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.

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

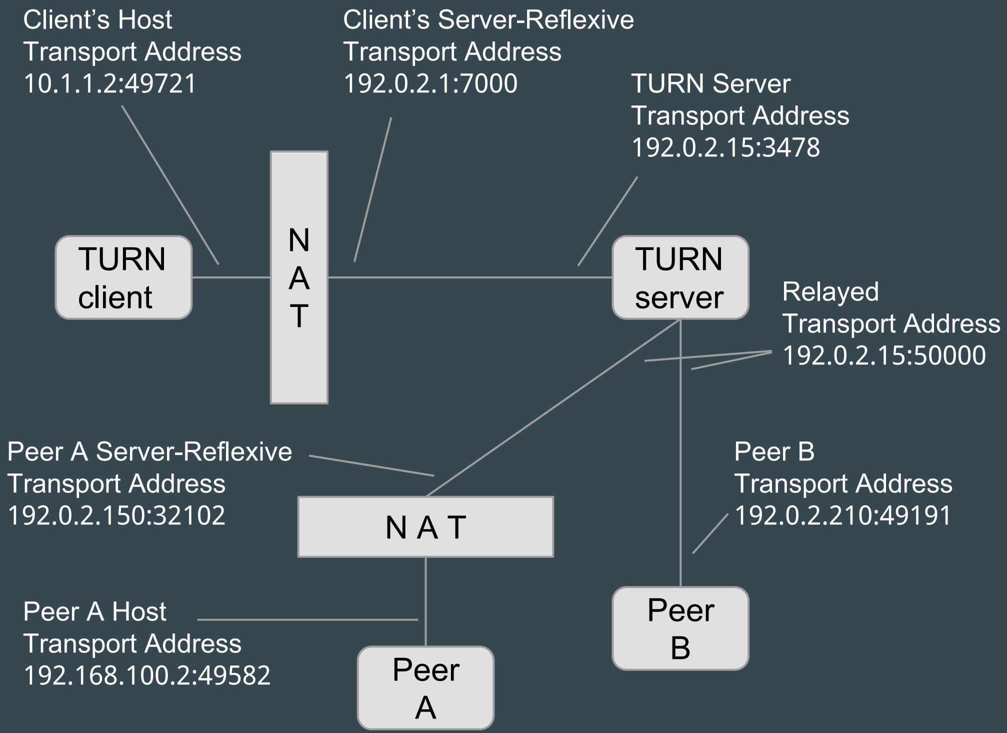
More broadly - STUN and TURN

- 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."

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.

TURN, contd.

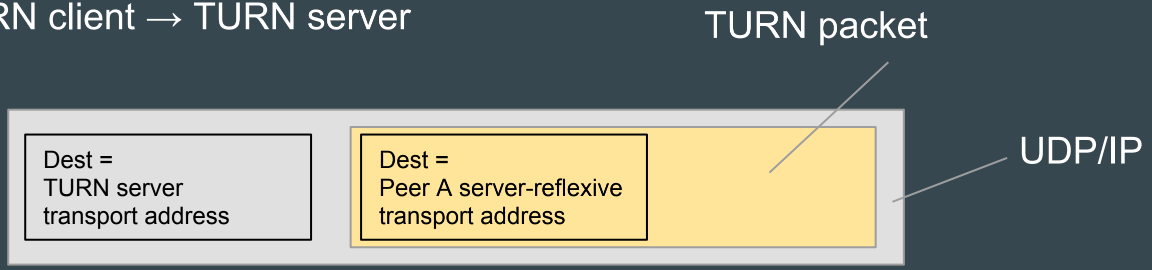


- Focus in figure is on TURN client to the far left, 10.1.1.2:49721
- It has reserved a 'relayed transport address,' at the TURN server, 192.0.2.15:50000. Other peers send UDP packets to this as destination so it is forwarded to the client.
- Client sends TURN packets to TURN server at the 'TURN server 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.

TURN + UDP

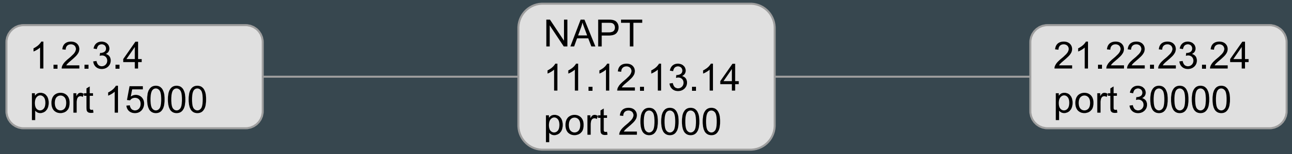
- TURN client → TURN server



- TURN server → peer

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.*

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.*

Quiz 2, contd.

- Security issue: Internet attacker packet's identity masked by NAT rewriting.
- But, endpoint-independent mapping allows, for example, UDP server to run inside internal network without need for, for example, TURN.