Modular Media Stack

IETF-101 London March 2018
draft-jennings-dispatch-new-media-00

fluffy@cisco.com

V1

Goals

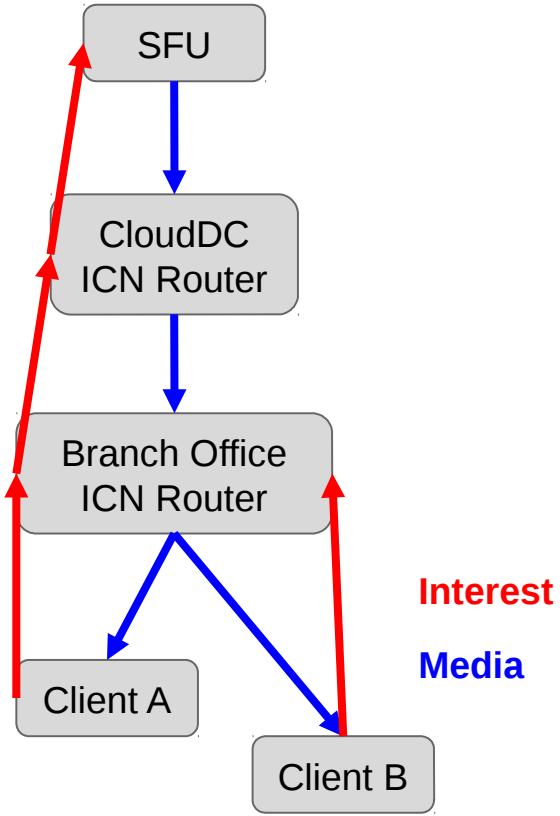
- Take a blank sheet of paper approach to rethinking what we want to be able to do with interactive media and how to do it
- Big stuff that is hard to do today
  - Scale with SDN, VPP, or ICN
  - Asymmetric media
  - End to End encryption by default
  - Pluggable congestion control, encryption, and codecs
  - Fast media setup
  - Simplify and improve ability to test and operate
- Many small things that are not easy to fix
  - Eliminate the problems with ROC in SRTP
  - Less codec negotiation failures

Example Simplification - STUN

- STUN today is used for 3 things for media: finding IP address of NAT, connectivity checks in ICE, and ongoing consent in WebRTC media

    a. For the first, Imagine an server that accepts QUIC connections, sends the client's IP address in binary followed by port back to the client then closes the connection. <- that could be the whole spec
    b. For the second, all we need is way to detect if a connection happened and verify we connected to the correct endpoint. STUN is way overkill for that. Shorter message use less bandwidth, so can send faster, so can setup call faster
    c. For the third, if we used a transport like QUIC, it would take care of keep alives

Information Centric Networks (or multicast)



Client express interest in a particular conferences

They receive the media packets for that conferences

ICN aware routers can aggregate requests and receive just one copy of a media packet but forward it to two clients

For this to work, we probably need to do encryption differently than its current done

Software Defined Networking

With SDN we can program a router to look at some header bits in the IP packet and then decide where to forward it

Could a conference bridge or TURN server be build so that a SDN controlled router handled most of the packets?

Probably Yes if we organized the bits in the right way.

Vector Packet Processors

fd.io has a router getting 1 Tbps using an off the shelf high end intel PC.

> To put that in perspective, that is like sending order 1,000,000 HD video streams from a conference bridge running on a single PC

The information processing of what is being done in the routing a packet is very similar to what a conference bridge does with an RTP packet. The semantic content is very similar.

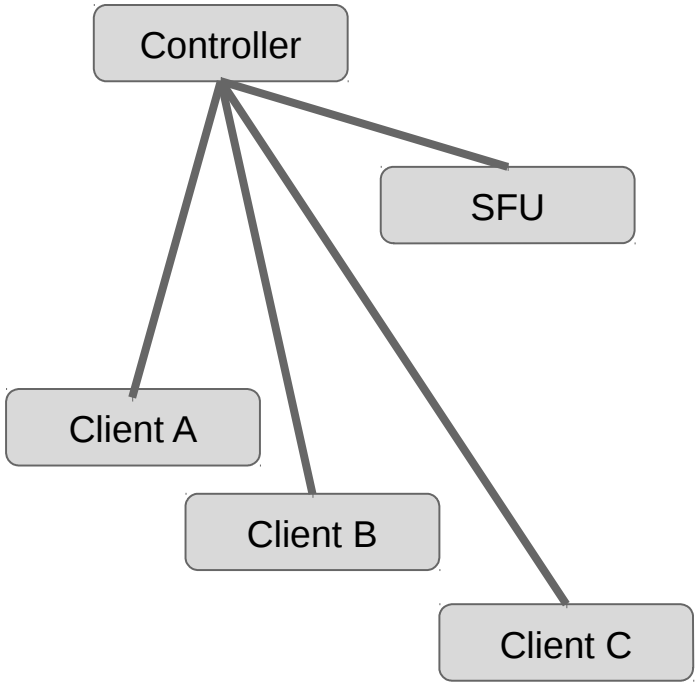
Unfortunately with RTP, the layout of the bits in the RTP packet does not seem to facilitate this high speed processing

Relax constraints - example TURN

Allow TURN to forward any inbound packets it received even if a permission is not installed (this would greatly simply TURN)

Or perhaps only allow it to forward inbound connectivity checks which balances the firewall like behavior with performance and simplicity (thanks EKR for this idea)

Controller Based Architectures



Most “calls” or “conference calls” have a controller, often in the cloud or a corp data center, that handles the negotiation with all the clients in the call

Having the controller know what each endpoint can do, then telling all endpoints to go do it is what happens

Offer / Answer, largely designed for 2 party peer 2 peer calls, is an awful match for this architecture

Style of specification

A document that covers the broad discussion, current state of thinking, alternative proposals, and decisions made and perhaps why. This is purely for people doing the work and is throw away once work is done

Small separable protocol specifications. That gives the high level view of the protocol and define the semantics (think boxes in arrows). Does not define the details of the bits on the wire. Think of this spec more like what you might find a wikipedia article about the protocol

A single reference implementation in github that defines the details of the protocol. And test for that code that end up testing the specification.

Next steps:

Start mail list, github, and experimental code to discuss:

1. Controller based architectures and the abstract APIs they need to control the media stack
2. Refactoring STUN, TURN, ICE
3. Separate RTP into an app that runs on top of a transport and refactor for ICN, SDN, and VPP
4. Sort out what transport and congestion control RTP needs
5. Sort out various uses of RTCP and find appropriate solution for each one
6. Consider if theses ideas can be done as extensions to existing protocols or if they would need to be new version

Is there interest at IETF?