ICE

How to find your way through the Internet

2023 GStreamer conference

Matthew Waters

25 September 2023

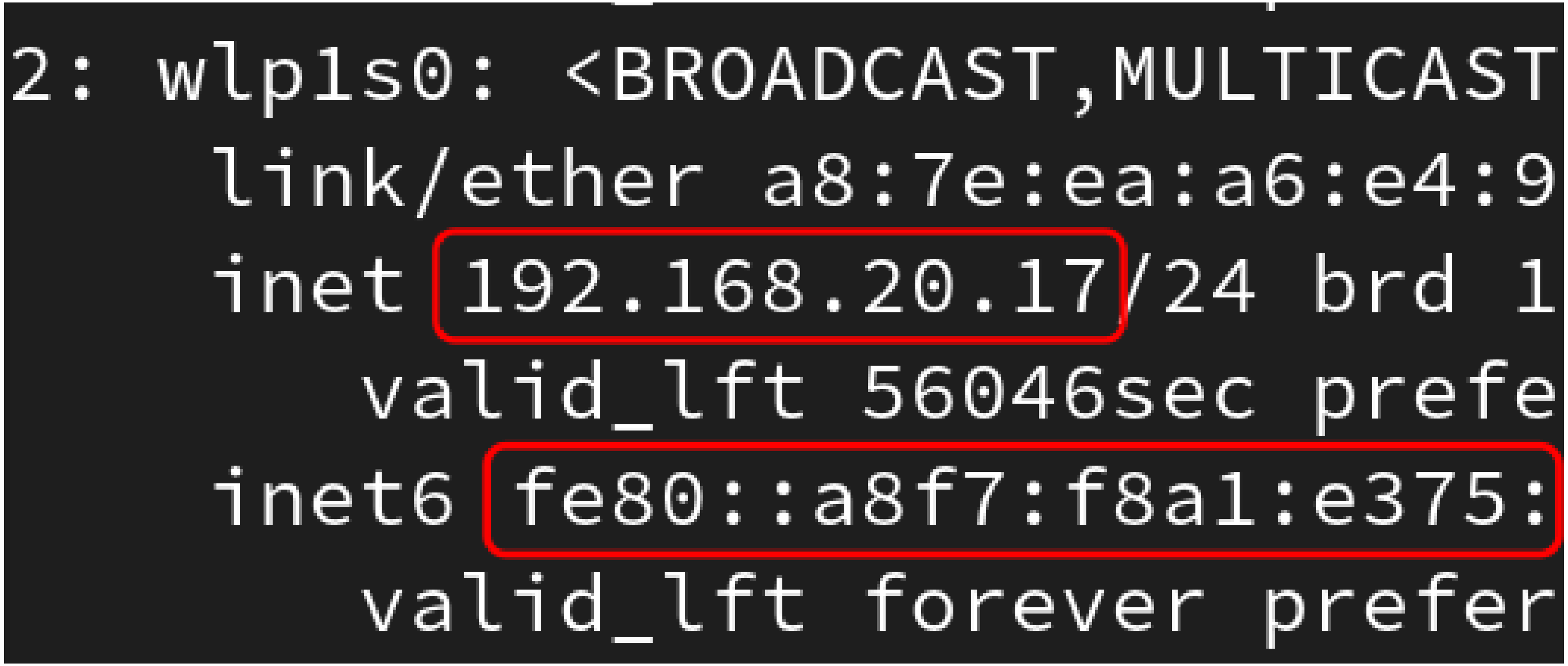
Centricular

Goal

Create a bidirectional communication channel between two internet connected devices as efficiently and fast as reasonable

The Internet Mail System

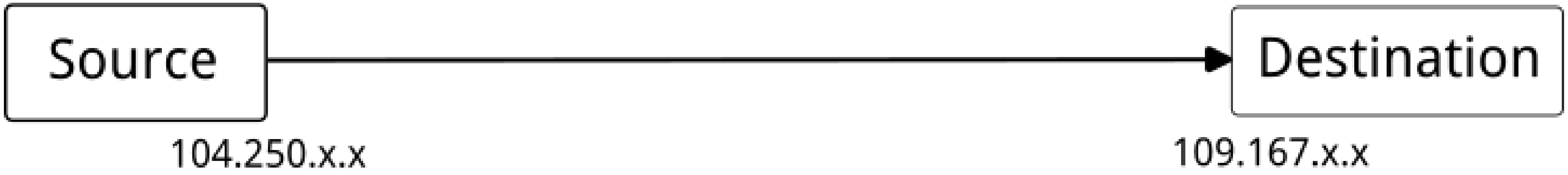
Addresses



```
2: wlp1s0: <BROADCAST,MULTICAST
    link/ether a8:7e:ea:a6:e4:9
    inet 192.168.20.17/24 brd 1
       valid_lft 56046sec prefe
    inet6 fe80::a8f7:f8a1:e375:
       valid_lft forever prefer
```

Solution 1

1. Send data to peer address
2. Done



Source

Destination

104.250.x.x

109.167.x.x

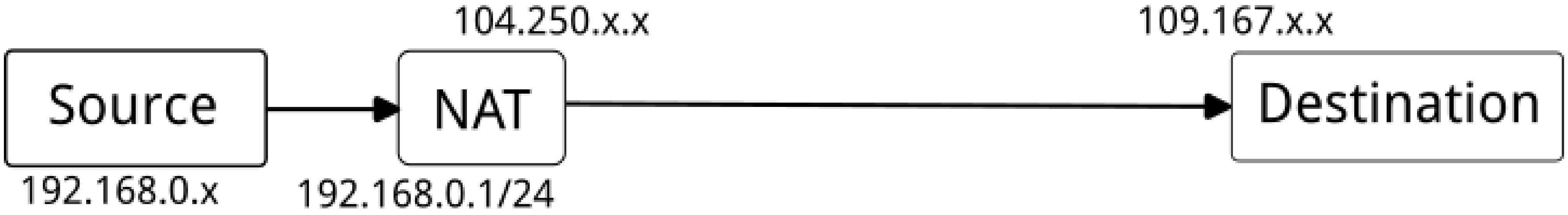
Problem

- Hosts are not always directly connected/accessible
- There may be an intermediate address translation layer that needs to be coerced into sending data to/from the correct device

Problem: NAT

Network Address Translation

- Estimated that 60%-80% of all devices are behind some form of NAT
- https://en.wikipedia.org/wiki/Network_address_translation



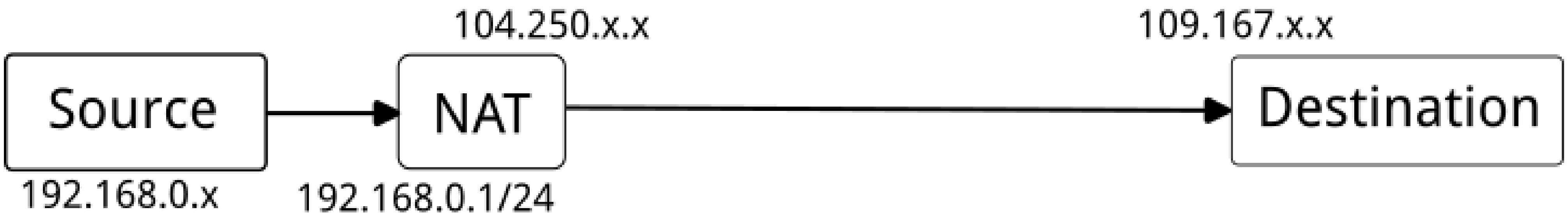
NAT Traversal (Or NAT behaviour)

- What happens to packets when traversing the NAT?

It Depends

What a NAT does

- Straddles 2 different network segments
  - May be called private network and public network
  - Or internal/external
- Rewrites IP address and port across the private/public boundary
- Adds internal mapping of the 5-tuple (or some subset) so response can be successfully sent to the requestor



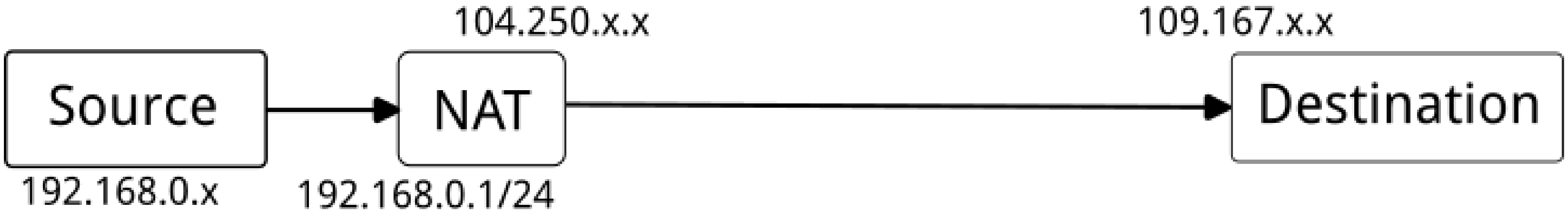
What constitutes a 'session' (from a NAT's point of view)

- Protocol (UDP/TCP/etc)
- Source IP Address
- Source Port
- Destination IP Address
- Destination Port
- aka 5-tuple

Solution 2

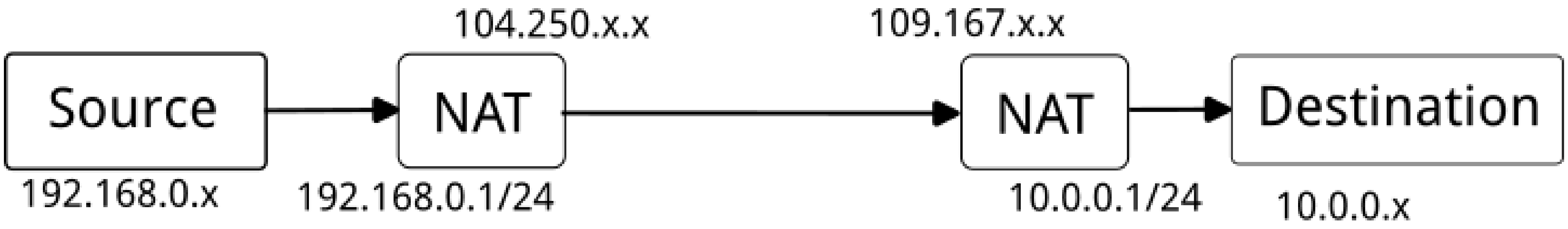
Have the device behind the NAT send initial data

- Client/Server model
  - Used by e.g. HTTP/FTP/SMTP/IMAP/etc
- Adds mapping on the NAT so response from the server is correctly routed



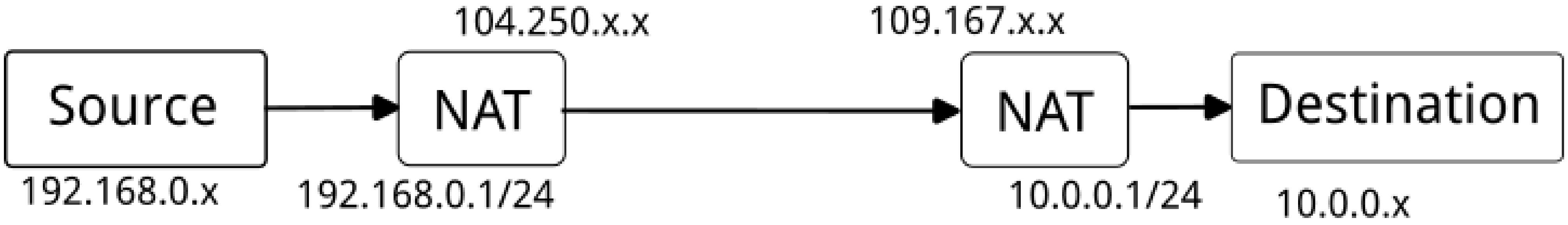
Problem: More NAT

This is where it gets interesting



Problem: More NAT

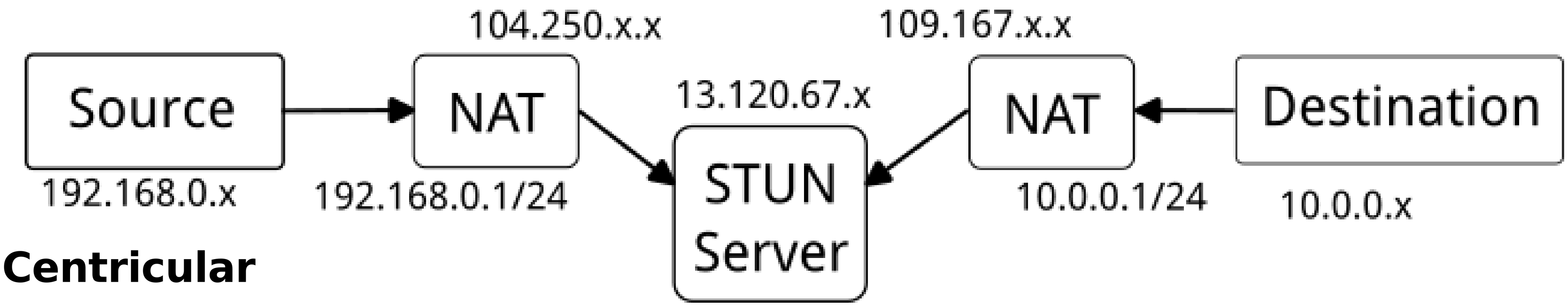
1. What IP address to send to?
2. How to let data through both NATs?



Solution: What IP address to send to?

Ask an external (STUN) server what IP address it sees the request from

- STUN: Session Traversal Utilities for NAT
- A client/server protocol
- Message based with extensible attributes
- One of the messages returns the address and port the server sees
- Low cost



Solution: How to let data through both NATs?

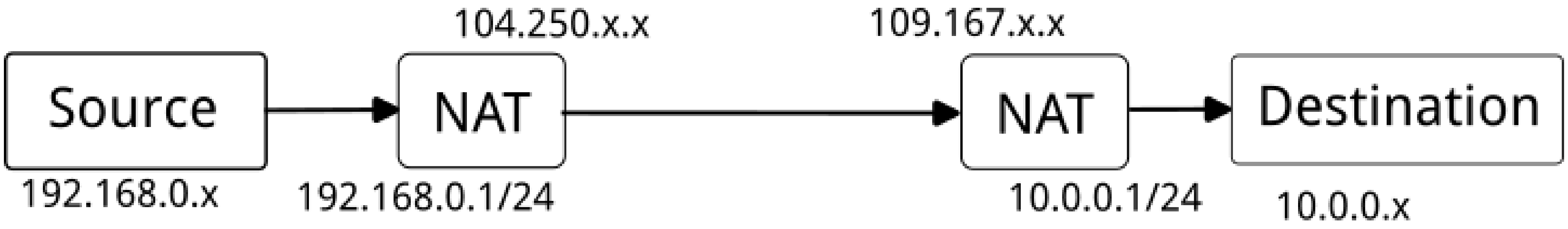
Both peers send STUN requests to their peer at the same time

- First packet hitting the peer's NAT will fail, but add the NAT mapping to the source's NAT
- Packet from the peer received by the source might succeed
  - On success, source tries again and will succeed (as all NAT mappings exist)

However!

This only works if the NAT mapping does not depend on the destination address and/or port

- Address (and Port) Dependent Mapping in RFC4787
- NAT may create a different external address/port for each 5-tuple
- Only a problem if both NAT exhibit this behaviour
- Not the common case or recommended behaviour



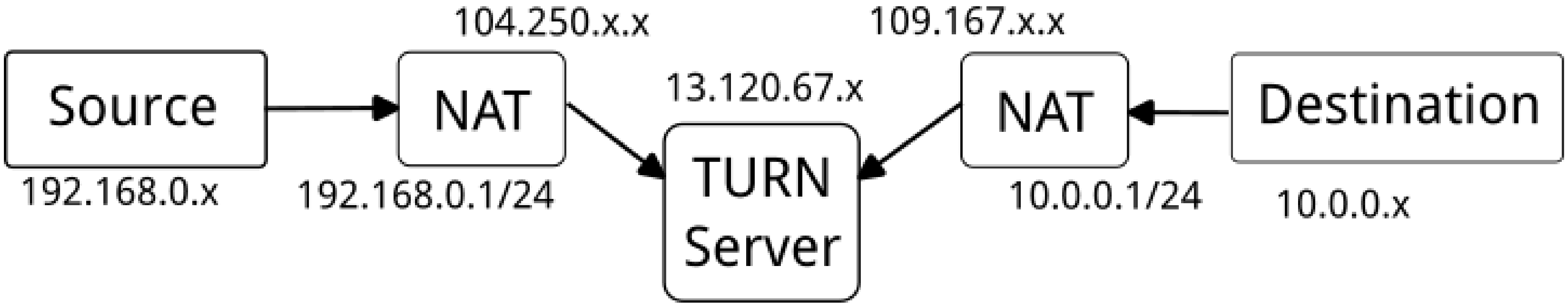
Centricular

Problem: NAT mappings that depend on the destination address/port

- This case is where only using a STUN server fails
- Cannot rely on off-path packets to do NAT hole punching

Solution: TURN

- TURN: Traversal Using Relays around NAT
- Send all data through intermediate server
- High bandwidth costs



Solution: TURN

- Solves connectivity in many different network topologies
  - Two Address (and Port) Dependent Mapping NATs
  - IPv4-only connecting with a IPv6-only device
  - UDP-only connecting with a TCP-only device
- A fallback to the client/server model

Putting it all together

ICE (Interactive Connectivity Establishment) Overview

1. Gathering
2. Try connecting (Connection Checks)
3. Choose a connection (Nomination)

Gathering Candidates

- Gather all the addresses we can send from
  - Host IP address
  - Ask external STUN server for the NAT external IP address/port
  - Allocate address/es on a TURN server
- Send all of these to the peer
  - Optionally as they arrive (trickle-ICE)

Candidate

- Component ID (RTP/RTCP)
- Protocol (UDP/TCP)
- IP address
- Port
- Priority
- Candidate type (Host, STUN, Relay)
- Username
- Foundation

Try connecting (Connection Checks)

- For each local and remote candidate combine them in priority order to create pairs
  - Send a binding request from the local socket, to the remote address
  - If success response, we have a valid pair

Error cases

- Timeout
- Not STUN response
- Missing STUN attributes
- Response not from address that was sent to
- STUN error code
  - Role conflict
  - Other error

Choosing a connection (Nomination)

- Once enough valid pairs (or some other criteria, e.g. timeout)
- Controlling agent nominates one of its valid pairs by sending another STUN binding request with a special STUN attribute (Regular Nomination)
- We have a successful connection!

Standalone Implementations

- libnice - GObject/C - used by webrtcbin/janus - https://gitlab.freedesktop.org/libnice/libnice/
- ice4j - Java - used by Jitsi - https://github.com/jitsi/ice4j
- libjuice - C - https://github.com/paullouisageneau/libjuice
- librice - Rust - very new - https://github.com/ystreet/librice
- webrtc-ice - Rust port of Go code - https://github.com/webrtc-rs/webrtc/

Thanks

- ystreet00 on #gstreamer on OFTC
- https://discourse.gstreamer.org/u/ystreet00
- https://gitlab.freedesktop.org/ystreet
- ystreet00@floss.social on mastodon

Centricular