RETURN

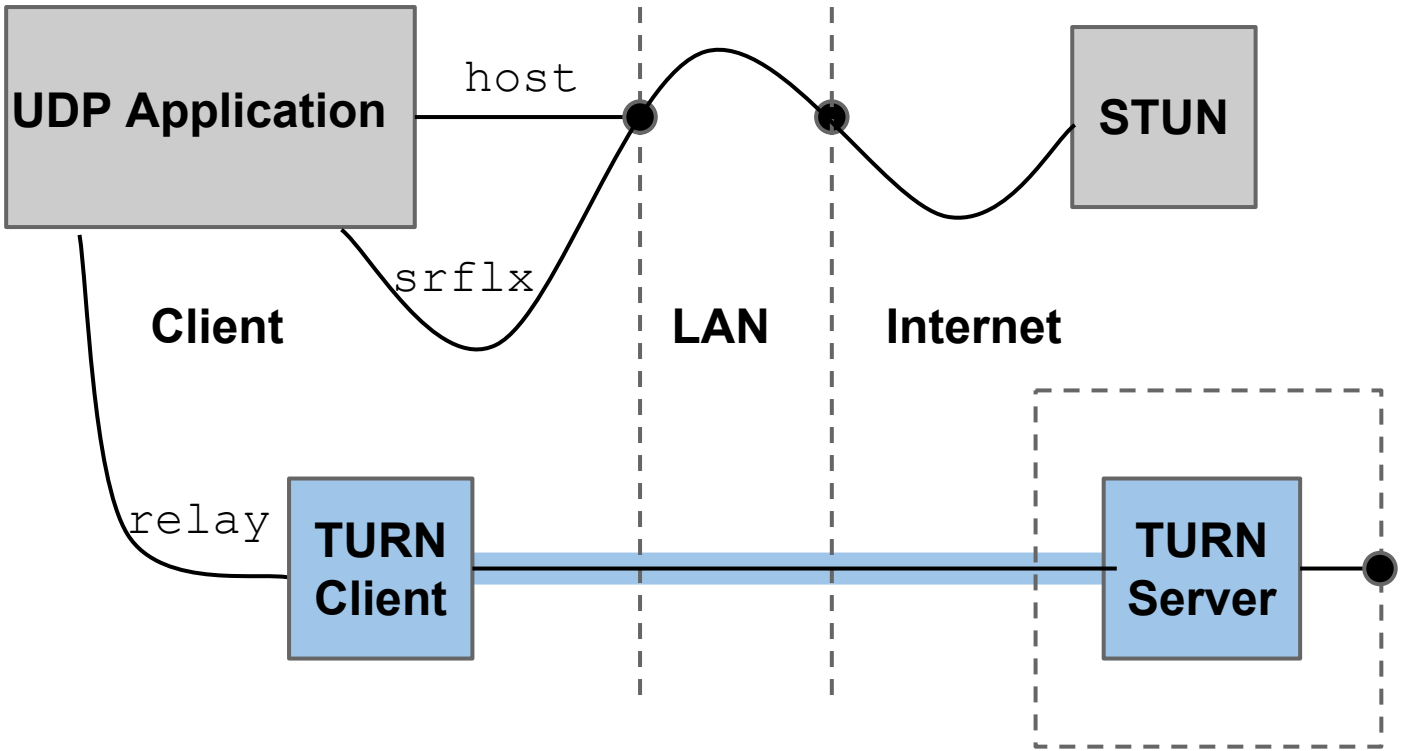
now with more TURN

Ben Schwartz
Justin Uberti

Background: TURN in WebRTC

- TURN server is configured by the page:
  - `new RTCPeerConnection([{urls:"turn:turn.example.org", username: "user", credential:"myPassword"}])`
- Produces a candidate like
  - `candidate:2157334355 1 udp 33562367 180.6.6.6 54278 typ relay raddr 46.2.2.2 rport 38135 generation 0`
- Used for **connectivity** (vs. NATs) and **privacy** (to hide one peer's IP from the other)

Classic TURN in WebRTC



Now: Enterprise relays (required!)

draft-ietf-rtcweb-use-cases-and-requirements-14, Section 3.3.5.1:

An enterprise ... deploy[s] a TURN server that straddles the boundary between the internal and the external network. … The RTCWEB functionality will need to utilize both network specific STUN and TURN resources and STUN and TURN servers provisioned by the web application.

Now: Enterprise relays (discovery)

draft-ietf-tram-turn-server-discovery:

To allow TURN applications operate seamlessly across different types of networks and encourage the use of TURN without the need for manual configuration, it is important that there exists an auto discovery mechanism for TURN services.

Bad options for handling these relays

- Just add them to the list along with the application's relays.
  - Wastes time on connection startup trying to reach unreachable application relays.
  - Reveals the peer's location to the other party, even if the application is trying to use TURN to conceal that.
- Force all WebRTC traffic through the relay.
  - Introduces unnecessary hairpinning for local or whitelisted connections.

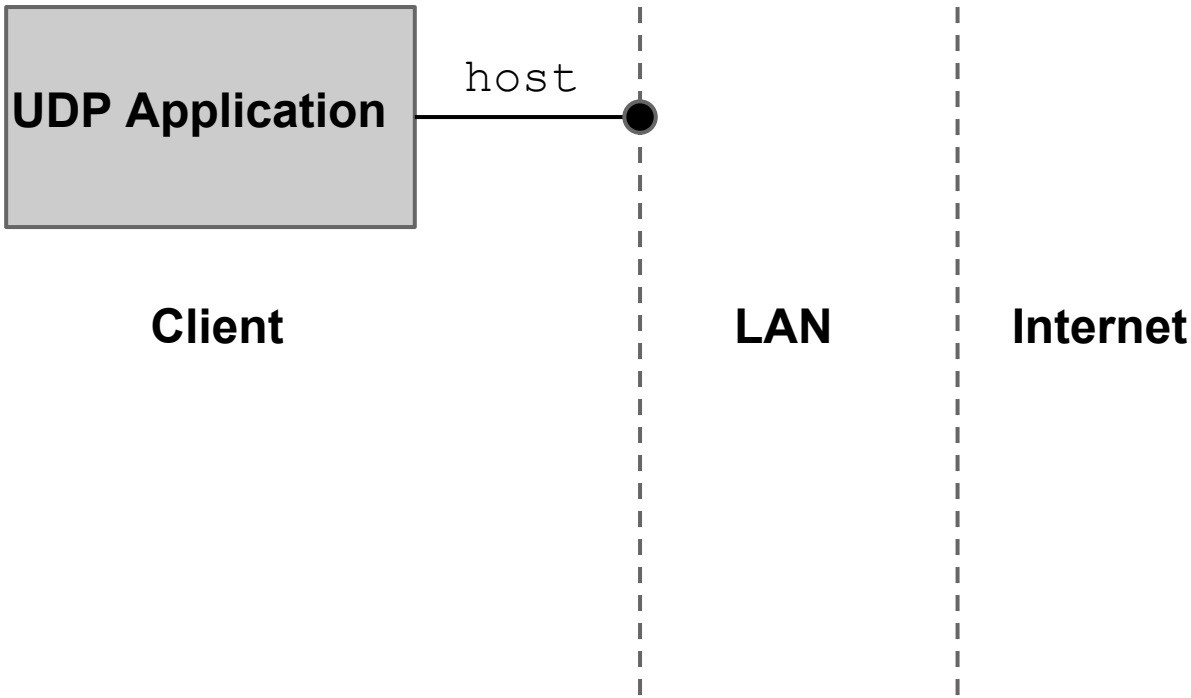
RETURN specifies a good behavior

- RETURN clarifies how browsers should treat an enterprise/non-webapp TURN server in WebRTC: as a **proxy**.
- RETURN specifies behaviors that make sense for autodiscovery, browsers' proxy control APIs, and browsers' proxy configuration user interfaces.

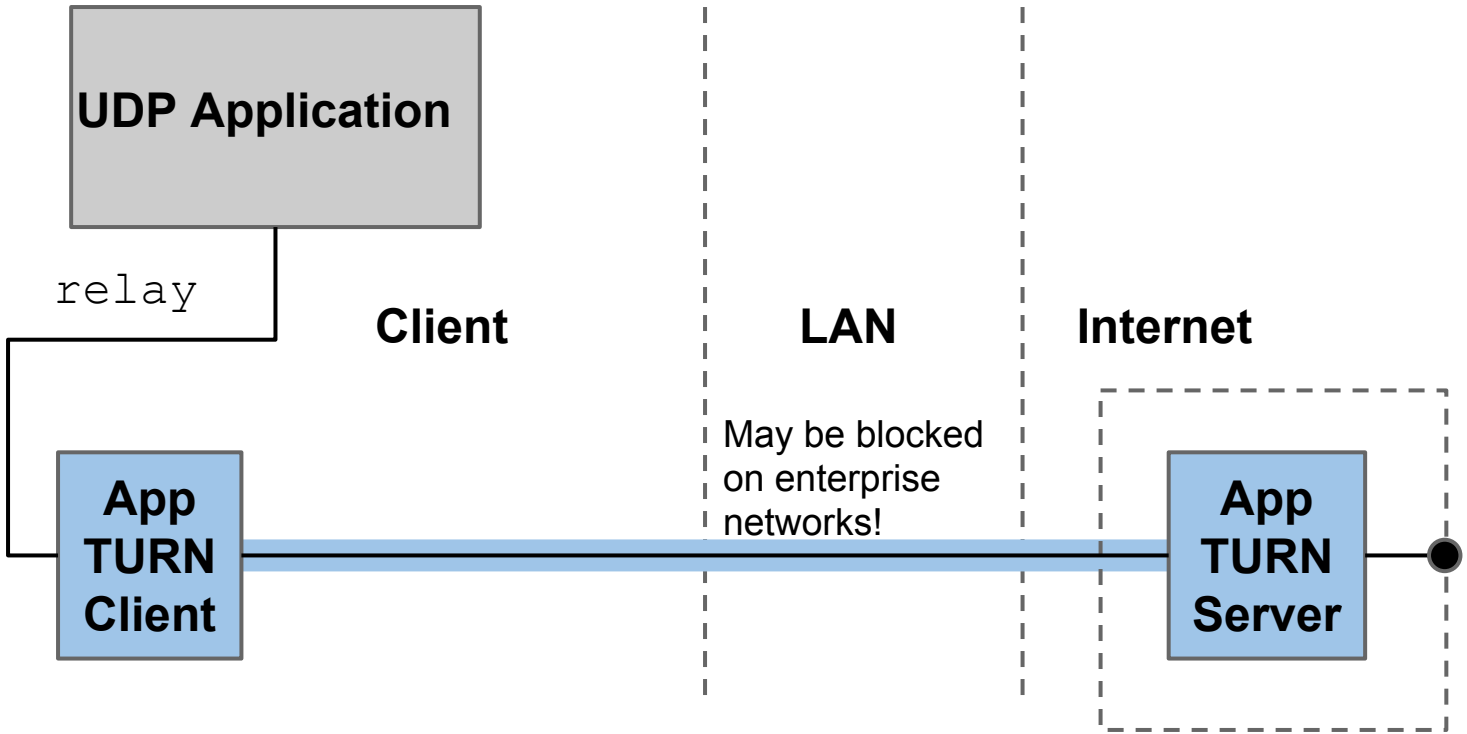
What kind of proxy?

- A lot like a SOCKS5 or HTTP CONNECT proxy, explicitly configured by an administrator or the user.
  - but for UDP. (SOCKS5-UDP is defined but not widely implemented.)
- **NOT** like a “transparent”, “intercepting”, “inline”, or “forced” proxy.

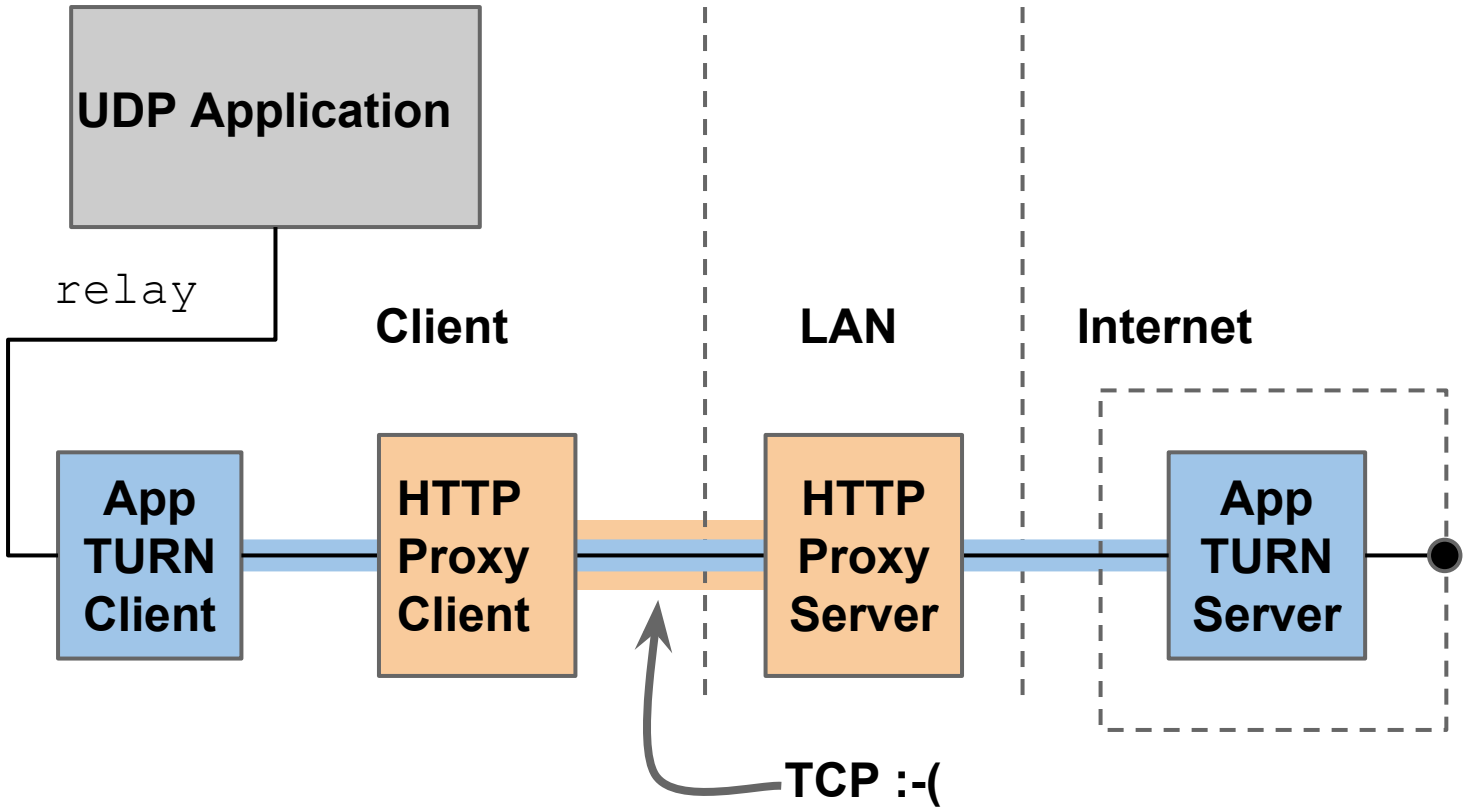
Example: Host candidate (normal)



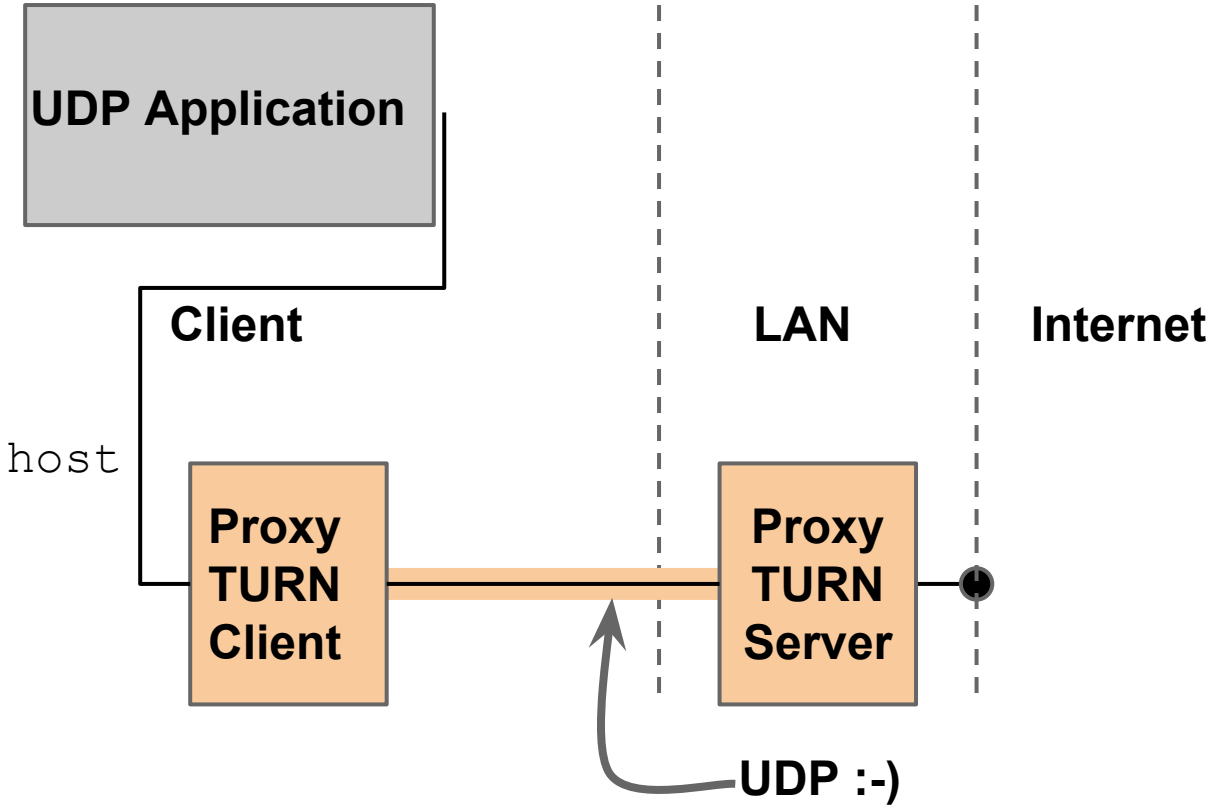
Example: TURN candidate (normal)



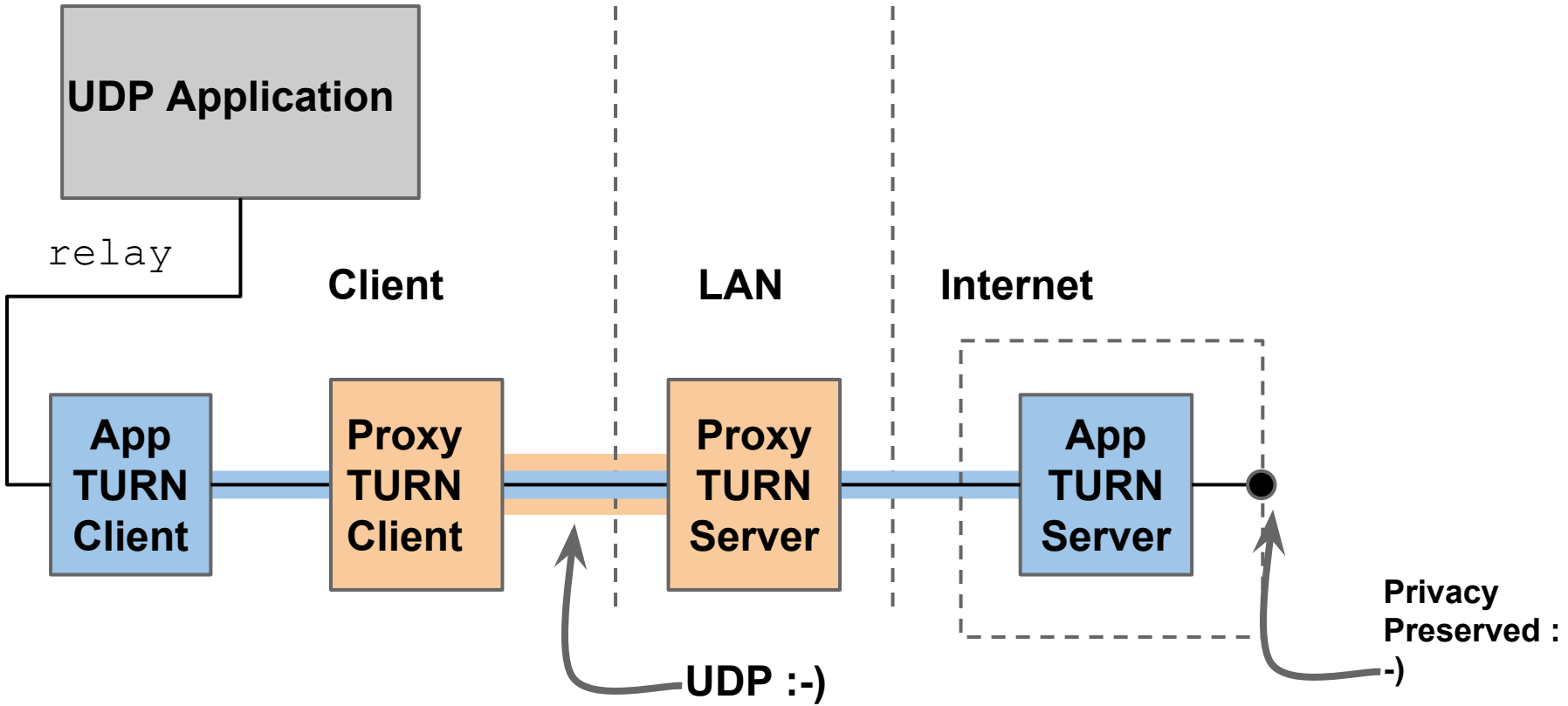
Example: TURN via HTTP Proxy



Example: Host candidate via RETURN



Example: TURN via RETURN



Questions resolved by RETURN

- Are enterprise TURN servers added to the list of RTCIceServers?
  -
- Are they visible to the web app?
  -
- What type of candidates do they generate?
  -
- Can they chain recursively through the web app’s TURN server?
  -
- Is all traffic forced through the relay?
  -

RETURN's answers

- Are enterprise TURN servers added to the list of RTCIceServers?
  - **No.**
- Are they visible to the web app?
  - **Not explicitly (but the web app can probably detect it).**
- What type of candidates do they generate?
  - **"host", or sometimes "relay" (see next question)***
- Can they chain recursively through the web app's TURN server?
  - **Yes. (This makes a relay candidate.)**
- Is all traffic forced through the relay?
  - **The network administrator can choose.**

*** and never "srflx" (except if the proxy is actually behind NAT!)**

In conclusion

RETURN

- DOES
  - specify precise behavior for browsers to use with enterprise and other non-application TURN servers.
  - enable improved privacy and connectivity.
- DOESN’T
  - introduce any new API or protocol.
  - require any changes to browsers that don’t yet support non-application TURN servers.