

librice

The TURNing point 2025 GStreamer conference

Matthew Waters

23 October 2025

Who Am I?

GStreamer developer and maintainer for over a decade

WebRTC, Vulkan, OpenGL

Short overview of ICE

1. Gather candidates (Host, Server reflexive, Relayed)
2. Connectivity checks
3. Send/Receive data

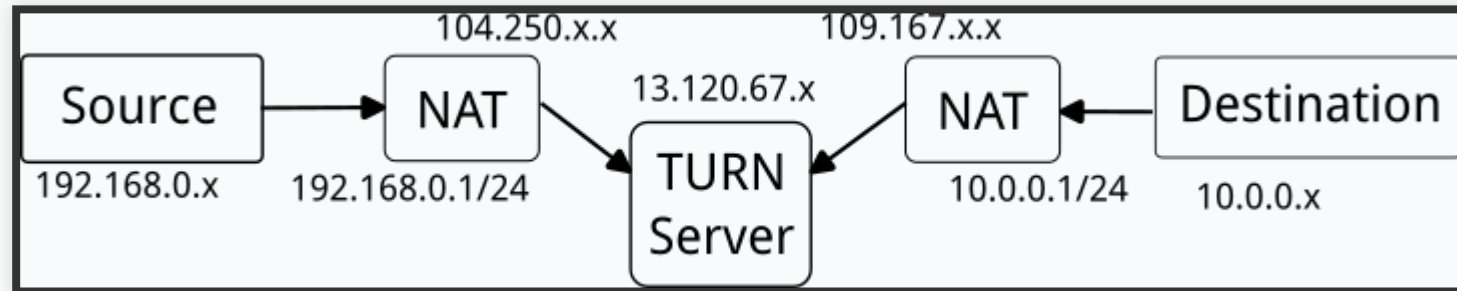
See also [ICE: how to find your way through the internet](#) from the GStreamer conference 2023.

TURN (relayed)

If direct connection is not possible, then data needs to be relayed through an intermediary.

TURN (relayed)

If direct connection is not possible, then data needs to be relayed through an intermediary.



TURN (relayed)

Four relevant RFCs:

TURN (relayed)

Four relevant RFCs:

- [RFC5766](#) (original TURN RFC)

TURN (relayed)

Four relevant RFCs:

- [RFC5766](#) (original TURN RFC)
- [RFC6062](#) (TCP allocation)

TURN (relayed)

Four relevant RFCs:

- [RFC5766](#) (original TURN RFC)
- [RFC6062](#) (TCP allocation)
- [RFC6156](#) (IPv6 allocations)

TURN (relayed)

Four relevant RFCs:

- [RFC5766](#) (original TURN RFC)
- [RFC6062](#) (TCP allocation)
- [RFC6156](#) (IPv6 allocations)
- [RFC8656](#) (dual IPv4/6 allocations on top of RFC5766/RFC6156, and DTLS)

TURN Allocation flow

Authentication uses the STUN long-term credential mechanism

TURN Allocation flow

Authentication uses the STUN long-term credential mechanism

1. Send unauthenticated ALLOCATE request.

```
Message(class: Request, method: 3(0x3: ALLOCATE), transaction:  
0x5865317f6e81ff1a653355bf, attributes: [13(0xd: Lifetime): '1800', 25(0x19:  
RequestedTransport): 17, 26(0x1a: DontFragment)])
```

TURN Allocation flow

Authentication uses the STUN long-term credential mechanism

1. Send unauthenticated ALLOCATE request.

```
Message(class: Request, method: 3(0x3: ALLOCATE), transaction:  
0x5865317f6e81ff1a653355bf, attributes: [13(0xd: Lifetime): '1800', 25(0x19:  
RequestedTransport): 17, 26(0x1a: DontFragment)])
```

2. Receive error response with a NONCE.

```
Message(class: Error, method: 3(0x3: ALLOCATE), transaction:  
0x5865317f6e81ff1a653355bf, attributes: [21(0x15: NONCE): n0TUegICDsqv108S,  
20(0x14: REALM): realm, 9(0x9: ERROR-CODE): 401 'Unauthorized'])
```

TURN Allocation flow

3. Resend ALLOCATE request with authentication parameters.

```
Message(class: Request, method: 3(0x3: ALLOCATE), transaction:  
0xee76e12628cad6a048e7c5a, attributes: [25(0x19: RequestedTransport): 17,  
13(0xd: Lifetime): '1800', 6(0x6: USERNAME): 'tuser', 20(0x14: REALM): realm,  
21(0x15: NONCE): n0TUegICDsqv108S, 8(0x8: MESSAGE-INTEGRITY):  
0x22d842ea213cd48dc4ec10045d08d43a410d7722])
```

TURN Allocation flow

3. Resend ALLOCATE request with authentication parameters.

```
Message(class: Request, method: 3(0x3: ALLOCATE), transaction:  
0xee76e12628cad6a048e7c5a, attributes: [25(0x19: RequestedTransport): 17,  
13(0xd: Lifetime): '1800', 6(0x6: USERNAME): 'tuser', 20(0x14: REALM): realm,  
21(0x15: NONCE): n0TUegICDsqv108S, 8(0x8: MESSAGE-INTEGRITY):  
0x22d842ea213cd48dc4ec10045d08d43a410d7722])
```

4. On success retrieve the lifetime.

```
Message(class: Success, method: 3(0x3: ALLOCATE), transaction:  
0xee76e12628cad6a048e7c5a, attributes: [22(0x16: XorRelayedAddress):  
127.0.0.1:33131, 13(0xd: Lifetime): '1800', 32(0x20: XOR-MAPPED-ADDRESS):  
192.168.20.41:48448, 8(0x8: MESSAGE-INTEGRITY):  
0x2655b5e6af641b2503e9d95693edcb09d1ecbeca])
```

TURN Allocation flow

3. Resend ALLOCATE request with authentication parameters.

```
Message(class: Request, method: 3(0x3: ALLOCATE), transaction:  
0xee76e12628cad6a048e7c5a, attributes: [25(0x19: RequestedTransport): 17,  
13(0xd: Lifetime): '1800', 6(0x6: USERNAME): 'tuser', 20(0x14: REALM): realm,  
21(0x15: NONCE): n0TUegICDsqv108S, 8(0x8: MESSAGE-INTEGRITY):  
0x22d842ea213cd48dc4ec10045d08d43a410d7722])
```

4. On success retrieve the lifetime.

```
Message(class: Success, method: 3(0x3: ALLOCATE), transaction:  
0xee76e12628cad6a048e7c5a, attributes: [22(0x16: XorRelayedAddress):  
127.0.0.1:33131, 13(0xd: Lifetime): '1800', 32(0x20: XOR-MAPPED-ADDRESS):  
192.168.20.41:48448, 8(0x8: MESSAGE-INTEGRITY):  
0x2655b5e6af641b2503e9d95693edcb09d1ecbeca])
```

There are many errors that can cause a failure e.g. allocation already exists, invalid credentials, invalid request, unsupported attributes in request or response, network timeout, etc.

TURN Allocation configuration

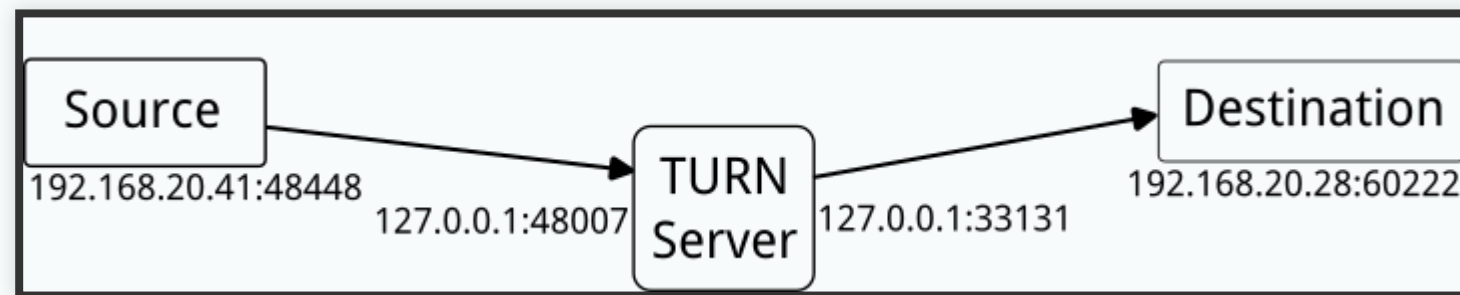
1. client $\leftrightarrow$ server
 - a. transport protocols (UDP/TCP)
 - b. encryption (TLS/DTLS)
 - c. address families (IPv4/6)
 - d. optional dual allocation (if supported)
2. server $\leftrightarrow$ peer
 - a. transport protocols (UDP/TCP)
 - b. address families (IPv4/6)

2^6 (64) different configurations for a single relay through a TURN

So, you have a TURN allocation

With ICE, your connection check table will look like:

ID:2	foundation:6:1	state:Succeeded	nom:false	con:true	priority:	14681343
ID:6	foundation:6:0	state:Succeeded	nom:false	con:true	priority:	14681343
ID:3	foundation:7:0	state:Succeeded	nom:false	con:true	priority:	14680831
ID:7	foundation:7:1	state:Succeeded	nom:false	con:true	priority:	14680831



After TURN Allocation

- Refreshing/Deleting
- Permissions
- Channels
- Send/Receive Data

Refreshing (and Deleting)

Send a REFRESH STUN request with the requested lifetime. Must be completed before expiry.

```
Message(class: Success, method: 4(0x4: REFRESH), transaction: ..., attributes: [13(0xd: Lifetime): '1800', 8(0x8: MESSAGE-INTEGRITY): ...])
```

Deleting an allocation requires sending a STUN REFRESH request with a lifetime of 0. Deleting an allocation removes all permissions/channels associated with that allocation.

Permissions

A permission controls whether a TURN allocation can send and receive data to a peer on behalf of the client. If there is no permission, then outgoing and incoming data will be ignored.

```
Message(class: Request, method: 8(0x8: CREATE_PERMISSION), transaction: ..., attributes:  
  [18(0x12: XorPeerAddress): 192.168.20.28:0, 6(0x6: USERNAME): 'tuser', 20(0x14:  
    REALM): realm, 21(0x15: NONCE): ..., 8(0x8: MESSAGE-INTEGRITY): ...])
```

Permissions have a fixed lifetime of 5 minutes and need to be refreshed before the lifetime expires.

Channels

A Channel is a more efficient encoding of data between the client and the server.

```
Message(class: Request, method: 9(0x9: CHANNEL_BIND), transaction: ..., attributes:  
[12(0xc: ChannelNumber): '16384', 18(0x12: XorPeerAddress): 10.0.0.3:9000, 6(0x6:  
  USERNAME): 'tuser', 20(0x14: REALM): realm, 21(0x15: NONCE): ..., 8(0x8: MESSAGE-  
  INTEGRITY): ...])
```

The Channel ID encodes the peer address to use.

Channels have a fixed lifetime of 10 minutes and need to be refreshed before the lifetime expires.

Send/Receive data

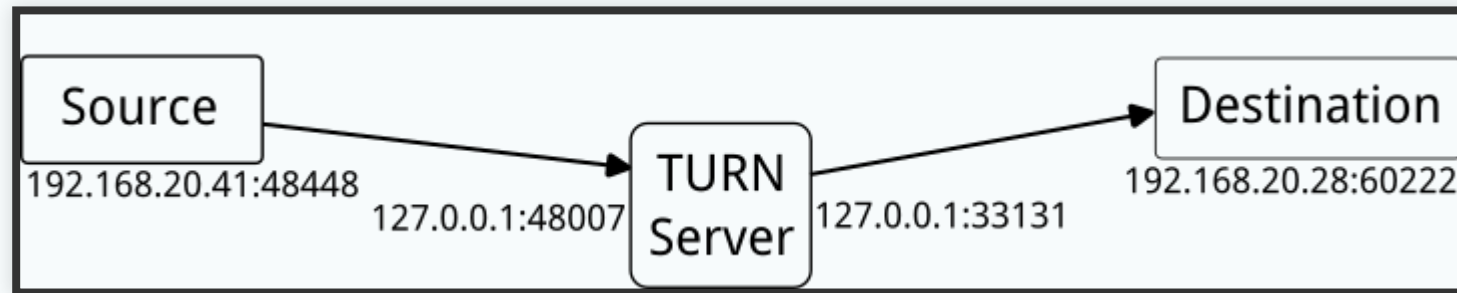
If there is a configured channel for the peer, then the TURN server will always send data to the client using the Channel.

The client can send data using the channel or with a STUN SEND indication.

If there is no channel, the TURN server will send a DATA indication to the client along with the peer's address.

Completed checks

ID:2	foundation:6:1	state:Succeeded	nom:false	con:true	priority:	14681343
ID:13	foundation:6:1	state:Succeeded	nom:true	con:true	priority:	14681343
ID:6	foundation:6:0	state:Succeeded	nom:false	con:true	priority:	14681343
ID:3	foundation:7:0	state:Succeeded	nom:false	con:true	priority:	14680831
ID:7	foundation:7:1	state:Succeeded	nom:false	con:true	priority:	14680831



TURN-TCP (RFC6062)

TURN-TCP (RFC6062)

Control TCP connection

Where all TURN control messages are sent. If the Control connection is closed, the allocation is determined to be deleted.

TURN-TCP (RFC6062)

Control TCP connection

Where all TURN control messages are sent. If the Control connection is closed, the allocation is determined to be deleted.

Data TCP connection

The application data travels over this connection as-is.

If either segment (client $\leftrightarrow$ server, or server $\leftrightarrow$ peer) is closed, the other segment must be closed.

librice

sans-IO networking library for handling ICE connections with a focus on testing, and reproducibility.

See also: [librice: a sans-IO networking library](#) from the GStreamer conference 2024.

TURN in `librice`

- New crates building on `stun-proto` and `stun-types`:
 - `turn-types` - STUN attributes and message identifiers for TURN
 - `turn-client-proto` - sans-IO protocol for a TURN client
 - `turn-server-proto` - sans-IO protocol for a TURN server

Transparently handled in `librice` once configured.

TURN in Librice details

Gathering:

TURN in Librice details

Gathering:

Involves setting up a TURN allocation with the appropriate configuration. e.g. TURN credentials, TURN server address, (D)TLS configuration, etc

TURN in Librice details

Connectivity checks

TURN in Librice details

Connectivity checks

Configures TURN permissions towards peers provided by remote candidates.

TURN in Librice details

Connectivity checks

Configures TURN permissions towards peers provided by remote candidates.

Regular connectivity checks and any further data is relayed through the TURN server.

TURN in Librice details

Allocation maintenance

TURN in Librice details

Allocation maintenance

Refresh the allocation, permissions, and channels.

Other Librice news

1. WIP GStreamer branch for webrtcbin:
<https://gitlab.freedesktop.org/ystreet/gstreamer/-/tree/webrtc-rice>
2. WIP WebKit branch for WPE and GTK ports:
<https://github.com/WebKit/WebKit/pull/50740>. Also see philn's talk later.
3. [A new matrix chat room](#)

[Librice]API

`rice-proto` can now be used from multiple libraries/applications.

Requires access through the C API. In rust, this is the new `rice-c` crate.

Thanks

- nl.net: <https://nlnet.nl/project/librice/>
- <https://crates.io/crates/rice-proto>
- ystreet00 on #gstreamer on OFTC
- <https://gitlab.freedesktop.org/ystreet>
- ystreet00@floss.social on mastodon

