BirdDog Connect networking overview

The BirdDog Connect daemon connects to the BDC Backend (connect.birddog.tv (on ip addr. 3.33.242.17 or 15.197.194.245) api.ipify.org and ipecho.net (used for getting the EP public ip address) over HTTPS (**TCP Port 443**). Users can override the pulic ip address on the endpoint and in that case the two latter domains (api.ipify.org and ipecho.net) will not be connected. It also needs a Websocket connection to the BDC Backend on **TCP Port 443**.

For the normal SRT use case (Fig. 1), the Backend communicates each Endpoint's public IP address to the other, both try then to connect over UDP on the port specified on the connection. This is what SRT calls "Rendezvous Connection Mode". It is assumed that the Firewalls on both sides support UDP Hole Punching or otherwise allow inbound traffic on the same port. The UDP connection is bidirectional, as SRT needs to communicate back to the sender. NB! This means that Static Port Mapping MUST be allowed. The firewalls in use might not allow this and then Caller --> Listener or Listener --> Caller must be used. The Endpoint that is set as Listener then need to have the UDP port forwarded to the local ip address of the Endpoint in the firewall.

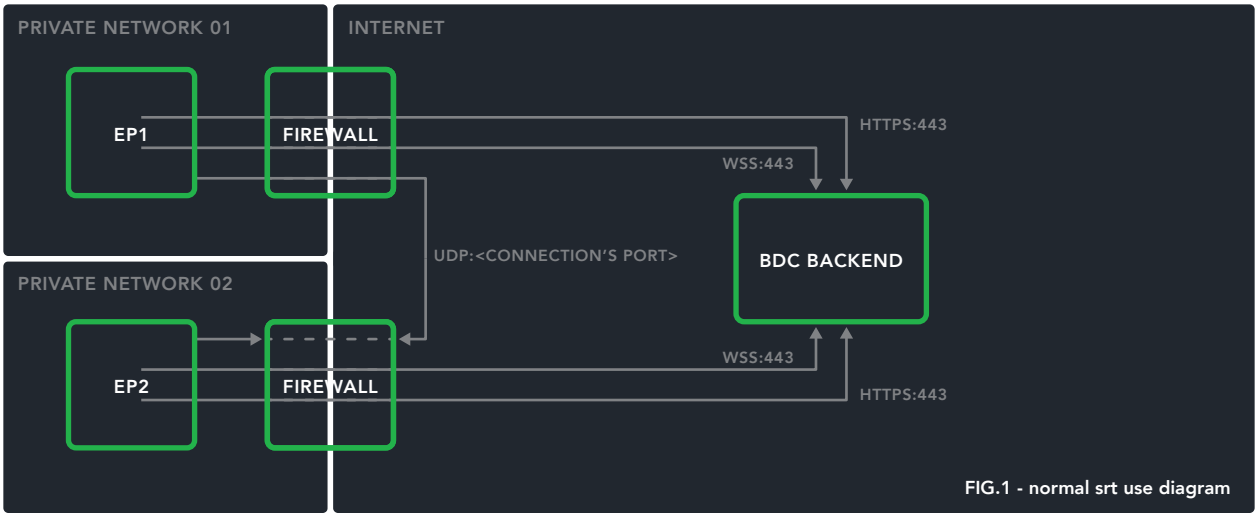

FIG.1 - normal srt use diagram

In the case that both Endpoints share the same public IP address, this won't work because they would try to connect to themselves. This is typically the case if they are both on the same network (Fig. 2). If both Endpoints on a connection have their "Local IP" field filled, the Backend instructs them to connect to each other's "Local IP" instead.

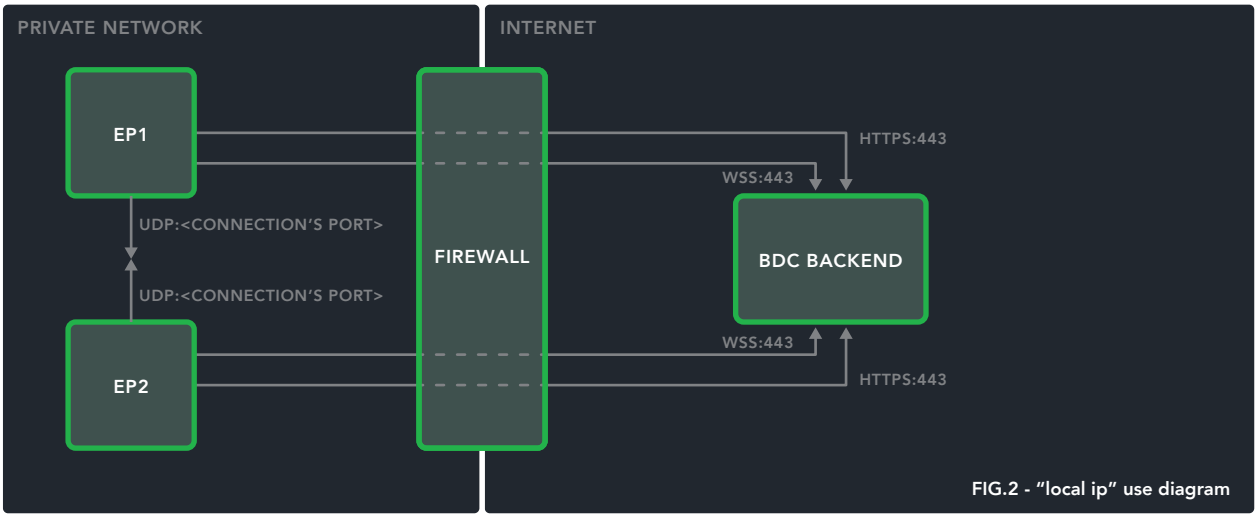

FIG.2 - "local ip" use diagram

DX (Fig.3) is a distribution endpoint that can run in an isolated environment (for example on AWS EC2 -

running Ubuntu Server 22.04) to act as and SRT relay. The TX EP connects to DX first and then a RX EP can connect and receive that stream. Since the DX can act as a SRT Listener for both the TX EP and RX EP it can be used as a firewall traversal. NB! That means however that DX have to have UDP ports opened inbound in to it. In addition it needs the same https and wss connecitons as a normal endpoint on port **443.**

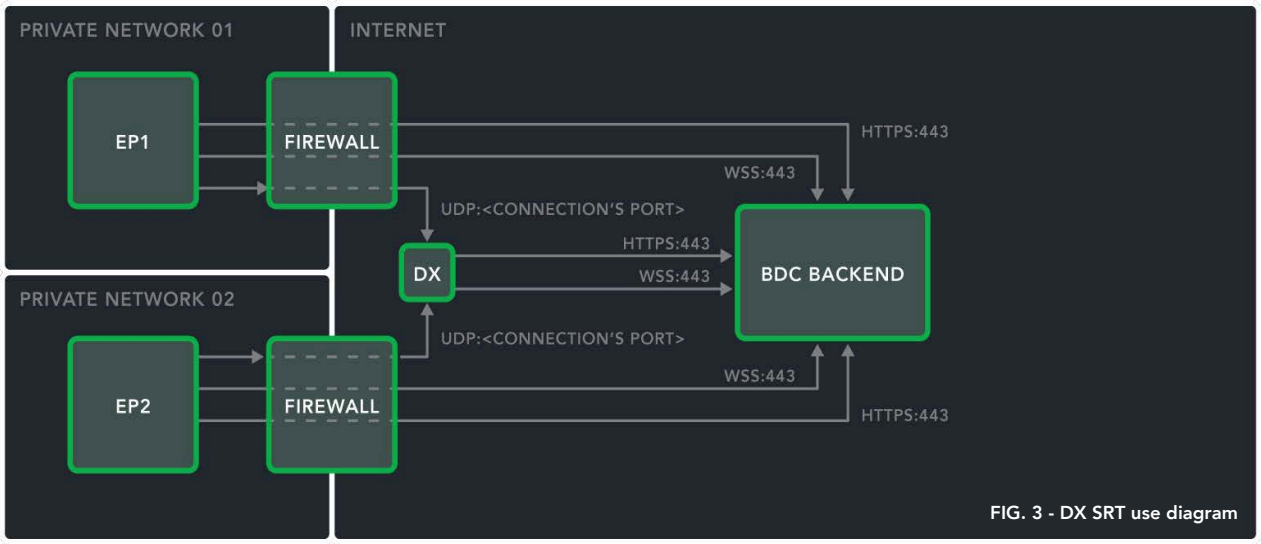

FIG. 3 - DX SRT use diagram

With **Connect WebStream** things get less transparent (Fig. 4). In addition to the BDC Backend, the daemon and the client browser also need to connect to the Company's configured TURN Server. The standard for this (STUN) is **UDP port 3478** or **TCP port 5349 if using TLS**. This however depends entirely on the Company's configuration (if they user their own TURN server).

The WebRTC peer connection that carries the stream then goes any possible way discovered by the TURN server and it might as well go the other way around (i.e. from the daemon to the browser). If a direct connection cannot be established, the TURN server relays all traffic. In this case both parties connect out to the TURN server.

It is to note that both parties might also be on the same network, the TURN server should then be able to discover a direct connection that doesn't traverse the internet.

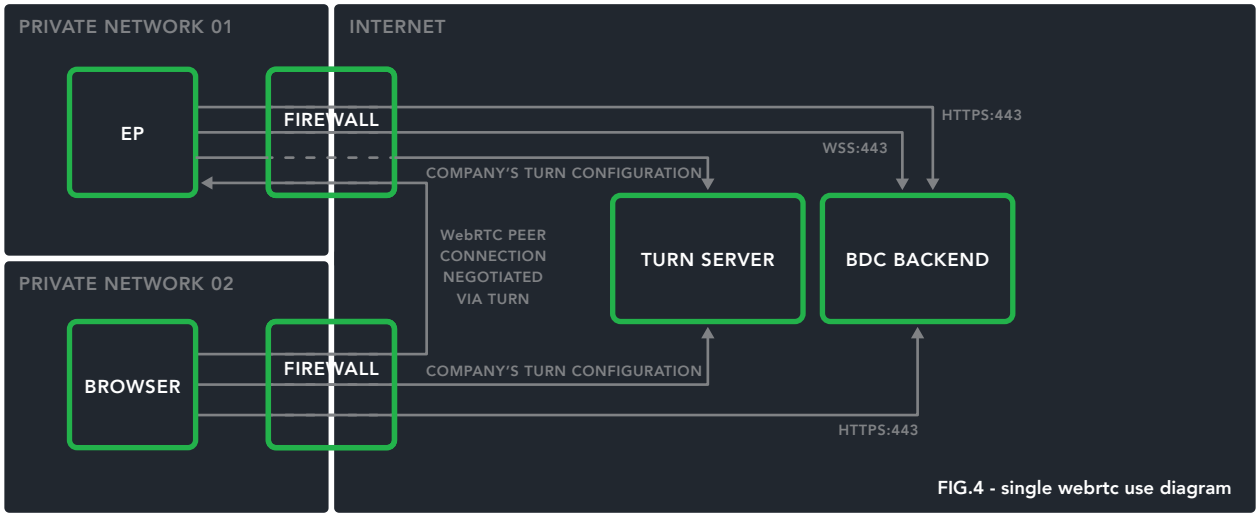

FIG.4 - single webrtc use diagram

Connect WebStream with the broadcast option (Fig. 5) again uses some more connections. Both the Endpoint and the browser need to connect to the Dolby.IO signaling server (director.millicast.com) on **TCP port 443**. The outgoing stream from the Endpoint again uses the Company's TURN server, the player in the browser however uses one provided by Dolby.IO.

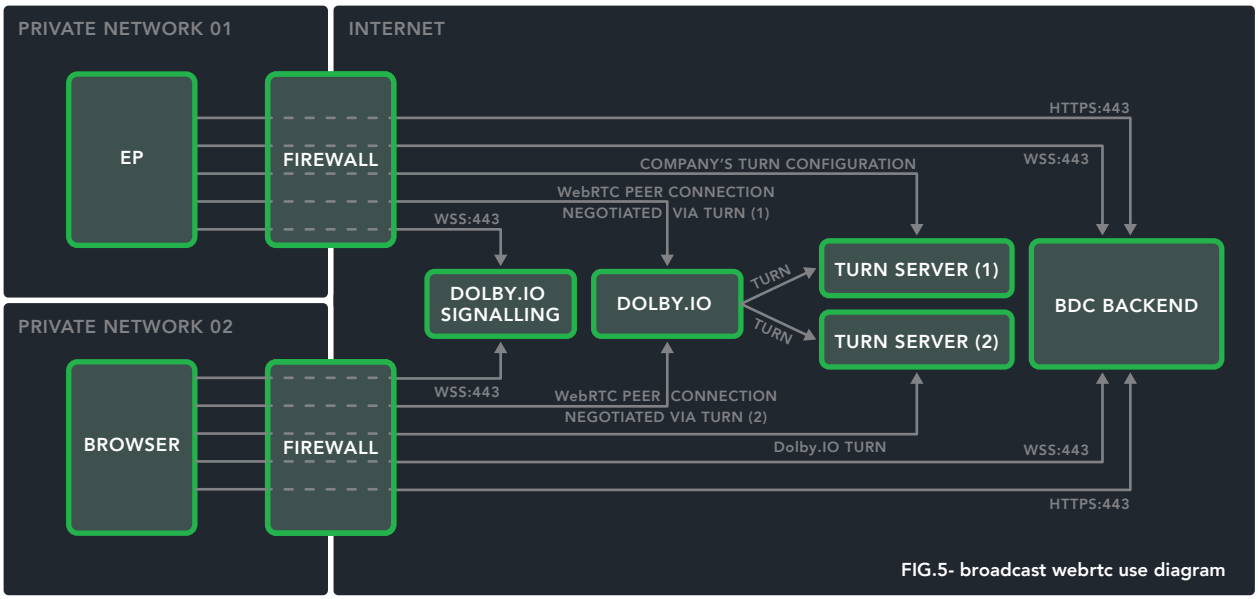

FIG.5- broadcast webrtc use diagram