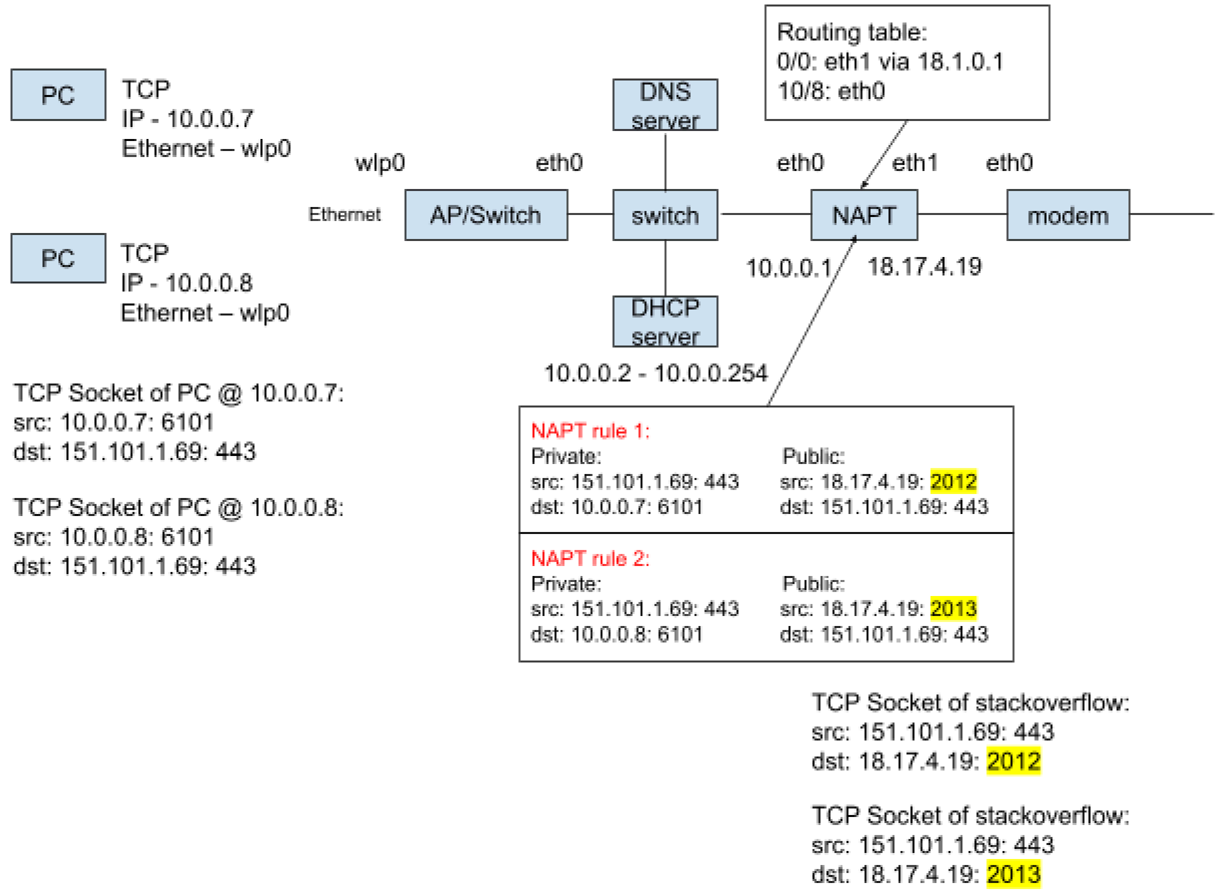
Level 8: NAPT



Routing table:
0/0: eth1 via 18.1.0.1
10/8: eth0

TCP Socket of PC @ 10.0.0.7:
src: 10.0.0.7: 6101
dst: 151.101.1.69: 443

TCP Socket of PC @ 10.0.0.8:
src: 10.0.0.8: 6101
dst: 151.101.1.69: 443

NAPT rule 1:

Private:	Public:
src: 151.101.1.69: 443	src: 18.17.4.19: 2012
dst: 10.0.0.7: 6101	dst: 151.101.1.69: 443

NAPT rule 2:

Private:	Public:
src: 151.101.1.69: 443	src: 18.17.4.19: 2013
dst: 10.0.0.8: 6101	dst: 151.101.1.69: 443

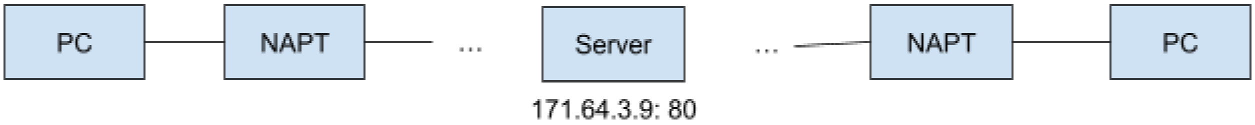
TCP Socket of stackoverflow:
src: 151.101.1.69: 443
dst: 18.17.4.19: 2012

TCP Socket of stackoverflow:
src: 151.101.1.69: 443
dst: 18.17.4.19: 2013

- If the PC @ 10.0.0.7 wants to start a connection with stackoverflow @ 151.101.1.69 either through a proxy or a transparent proxy or a NAPT translator.
    - The number of TCP connections between any PC on the subnet to stackoverflow is equal to the number of possible port numbers (65536)
    - Each new TCP connection between a PC on the subnet to a public IP address adds a new NAPT rule
    - A NAPT rule is garbage-collected either when a TCP connection is closed or the rule has not been used for a while
- However, what happens if stackoverflow @ 151.101.1.69 wants to start a connection with the PC @ 10.0.0.7? Or how to allow PC @ 10.0.0.7 to host a file server?
    - The dumbest way: have file servers on the public internet that are not behind NAPT, and upload any files to those public servers for sharing

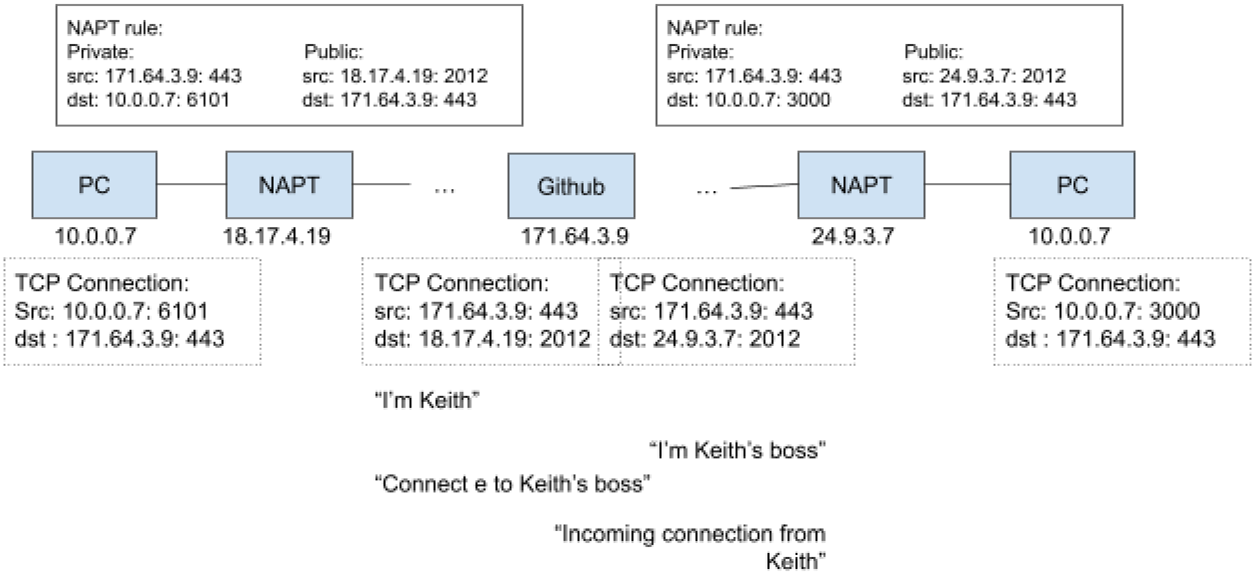
Level 9a: P2P networking via public server

- Use one public server to hold the files between PCs behind NAPTs



-
- How to achieve this without having the server to hold on to some files?

Level 9b: P2P networking via public proxy/relay/ TURN (Traversal Using Relays around NAPT)



NAPT rule:
Private:
src: 171.64.3.9: 443
dst: 10.0.0.7: 6101
Public:
src: 18.17.4.19: 2012
dst: 171.64.3.9: 443

NAPT rule:
Private:
src: 171.64.3.9: 443
dst: 10.0.0.7: 3000
Public:
src: 24.9.3.7: 2012
dst: 171.64.3.9: 443

TCP Connection:
Src: 10.0.0.7: 6101
dst : 171.64.3.9: 443

TCP Connection:
src: 171.64.3.9: 443
dst: 18.17.4.19: 2012

TCP Connection:
src: 171.64.3.9: 443
dst: 24.9.3.7: 2012

TCP Connection:
Src: 10.0.0.7: 3000
dst : 171.64.3.9: 443

"I'm Keith"

"I'm Keith's boss"

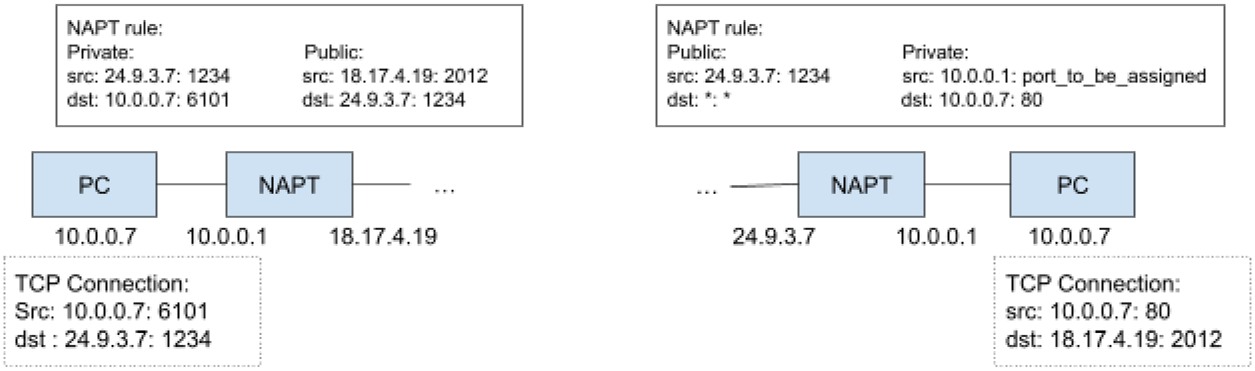
"Connect e to Keith's boss"

"Incoming connection from
Keith"

-
- What could we do if we don't want to connect through any kind of relay server in between?

Level 9c: P2P networking via explicit NAPT rules (port forwarding)

- Why can't two PCs talk to each other when they are both behind NAPT?
    - Their IP addresses are not meaningful on the public Internet
- Why can't one PC connect to the NAPT on the other side?



NAPT rule:
Private:
src: 24.9.3.7: 1234
dst: 10.0.0.7: 6101
Public:
src: 18.17.4.19: 2012
dst: 24.9.3.7: 1234

NAPT rule:
Public:
src: 24.9.3.7: 1234
dst: *: *
Private:
src: 10.0.0.1: port_to_be_assigned
dst: 10.0.0.7: 80

TCP Connection:
Src: 10.0.0.7: 6101
dst : 24.9.3.7: 1234

TCP Connection:
src: 10.0.0.7: 80
dst: 18.17.4.19: 2012

-

- If we add the NAPT rule to the NAPT @ 24.9.3.7 before the TCP Connection is started, then a direct TCP Connection can be established between the two PC behind NAPTs.

—---------------------------The following content was not part of the lecture—-----------------------------
There was some confusions around how the private src and public dst are set in a NAPT rule, and this is decided by the NAT implementations defined here: https://www.rfc-editor.org/rfc/rfc3489#section-5.

Say PC @ 10.0.0.7: 6101 starts a TCP connection to PC @ 24.9.3.7:1234 by sending a packet, the rule established would be:

- Full Cone

private:	public :
src: * : *	src: 18.17.4.19: 2012
dst: 10.0.0.7 : 6101	dst: * : *

- Restricted Cone

private:	public :
src: * : *	src: 18.17.4.19: 2012
dst: 10.0.0.7 : 6101	dst: 24.9.3.7: *

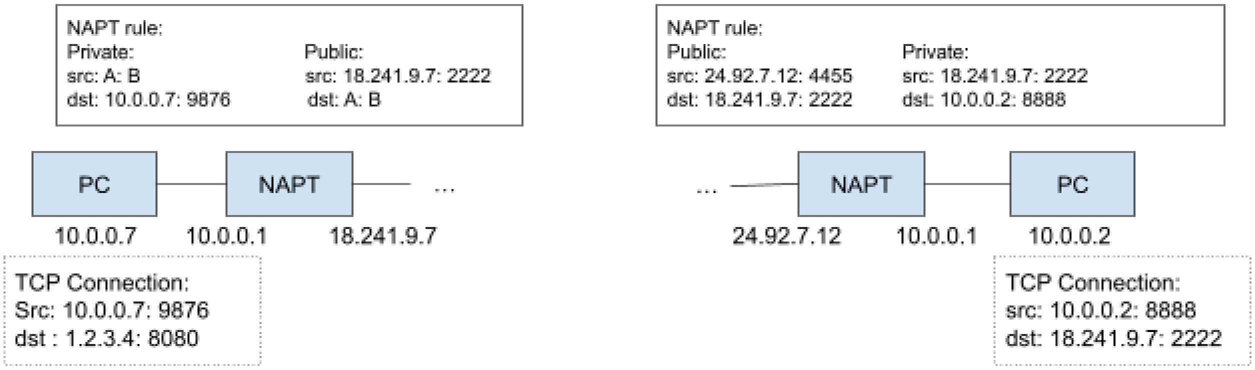
- Port Restricted Cone

private:	public :
src: * : *	src: 18.17.4.19: 2012
dst: 10.0.0.7 : 6101	dst: 24.9.3.7: 1234

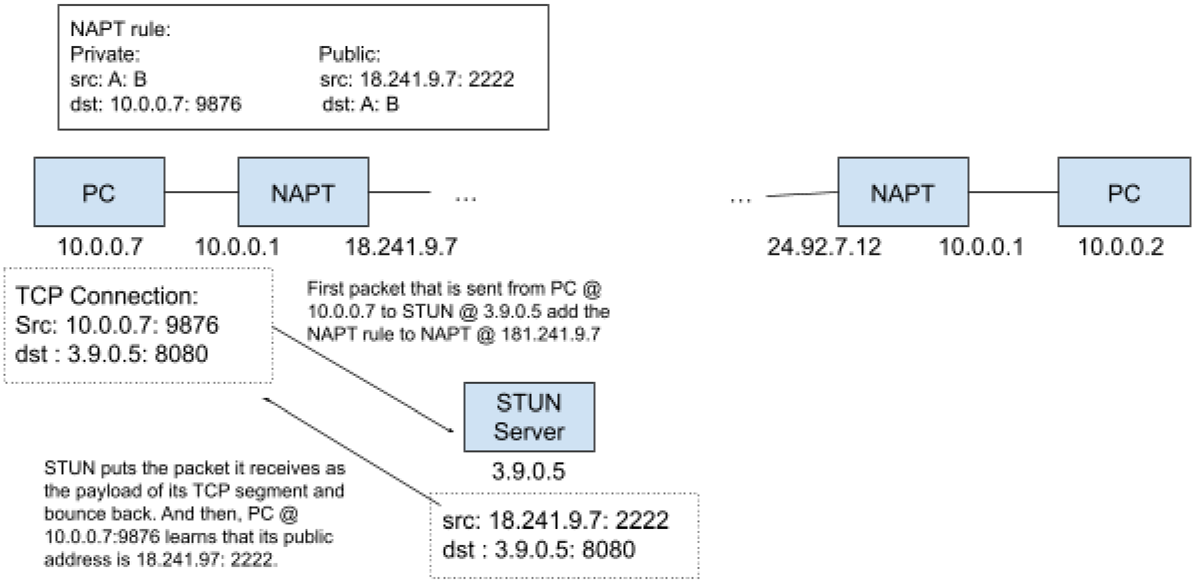
- Symmetric

private:	public :
src: 24.9.3.7: 1234	src: 18.17.4.19: 2012
dst: 10.0.0.7: 6101	dst: 24.9.3.7: 1234

Level 9d: P2P networking via NAT traversal



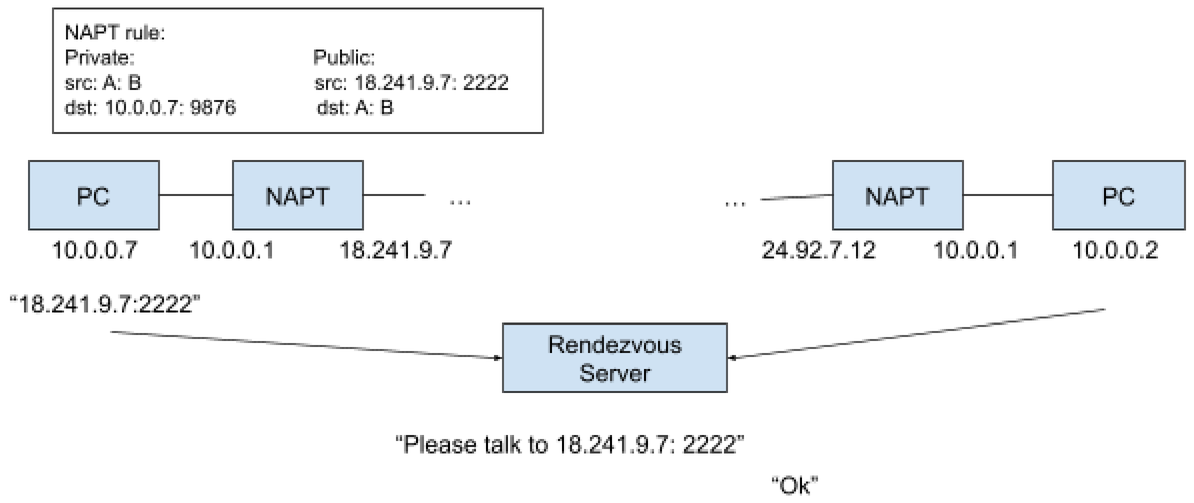
- "cone" NAPT rule
    - Any connection that goes to 18.241.9.7: 2222 would be reroute to 10.0.0.7:9876
- What would be needed to make this TCPConnection happen?
    - Step 1: PC @ 10.0.0.7 needs to know its public IP address
        - STUN Server @ 3.9.0.5 (STUN Servers are stateless)



First packet that is sent from PC @ 10.0.0.7 to STUN @ 3.9.0.5 add the NAPT rule to NAPT @ 181.241.9.7

STUN puts the packet it receives as the payload of its TCP segment and bounce back. And then, PC @ 10.0.0.7:9876 learns that its public address is 18.241.97: 2222.

    - Step 2: PC @ 10.0.0.7 wants to tell its peer about its public address: Rendezvous server
        - A rendezvous server is similar to a chat room, and the rendezvous server itself doesn't have to have any persistent mapping between user names and public addresses
        - When one user is trying to talk to another users, the rendezvous server checks whether the other party is logged in on the server, and if true send the message
        - Rendezvous servers are often run by applications (e.g. Minecraft, or Bittorrent) since rendezvous servers are cheaper than TURN servers, and TURN servers would be needed if P2P connections cannot be established


NAPT rule:
Private:
src: A: B
dst: 10.0.0.7: 9876
Public:
src: 18.241.9.7: 2222
dst: A: B
PC
NAPT
…
…
NAPT
PC
10.0.0.7
10.0.0.1
18.241.9.7
24.92.7.12
10.0.0.1
10.0.0.2
“18.241.9.7:2222”
Rendezvous Server
“Please talk to 18.241.9.7: 2222”
“Ok”