WebRTC and Firewall Issues

WebRTC uses the UDP protocol (and may use TCP in some cases when UDP is not available) to transport your desktop.

While WebRTC is the reason Selkies-GStreamer enjoys low latency and better performance compared to other solutions, your network firewall may still prevent you from connecting to your instance.

(IMPORTANT) Instructions here are mandatory if the HTML5 web interface loads and the signaling connection says it works, but the WebRTC connection fails and therefore the remote desktop does not start.

The Easy Fix

This section is intended for starters and self-hosting users. You may require a TURN Server for more complex configurations.

Self-Hosted Instances

For standalone self-hosted instances, **open UDP and TCP ports 49152–65535 in your host server network**.

A configuration in your internet router called `Full Cone NAT` (otherwise called peer-to-peer/P2P mode or game mode) or (if not available) `(Address-)Restricted Cone NAT` is another alternative.

Containers

For an easy fix for containers, add the option `--network=host` to your Docker® command, or add `hostNetwork: true` under your Kubernetes YAML configuration file's pod `spec:` entry, which should be indented in the same depth as `containers:` (note that your cluster may have not allowed this, resulting in an error).

Note that running multiple desktop containers in one host under this configuration may be problematic and is not recommended. You must also pass new environment variables such as `-e DISPLAY=:22`, `-e NGINX_PORT=8082`, `-e SELKIES_PORT=8083`, and `-e SELKIES_METRICS_HTTP_PORT=9083` into the container, all not overlapping with any other X11 server or container in the same host. Access the container using the specified `NGINX_PORT`.

This exposes your container to the host network, which disables container network isolation.

UDP and TCP ports 49152–65535 in your host server network should still be open or be under `Full Cone NAT` as in Self-Hosted Instances.

If this does not fix the connection issue (normally when the server is behind another additional firewall), you cannot use the above fixes for security or technical reasons, or when deploying multiple desktop containers in one host, **move on to the next section**.

TURN Server

A TURN server is required if trying to use this project inside a Docker® or Kubernetes container without host networking, or in other cases where the HTML5 web interface loads but the connection to the server fails.

This is required for all WebRTC applications, especially since Selkies-GStreamer is self-hosted, unlike other proprietary services that provide a TURN server as part of the infrastructure.

In most cases when either of your server or client does not have a restrictive firewall, the default Google STUN server configuration will work without additional configuration. However, when connecting from networks that cannot be traversed with STUN, a TURN server is required.

Open Relay is a free TURN server instance that may be used for personal testing purposes, but may not be optimal for production usage.

While the Open Relay TURN server is the default when no TURN server is set, because there is only one server location, any connection with the type `relay` will add substantial latency as well as stutters to your connections.

For self-hosting with restricted host networks, the Oracle Cloud Free Tier Arm Compute Instance provides up to 10 TB Outbound Data Transfer every month, which accounts to a total of nearly 30 mbps bandwidth even when Selkies-GStreamer is utilized 24/7. You may configure coTURN with any computer, server, cloud service, or virtual machine you want. Make sure to use a location that is as close as possible to the web client or the host server.

Otherwise, Cloudflare and Twilio also provide paid TURN server services.

Selkies-GStreamer with TURN Server Credentials

Configure coTURN or any other TURN server by reading the later sections.

After configuring your TURN server, the following options are required to be used with Selkies-GStreamer:

- TURN server hostname (command-line option `--turn_host=` or environment variable `SELKIES_TURN_HOST`, not used with `--rtc_config_json` JSON file authentication or `--turn_rest_uri` TURN REST API authentication)
- TURN server port (command-line option `--turn_port=` or environment variable `SELKIES_TURN_PORT`, not used with `--rtc_config_json` JSON file authentication or `--turn_rest_uri` TURN REST API authentication)
- You may set the command-line option `--turn_protocol=tcp` or the environment variable `SELKIES_TURN_PROTOCOL` to `tcp` if you are unable to open the UDP listening port to the internet for the coTURN container, or if the UDP protocol is blocked or throttled in your client network.
- You may also set the command-line option `--turn_tls=true` or the environment variable `SELKIES_TURN_TLS` to `true` if TURN over TLS/DTLS was properly configured with a certificate and key combination from a legitimate certificate authority such as ZeroSSL or Let's Encrypt with a valid hostname which resolves to the TURN server.
- **One of the methods in the TURN Server Authentication Methods section for authentication are required.**

TURN Server Authentication Methods

There are currently four different supported TURN server authentication methods, in the order of priority:

- Using the JSON configuration file authentication method with the `selkies-gstreamer --rtc_config_json=` option or the `SELKIES_RTC_CONFIG_JSON` environment variable. Selkies-GStreamer probes this file periodically, so it will automatically update the TURN authentication credentials if the JSON file is updated. **All other STUN/TURN credentials are overridden if this file exists.**
- Using the TURN REST API authentication method with the the `selkies-gstreamer --turn_rest_uri=` option or the `SELKIES_TURN_REST_URI` environment variable. Selkies-GStreamer probes this REST API endpoint periodically, so it will automatically update the TURN authentication credentials. Consult the **TURN-REST** section for more details of this authentication method. **All other STUN/TURN credentials below are overridden if this option is provided and is valid.**
- **Note that the below two methods are only safe when the Selkies-GStreamer user also has legitimate control of the TURN server. Otherwise, if you maintain a multi-user environment, you are looking for the TURN REST API authentication method, right above.**
- Using traditional long-term credential authentication with a fixed username and password combination using the `selkies-gstreamer --turn_username= --turn_password=`, or both environment variables `SELKIES_TURN_USERNAME` and `SELKIES_TURN_PASSWORD`.

- Directly inputting the time-limited TURN shared secret credential using the `selkies-gstreamer --turn_shared_secret=` option or the `SELKIES_TURN_SHARED_SECRET` environment variable.

The `--turn_host=` and `--turn_port=` options or the equivalent environment variables `SELKIES_TURN_HOST` and `SELKIES_TURN_PORT` are required for traditional long-term credential authentication or time-limited TURN shared secret credential authentication, but are NOT used for the JSON configuration file authentication method and TURN REST API authentication method.

Moreover, the `--stun_host=` and `--stun_port=` STUN server options, defaulting to `stun.l.google.com` and `19302` (the default Google STUN servers), take priority for for traditional long-term credential authentication or time-limited TURN shared secret credential authentication, but are NOT used for the JSON configuration file authentication method and TURN REST API authentication method. The JSON configuration file authentication method and TURN REST API authentication method takes the first STUN server received from their respective configuration file or API.

If you are using Selkies-GStreamer in a private network without access to the internet, you must update the `--stun_host=` and `--stun_port=` options to your local STUN/TURN server if using traditional long-term credential authentication or time-limited TURN shared secret credential authentication. Else, you may have WebRTC connectivity issues. The JSON configuration file authentication method and TURN REST API authentication method uses the first STUN server received from the configuration file or API.

coTURN

An open-source TURN server for Linux or UNIX-like operating systems that may be used is coTURN, available in major package repositories or as an official container `coturn/coturn:latest`.

The Selkies-GStreamer coTURN image `ghcr.io/selkies-project/selkies-gstreamer/coturn:main` is also included in this repository, and may be used to host your own STUN/TURN infrastructure. As this image contains additional features for identifying the external server IP in cloud environments, usage of this container is recommended.

Pion TURN's `turn-server-simple` executable or eturnal are recommended alternative TURN server implementations that support Windows as well as Linux or MacOS. STUNner is a Kubernetes-native STUN and TURN deployment if Helm is possible to be used.

Install and run coTURN on self-hosted standalone machines, cloud instances, or virtual machines

It is possible to install coTURN on your own server or PC from a package repository, as long as the listening port and the relay ports may be opened.

1. Installation for Ubuntu or Debian-based distributions (available in EPEL for CentOS/RHEL):

```
sudo apt-get update && sudo apt-get install --no-install-recommends -y coturn
```

2. For self-hosted standalone coTURN servers, a minimal barebones configuration for `/etc/turnserver.conf` is available below, where options are also all available as command-line options (check the coTURN Example Configuration for more information)**:**

```
listening-ip=0.0.0.0
listening-ip=::

listening-port=3478

# Choose one (mandatory):

# When using static auth secret and/or TURN REST API authentication:
# use-auth-secret
# static-auth-
secret=n0TaRealCoTURNAuthSecretThatIsSixtyFourLengthsLongPlaceholdPlace

# When using traditional long-term credential authentication:
# lt-cred-mech
# user=username1:password1
# user=username2:password2

realm=example.com

# Specify minimum and maximum ports to allocate for coTURN TURN relay ports
min-port=49152
max-port=49172

log-file=stdout
pidfile=/tmp/turnserver.pid
userdb=/tmp/turnserver.db

# Certificate paths if TURN over TLS is to be used
# cert=/etc/coturn_tls.crt
# pkey=/etc/coturn_tls.key

# Prometheus statistics
prometheus

# Add `allow-loopback-peers` if coTURN and Selkies-GStreamer are on the same
node
no-software-attribute
no-rfc5780
no-stun-backward-compatibility
response-origin-only-with-rfc5780
```

`/etc/turnserver.conf` must have the lines `listening-ip=0.0.0.0` and `realm=example.com` (change the realm as appropriate), and either the lines `use-auth-secret` and `static-auth-secret=n0TaRealCoTURNAuthSecretThatIsSixtyFourLengthsLongPlaceholdPlace`, or the lines `lt-cred-mech` and `user=username1:password1` are required.

For single-user environments, traditional long-term credential authentication is the easiest, but multi-user environments likely need TURN REST API authentication with a static auth secret.

Ports specified in `listening-port=`, `min-port=` and `max-port=` must be opened.

It is strongly recommended to set the `min-port=` and `max-port=` parameters which specifies the relay ports (all ports between this range must be open). Add the line `no-udp-relay` if you cannot open the UDP `min-port=` to `max-port=` port ranges, or the line `no-tcp-relay` if you cannot open the TCP `min-port=` to `max-port=` port ranges. Note that the `no-udp-relay` option may not be supported with web browsers and may lead to the TURN server not working.

The `cert=` and `pkey=` options are required for using TURN over TLS/DTLS, but are otherwise optional. They should lead to the certificate and the private key from a legitimate certificate authority such as ZeroSSL or Let's Encrypt with a valid hostname which resolves to the TURN server.

3. Enable the coTURN service:

Modify the file `/etc/default/coturn`:

```
sudo nano /etc/default/coturn
# Or
sudo vi /etc/default/coturn
```

Uncomment the following line:

```
# From
#TURNSERVER_ENABLED=1
# To
TURNSERVER_ENABLED=1
```

After this is configured, enable the coTURN systemd service:

```
sudo systemctl enable coturn
sudo systemctl restart coturn
```

Consult the coTURN Documentation and Example Configuration for specific usage directions.

Deploy coTURN with Docker®

The coTURN Container is a reference container which provides the coTURN TURN server. Other than options including `-e TURN_SHARED_SECRET=`, `-e TURN_REALM=`, `-e TURN_PORT=`, `-e`

`TURN_MIN_PORT=`, and `-e TURN_MAX_PORT=`, add more command-line options in `-e TURN_EXTRA_ARGS=`.

Read the coTURN section to get started.

Alternatively, using the official coTURN container (`--min-port` is at least `49152` and `--max-port` is at most `65535`):

For time-limited shared secret TURN authentication:

```
docker run --name coturn --rm -d -p 3478:3478 -p 3478:3478/udp -p 65500-65535:65500-65535 -p 65500-65535:65500-65535/udp coturn/coturn -n --listening-ip="0.0.0.0" --listening-ip="::" --realm=example.com --external-ip="$(curl -fsSL checkip.amazonaws.com 2>/dev/null || hostname -I 2>/dev/null | awk '{print $1; exit}' || echo '127.0.0.1')" --min-port=65500 --max-port=65535 --use-auth-secret --static-auth-secret=n0TaRealCoTURNAuthSecretThatIsSixtyFourLengthsLongPlaceholdPlace
```

For legacy long-term TURN authentication:

```
docker run --name coturn --rm -d -p 3478:3478 -p 3478:3478/udp -p 65500-65535:65500-65535 -p 65500-65535:65500-65535/udp coturn/coturn -n --listening-ip="0.0.0.0" --listening-ip="::" --realm=example.com --external-ip="$(curl -fsSL checkip.amazonaws.com 2>/dev/null || hostname -I 2>/dev/null | awk '{print $1; exit}' || echo '127.0.0.1')" --min-port=65500 --max-port=65535 --lt-cred-mech --user=username1:password1
```

The relay ports and the listening port must all be open to the internet.

If the TURN relay port range is wide, it may take a very long time for the containers to start up. Simply using `--network=host` instead of specifying `-p 65500-65535:65500-65535` and `-p 65500-65535:65500-65535/udp` can also be plausible.

Modify the relay ports `-p 65500-65535:65500-65535` and `-p 65500-65535:65500-65535/udp` combined with `--min-port=65500` and `--max-port=65535` after `-n` as appropriate (at least two relay ports are required per connection).

In addition, use the option `--no-udp-relay` after `-n` if you cannot open the UDP `--min-port=` to `--max-port=` port ranges, or `--no-tcp-relay` after `-n` if you cannot open the TCP `--min-port=` to `--max-port=` port ranges. Note that the `--no-udp-relay` option may not be supported with web browsers and may lead to the TURN server not working.

Consult the coTURN Documentation and Example Configuration for specific usage directions.

TURN over TLS

In both types of containers, the `--cert=` and `--pkey=` options are required for using TURN over TLS/DTLS, but are otherwise optional. They should lead to the certificate and the private

key from a legitimate certificate authority such as ZeroSSL or Let's Encrypt with a valid hostname which resolves to the TURN server.

Provide the certificate and private files to the coTURN container with `-v /my_local_path/coturncert.crt:/etc/coturn_tls.crt -v /my_local_path/coturnkey.key:/etc/coturn_tls.key` (specified paths are an example), then add the options `-e TURN_EXTRA_ARGS="--cert=/etc/coturn_tls.crt --pkey=/etc/coturn_tls.key"` for the Selkies coTURN Container and use the options `--cert=/etc/coturn_tls.crt --pkey=/etc/coturn_tls.key` for the official coTURN container.

Deploy coTURN With Kubernetes

Before you continue, STUNner is a pretty good method to deploy a TURN or STUN server on Kubernetes if you are able to use Helm.

You are recommended to use a `ConfigMap` for creating the configuration file for coTURN.

Consult the coTURN Documentation and Example Configuration for specific usage directions.

Other instructions for the configuration are same as the Standalone Deployment.

Use `Deployment` (with `affinity.podAntiAffinity` to distribute the coTURN server to multiple nodes) or `DaemonSet`, then use `containerPort` and `hostPort` under `ports:` to open the listening port you set in `/etc/turnserver.conf` with `listening-port=` (both TCP and UDP).

Then, all ports between `min-port=` and `max-port=` that you set in `/etc/turnserver.conf` must also be opened, but this may be skipped if `hostNetwork: true` is used instead.

The relay ports and the listening port must all be open to the internet.

Add the line `no-udp-relay` if you cannot open the UDP `min-port=` to `max-port=` port ranges, or the line `no-tcp-relay` if you cannot open the TCP `min-port=` to `max-port=` port ranges. Note that the `no-udp-relay` option may not be supported with web browsers and may lead to the TURN server not working.

In the configuration, under `args:`, set `-c /etc/turnserver.conf` and use the `coturn/coturn:latest` image.

The `cert=` and `pkey=` options are required for using TURN over TLS/DTLS, but are otherwise optional. Use cert-manager to issue a valid certificate from a legitimate certificate authority such as ZeroSSL or Let's Encrypt with a valid hostname which resolves to the TURN server, and mount to the paths which lead to the `cert=` and `pkey=` options.

Consult the coTURN Documentation and Example Configuration for specific usage directions.