What is WAVE Sync?

Affected Roles: System Owner • System Administrator
Related WAVE VMS Apps: WAVE Server • WAVE Sync
Difficulty: Low

Allows connecting to the WAVE Server behind a NAT without setting up port forwarding.

Services that take part in the WAVE Sync

Mediator

This is an intermediate service which allows the server and the client (each behind its own NAT) to find out the address of each other to establish a direct connection.

Relay (proxy)

This is another cloud service responsible for relaying traffic between the client and the media server in the event that UDP hole punching does not work.

Media Server

If a System is connected to WAVE Sync:

- Just after it starts, Server reports all of its local addresses to the mediator. It is needed in case the media server is installed on a machine with a public IP.
- The media server connects to the relay and waits for incoming requests.

Client

The client takes the following steps to connect to the media server:

- If the media server can be found in a LAN using multicast, then the client establishes a direct TCP connection to that server. In this case, the mediator is not used.
- Otherwise, the client sends a *connect* request to the mediator.

- The mediator responds with all known addresses of the media server, this includes TCP addresses (forwarded and local) and the UDP address obtained by punching a hole in NAT.
- After receiving this information, the client tries to establish a connection to every address simultaneously.
- At the same time the client tries to connect to the media server through a cloud relay service.

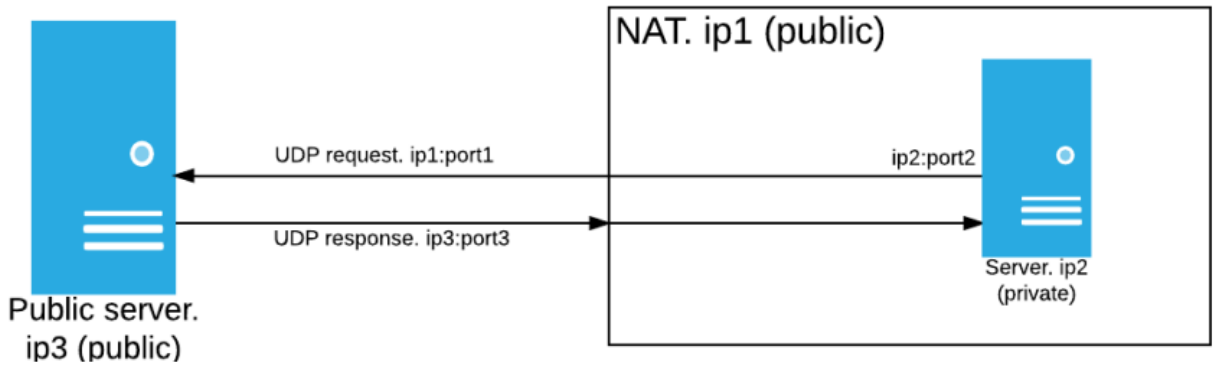
Note: To minimize connection time, there is no preference between TCP or UDP. In some network environments, TCP may be the only option, while in others - UDP. Because of this possibility, giving preference to one of them will slow down the time it takes to connect.

NAT traversing methods

UDP hole punching

Description

NAT allows a host to send UDP packet to some public UDP service.



When the host sends a UDP packet, NAT allocates a public port (port1 on address ip1) for that packet and sends the request as if it has been sent from *ip1:port1*.
When that service sends a response back to the *ip1:port1*, NAT knows it is actually Server that awaits the response and forward response to it.
So, when sending a UDP request from within NAT to the public, NAT punches a **hole**. This method is used in hole punching to establish a reliable connection over UDP between two peers, each behind its own NAT.

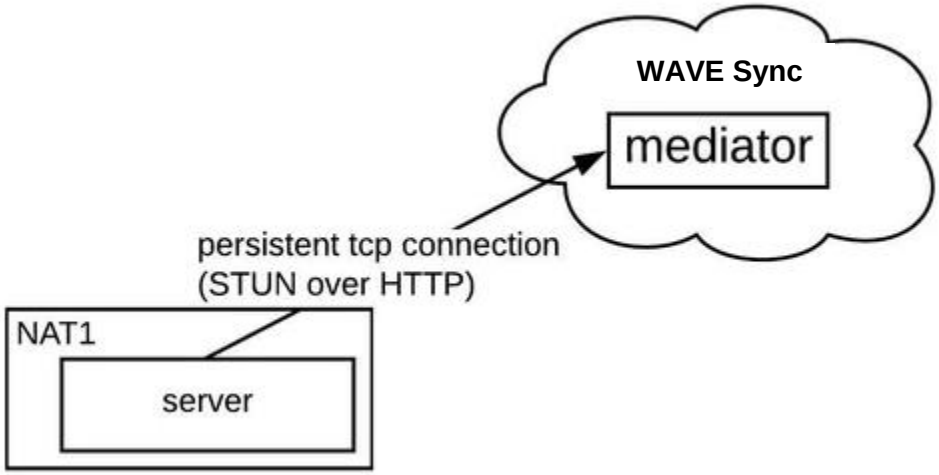
Important:

UDP hole punching does not guarantee success in 100% of cases. Success is dependent upon client and media server NAT types.

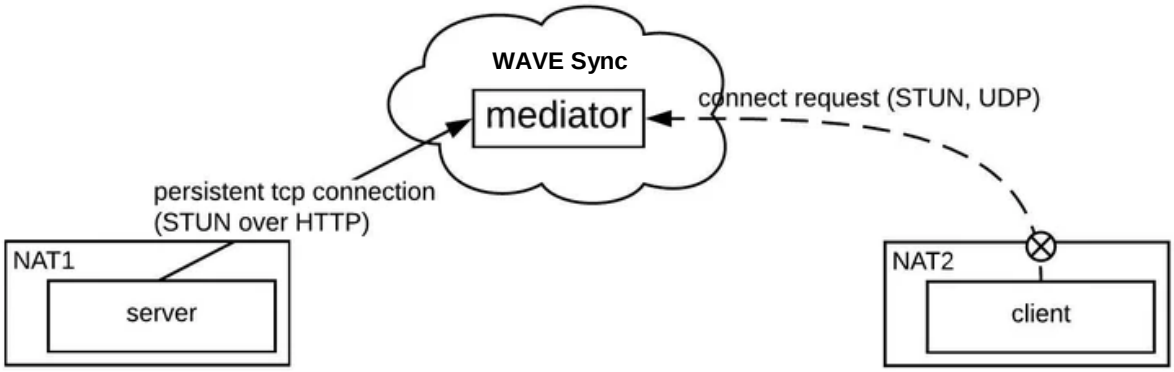
All NAT types except for symmetric NAT allow for UDP hole punching. Although a firewall/antivirus can still block UDP traffic (like any other traffic, actually).

Algorithm

1. In its initial state right after it starts, the server connects to the mediator. The mediator identifies itself by providing its GUID and waits for connection requests. The server maintains a persistent TCP connection to the mediator.

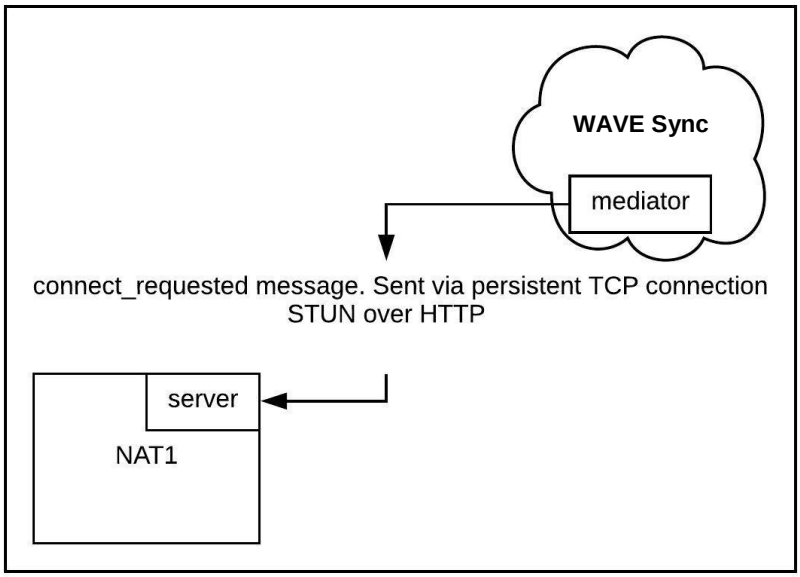


2. The client sends a *connect* request to the mediator via UDP, providing the GUID of a System/specific server it wants to connect to.

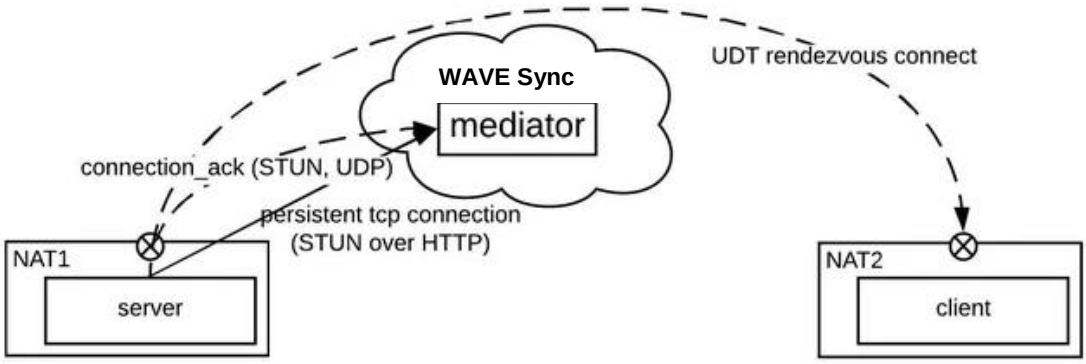


3. The mediator identifies the server by its GUID and sends a *connection_requested* message to the server via the pre-established TCP connection (this message contains the client's UDP address). Note: The client can specify the GUID of the whole System, not just a specific

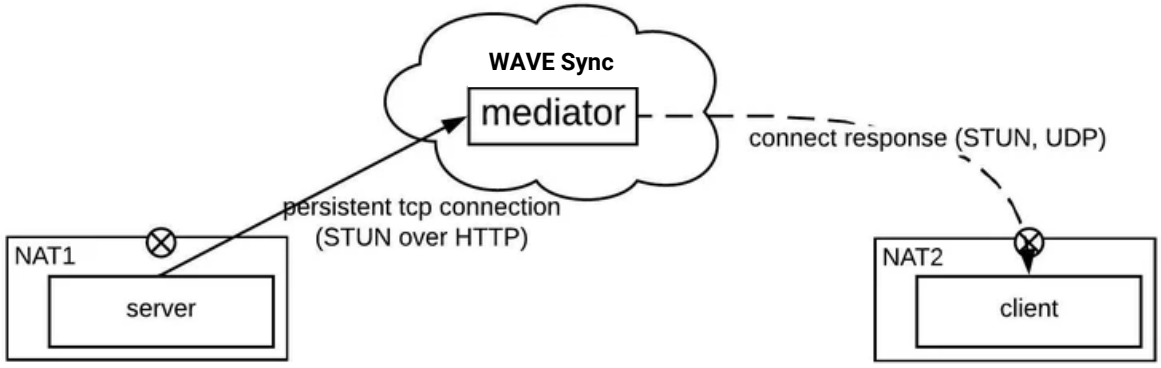
server. In this case, the mediator will select any online server of that System.



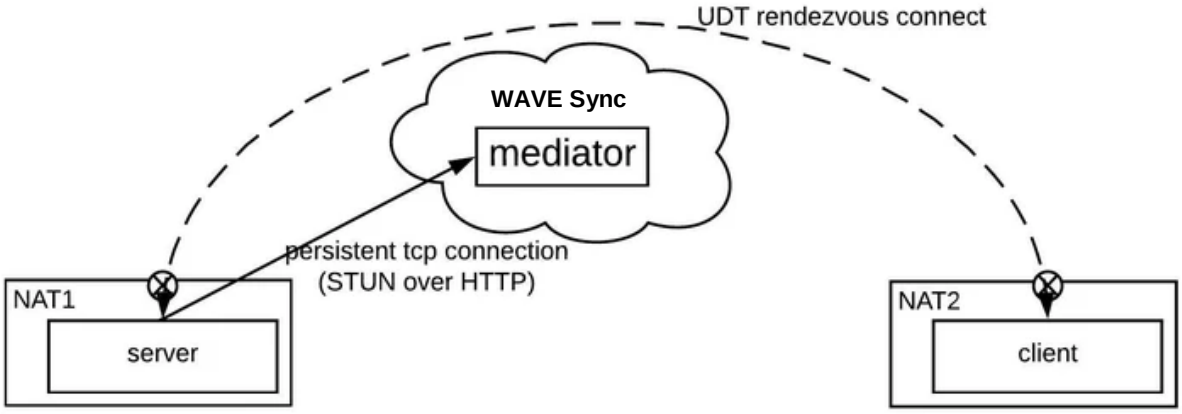
4. The server opens a hole on its side and simultaneously
    a. sends a *connection_ack* request to the mediator using UDP, and
    b. initiates UDT rendezvous connection with the client address provided in the *connection_requested* message (i.e., sends a number of UDP packets to the client).



5. The mediator receives a *connection_ack* request from the server and sends a *connect reponse method to the client* message to the client.



6. The client receives a *connect_response* message and initiates UDT rendezvous connection to the server address found in *connect_response*. As a result, we now have a UDT connection between the server and client.

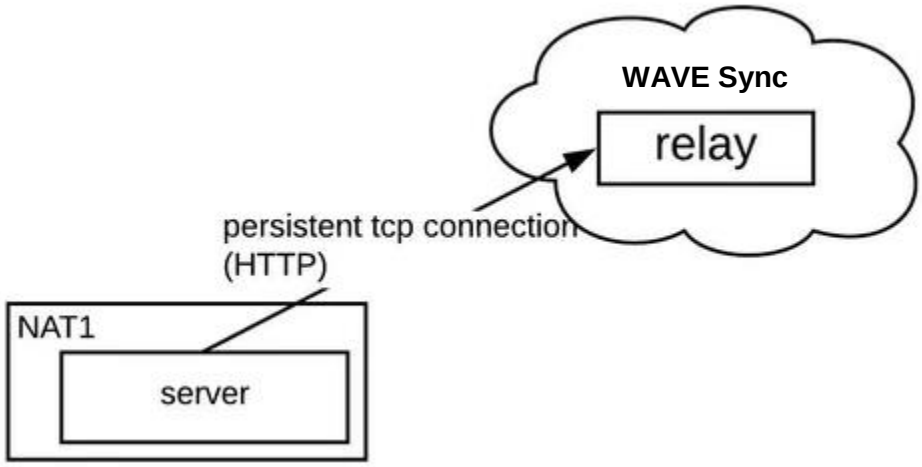


Opening subsequent connections between the server and the client does not require all these steps to be repeated. Having at least one active UDT connection means that holes are punched in both NATs. Subsequent connections can be established with a regular UDT mechanism (similar to a TCP connection to a directly accessible address).

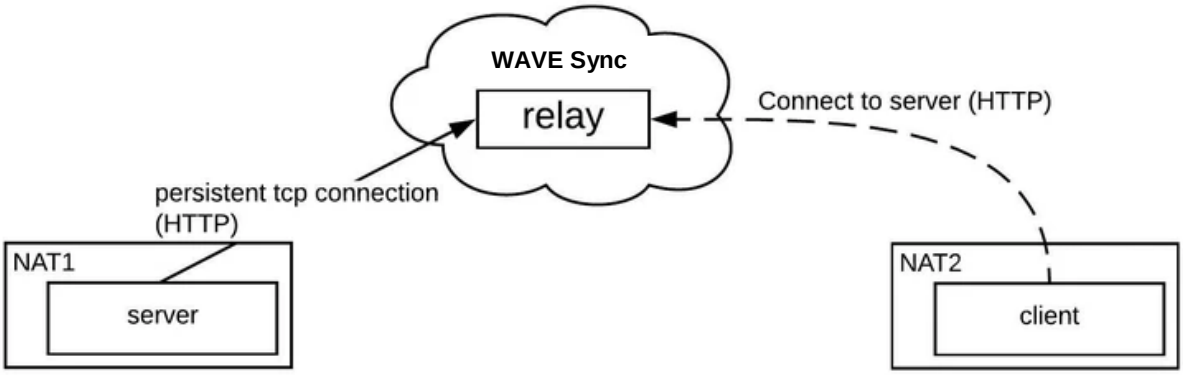
Relaying (proxying)

In this case, all traffic between the client and the server is relayed through the corresponding cloud service. It is the server and the client who are responsible for encrypting traffic. The cloud relay does not force encryption and does not distinguish SSL/non-SSL traffic when relaying.

1. The server establishes a connection to the relay and waits for requests from the client.

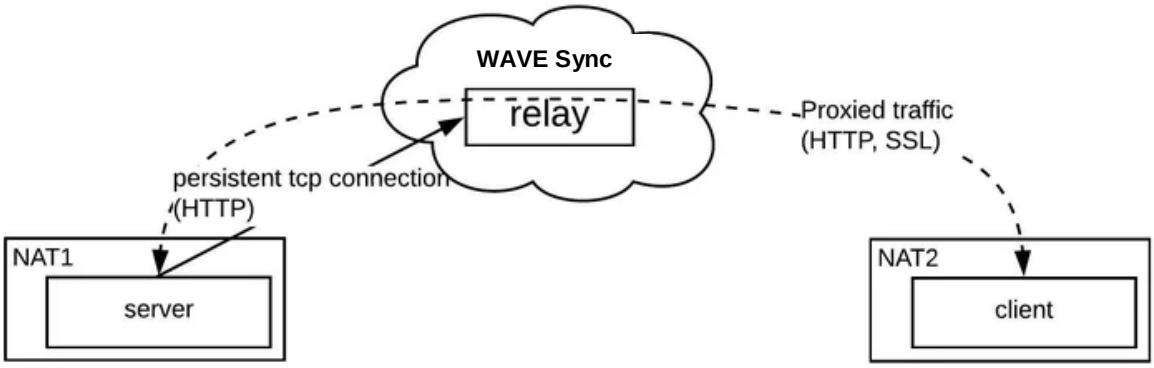


2. The client sends a request to the relay asking to provide a relayed connection to the server.



3. The relay uses an existing connection established by the server to transfer the client's traffic back to the server. At the same time, the server establishes another connection to the relay and waits for more client requests. To minimize client-side latency, the server always keeps at least 7 inactive connections to the relay (it is common for a client to request multiple

connections to the server simultaneously).



Common problems

The problems described below will prevent you from connecting to the server.

The server is behind a Squid-like HTTP proxy

`This impacts server - mediator (UDP hole punching algorithm, p.1) and/or server - relay (Relaying, p.1) connections.`

The Squid (http://www.squid-cache.org/) does not support any kind of HTTP tunneling (e.g., WebSocket or POST with the infinite body).

As a result, the media server can have problems establishing a persistent HTTP connection from to the mediator or the cloud relay.

Other HTTP proxies and firewalls can have different limitations on HTTP dialect.

We address these limitations by introducing multiple HTTP tunneling methods, probing and selecting the most suitable one for the location (introduced in 3.2 patch).

UDP packets from the cloud do not reach the client

`Impacts client - mediator message exchange (UDP hole punching algorithm, p.5).`

Various intermediate networking hardware/software (e.g., firewall/antivirus) can lead to non-working UDP. As a result, the response from the mediator never reaches the client and the connection fails.

This is addressed by duplicating a UDP request with a TCP request in VMS 4.0.

To report an issue

Open a support ticket with us here or:
https://wavevms.com/support/