RETURN-05

for IETF 92, 2015 March 23-27

Ben Schwartz
Justin Uberti

Changes since last time

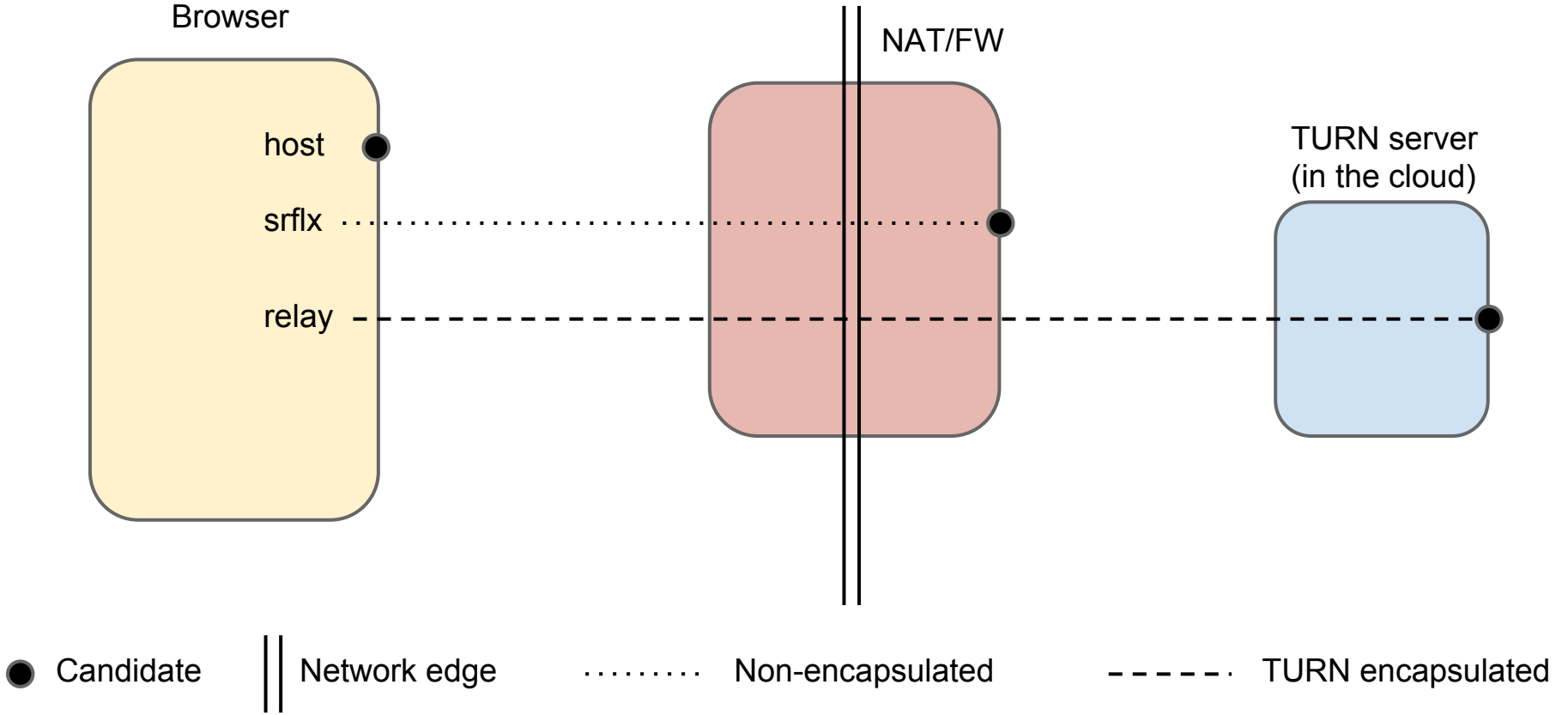
- New figures and visual overview
- Broadened motivation section to include more reasons for TURN
- Clarified rationale for nomenclature
- Added discussion of multi-tenant (cloud) TURN servers

Thanks to Alan Johnston and John Yoakum!

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, QoS, routing through fast private networks, monitoring, recording, troubleshooting, and IP privacy.

Classic TURN in WebRTC

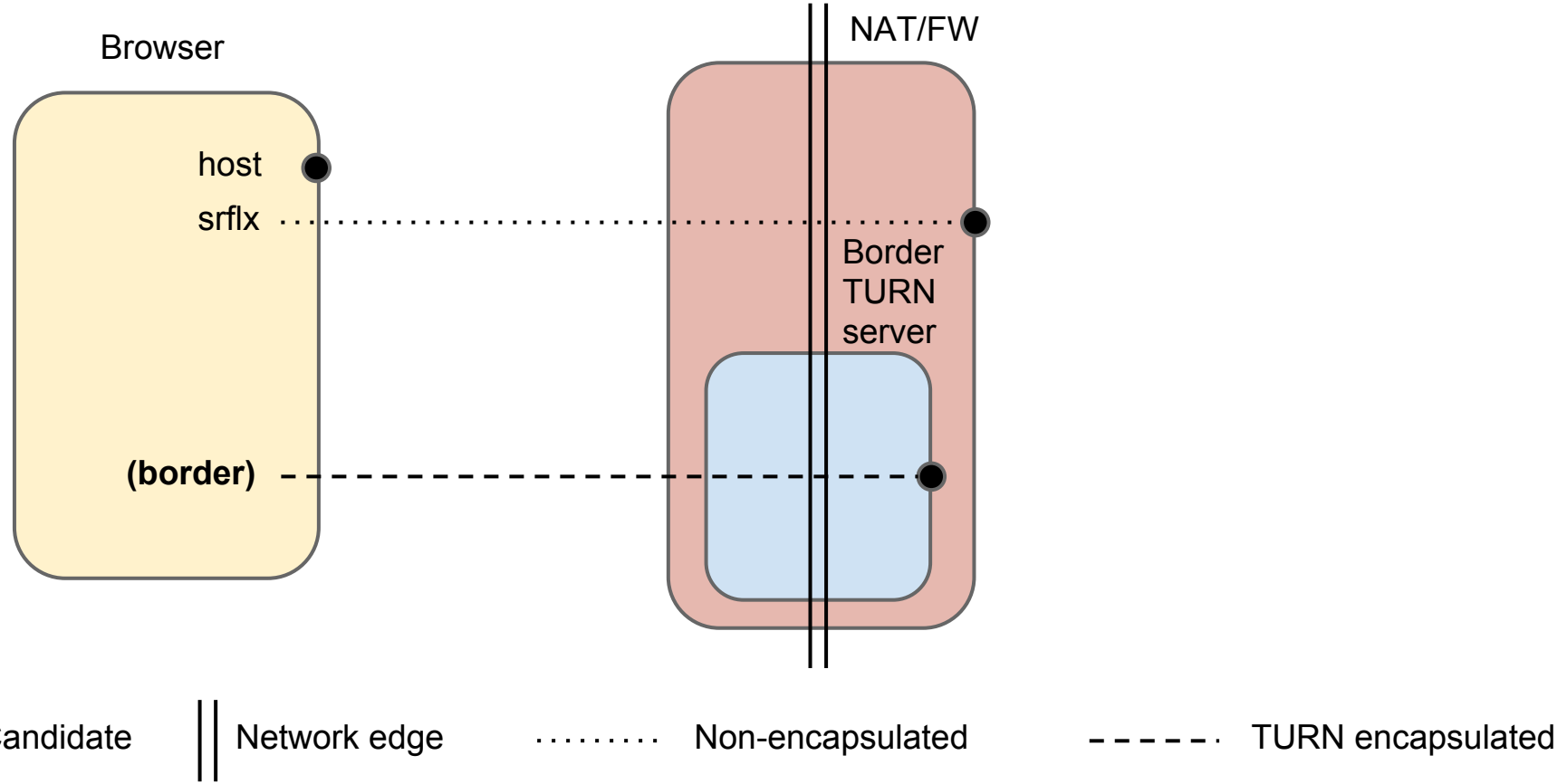


Now: Enterprise relays (required!)

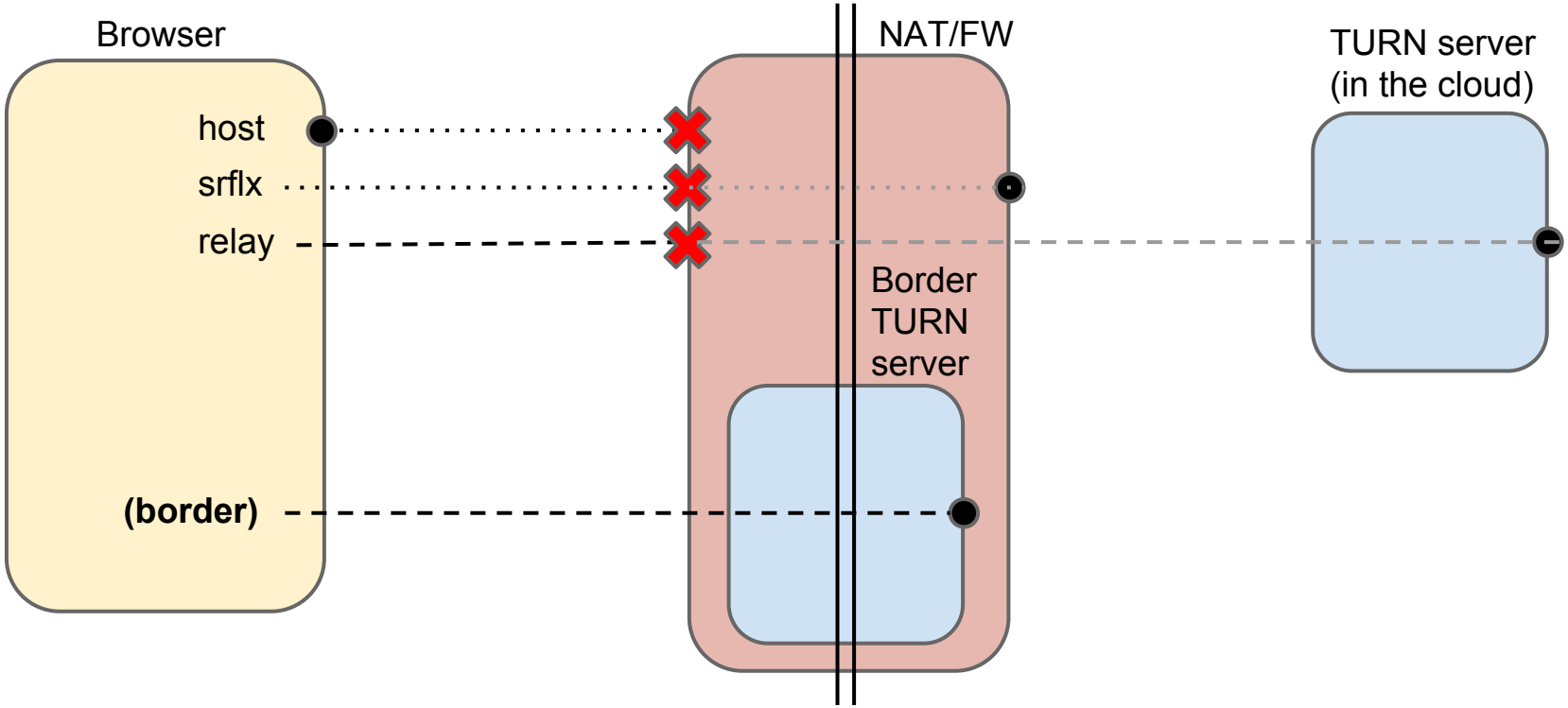
RFC 7478, Section 2.3.5.1:

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

Border TURN server (enterprise)



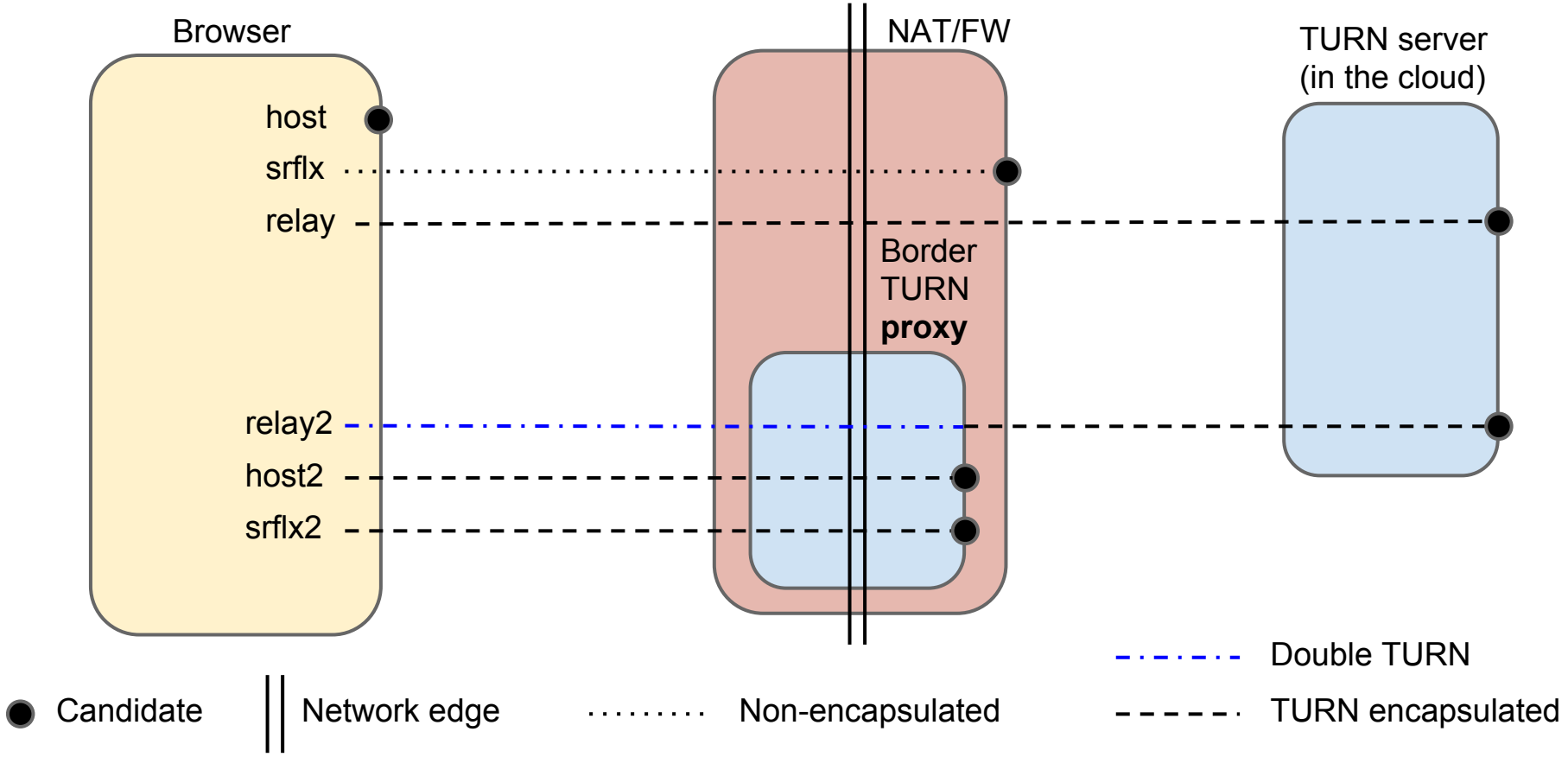
Border TURN server and UDP block



What about this (border) candidate?

- Client doesn’t know what kind of candidate to generate.
  - There is not yet any specification for how Web should interact with a Border TURN server tha not provided by the application.
- RETURN answers this: the port allocated on the Border TURN server should be treated as a **virtual network interface**.

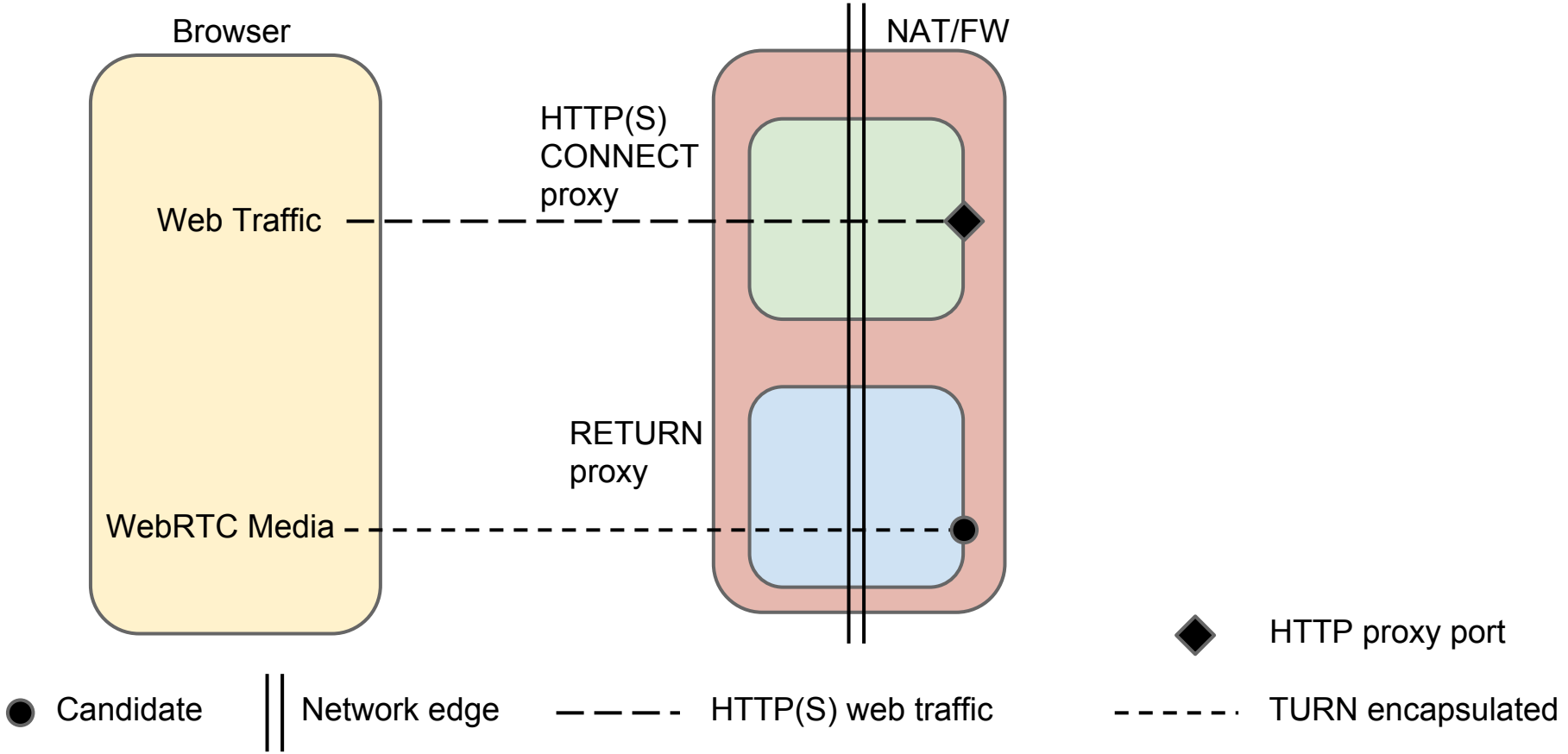
Border TURN proxy with RETURN



Why do we call it a proxy?

- It’s analogous to an HTTP CONNECT or SOCKS proxy.
  - It performs a similar function and can be configured in a similar fashion.
- It will be the destination of traffic generated by the client.
  - **NOT** like a “transparent”, “intercepting”, “inline”, or “forced” proxy.

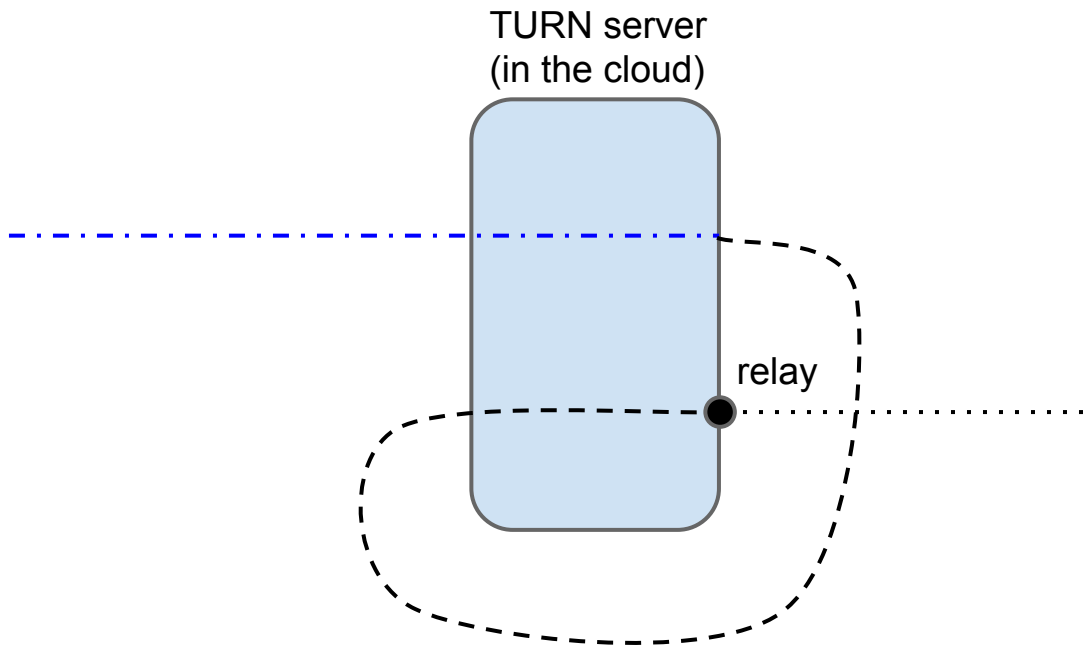
Comparison to an HTTP proxy



What if it’s actually the same server?

- e.g. the enterprise/proxy-provider and the application both contract with the same multi-tenant (cloud) TURN server operator.
- New clarification: in this case the client MUST transit the server **twice**!
  - Otherwise authorization and origin labeling will not work, and metadata may leak to the wrong party.
  - Can always revisit if we find a safe optimization.

Same server twice



Candidate Non-encapsulated TURN encapsulated Double TURN

In conclusion, RETURN

- still
  - specifies precise browser behavior to help us meet our enterprise configuration requirements.
  - doesn’t introduce any new API or protocol.
- but now includes
  - better figures and a visual overview/introduction
  - clearer explanations of motivation and corner cases
  - feedback from a wider range of implementers