 30000, dst-ip: 1.2.3.4, dst-port 15000⟩
    2. ⟨src-ip: 21.22.23.24, src-port 40000, dst-ip: 1.2.3.4, dst-port 15000⟩

Answer:

- *If UDP, we expect both to be delivered to the application.*
  - *Session is associated with 2-tuple, ⟨dst-ip, dst-port⟩, only.*
- *If TCP, we expect neither to be delivered to the application.*
  - *Connection is associated with 4-tuple.*