go get RIPT

@IETF 107
goRIPT

https://github.com/WhatIETF/goRIPT

• golang RIPT client and server. (WIP)

• Scope: Experiment if the spec **as it stands** is implementable for real-time media use-cases

  • H3 for media transport

    • Use PUT/GET for audio media send and receive

• Cloud deploy the server

  • Verify connectivity for H3 and elasticity

• Subject Quality of Experience - latency

• Experiment with signaling protocol (trunkDiscovery, handler registration, advertisement for media capability)

• Influence Spec via running code eventually
VPC

Public Subnet

EC2 Public IP

RIPT Server
Security Gateway

Domain/DNS zone
A-record -> public IP address

IGW

Signaling/Media over H3

RIPT Client
quic-go
PortAudio

quirent
PortAudio
goRIPT Media Flow

**GET** `carrier.com/well-known/rtpt/v1/ providertgs`
- Trunk Group Discovery
- Default Trunk group details

**POST** `carrier.com/well-known/rtpt/v1/ providertgs/<tg>/handlers`
- Handler Registration
  - Alice provides handlerId, media cap advertisement

**200 OK, HandlerUri**

**GET** `carrier.com/well-known/rtpt/v1/ providertgs`
- Trunk Group Discovery
- Default Trunk group details

**POST** `carrier.com/well-known/rtpt/v1/ providertgs/<tg>/calls`
- Call Setup:
  - Alice provides handlerUri, Destination Address (hardcoded)

**200 OK, CallUri, Client/Server Directives**

**POST** `carrier.com/well-known/rtpt/v1/ providertgs/<tg>/calls`
- Call Setup:
  - Bob provides handlerUri, Destination Address (hardcoded)

**200 OK, CallUri, Client/Server Directives**

**PUT** `carrier.com/well-known/rtpt/v1/ providertgs/<tg>/calls/<callId>/media`
- Media Send:
  - Alice sends media chunk

**200 OK**

**GET** `carrier.com/well-known/rtpt/v1/ providertgs/<tg>/calls/<callId>/media`
- Media Receive:
  - Bob pulls media chunk
What’s in goRIPT today

• Media Byways*
  • H3 for pushing and pulling opus audio @ 48khz/60 ms samples
  • Unidirectional media (sendonly and recvonly)
  • Media is pulled **synchronously per media chunk** (no pipelining of pull supported yet)
  • No datagram support yet.

• Signaling Features (partially supported)
  • Trunk Discovery API, Handler Registration API, Calls API.
  • Rudimentary capability advertisement and negotiation.

• Not supported : Call State Management via Signaling Byways, Server Peering, Passport/Certificate Enrollment ..
### goRIPT Server SLOC breakdown

<table>
<thead>
<tr>
<th>Task</th>
<th>SLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trunk Discovery</td>
<td>~35</td>
</tr>
<tr>
<td>Handler Registration</td>
<td>~60</td>
</tr>
<tr>
<td>Media Handling</td>
<td>~60</td>
</tr>
<tr>
<td>Media Distribution</td>
<td>~150</td>
</tr>
<tr>
<td>API DTO</td>
<td>~150</td>
</tr>
<tr>
<td>Stack Setup Framework, Miscellaneous</td>
<td>~400</td>
</tr>
</tbody>
</table>

**goRIPT client**: ~800 SLOC, **goRIPT Server**: ~1100 SLOC
Latency and Bandwidth Estimates

- **Bandwidth**
  - Opus @16khz, 20ms frames ==> ~48kbps

- **Latency (Sender to Receiver)**
  - Test Setup: Sender and Receiver (Calgary), Sever (AWS Oregon)
  - Network Latency from Sender to Receiver via Server ==> 35-50 ms
  - Mic to Speaker Latency (end to end) latency: ~350 ms
  - Mic to Speaker Latency for WebEx ==> ~350 ms
Credits ...

- quic-go: H3 stack in golang
  - https://github.com/lucas-clemente/quic-go

- TLS1.3 Syntax Library for binary encoding of the media chunk
  - https://github.com/bifurcation/mint

- Go PortAudio
  - https://github.com/gordonklaus/portaudio
“It’s never the end ...”
Media Chunk Format
(Binary encoded using TLS1.3 Syntax)
Signaling Messages & Framing

```go
type TrunkGroupInfo struct {
    Uri string
}

type TrunkGroupsInfoMessage struct {
    TrunkGroups []TrunkGroupInfo
}

type HandlerRequest struct {
    HandlerId string `json:"handler-id"
    Advertisement string `json:"advertisement"
}

type HandlerResponse struct {
    Uri string `json:"uri"
}

type CallRequest struct {
    HandlerUri string `json:"uri"
    Destination string `json:"destination"
}

type CallResponse struct {
    CallUri string `json:"uri"
    ClientDirective Directive `json:"clientDirectives"
    ServerDirective Directive `json:"serverDirectives"
}
```

Trunk Discovery

Handler Registration

Calls