Real Time Internet Peering Protocol*

draft-rosenbergjennings-dispatch-ripp-03

IETF 106 – Nov 18, 2019

J. Rosenberg Five9

C. Jennings Cisco

A. Minnesale SignalWire FreeSWITCH

J. Livingood Comcast

J. Uberti Google

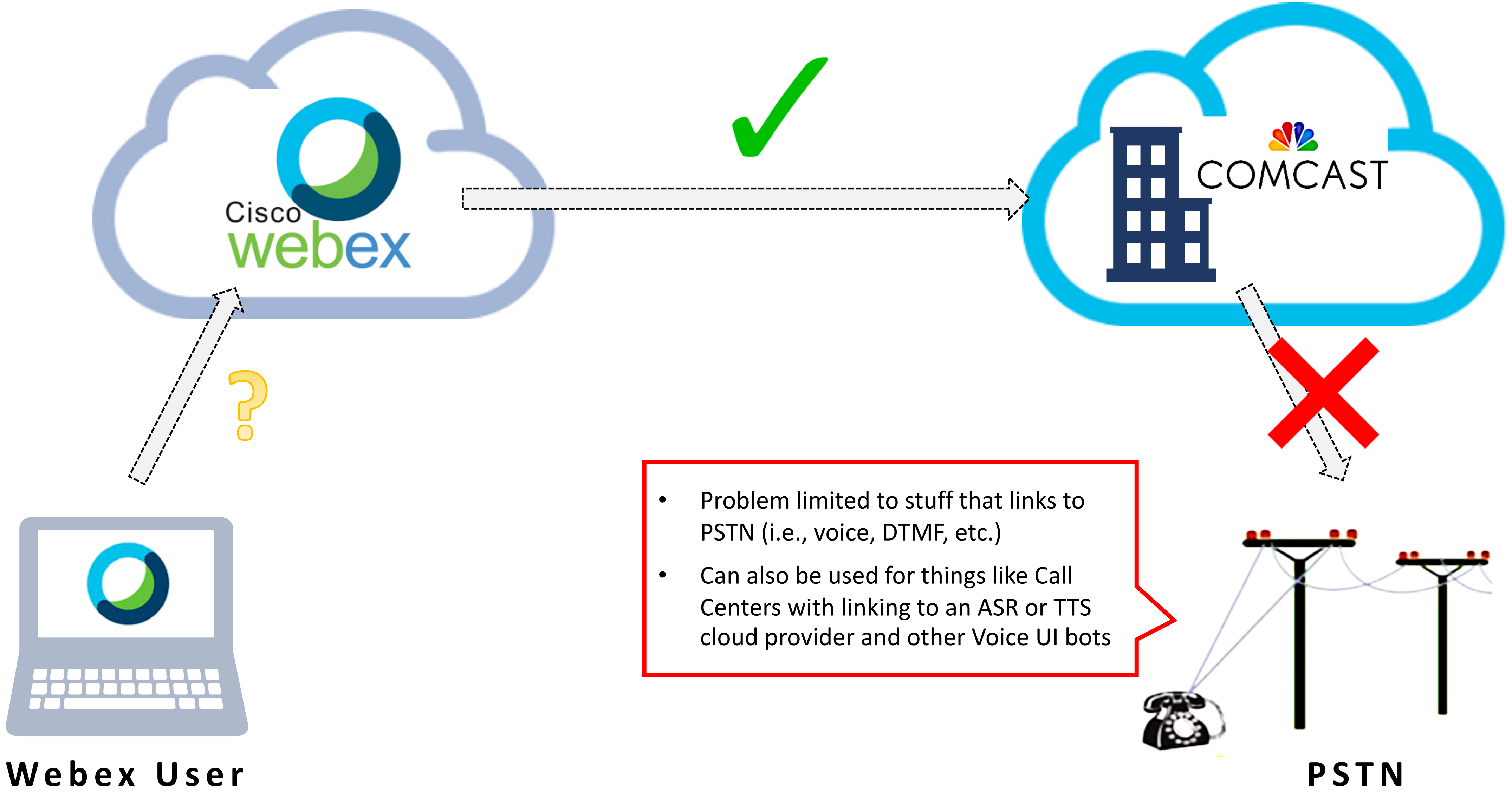
V10

* Five9 may have IPR on this draft: https://datatracker.ietf.org/ipr/3643/

Goals for Today

- Share the problems we’re trying to solve
- Help scope the problem and range of solutions
- Technical details aren’t important right now

RIPP Problem Space



- Problem limited to stuff that links to PSTN (i.e., voice, DTMF, etc.)
- Can also be used for things like Call Centers with linking to an ASR or TTS cloud provider and other Voice UI bots

Why don’t we just stick with what we already have (SIP + SDP + RTP)?

A cloud service like Webex can use existing cloud services for all of the following:

- Autoscaling
- Load balancing
- High reliability
- API gateways
- DDos mitigation tools
- Service Mesh (e.g., Istio)
- APM
- Tools the workforce already knows how to use

We *could* build all of these in a way that achieves all of this, but it would be:

- More work
- Harder to deploy
- Harder to use
- Sub-optimal because wouldn’t leverage the vast changes that have occurred to support HTTP-based APIs

Use Cases

- Add browser voice customer care to eCommerce sites
  - WebRTC solves browser to web servers, but we need to trunk the call to the contact center from there
  - This requires softswitches, SBCs, and special network considerations today (high complexity and not done often or broadly)
  - With RIPP, it is “just another web app”
- Telecom apps on web PaaS
  - AWS, Azure, Google – increasing sets of services make it very easy to deploy and run non real-time apps (e.g., Google AppEngine, HTTP LB, Istio on GKE, APIGee, etc)
  - They *could* modify code to ALSO work for SIP, but they haven’t and likely won’t
  - **Goal**: make it so that once they go HTTP3, real-time will just work

Use Cases

- Increase reliability SaaS to trunking provider calls
  - Webex to Comcast; Five9 to Tata; RingCentral to Verizon
  - Difficult to achieve hitless upgrades – server restarts cause calls to drop
  - Underlying VM restarts & migrations cause server restarts and dropped calls
  - Cluster expansion / contractions difficult – no way to drain and migrate calls reliably and consistently
  - Shouldn’t it be as easy to achieve hitless upgrades, handle VM restarts & migrations, and deal with automated cluster expansion / contraction as it is today for web apps??

Use Cases

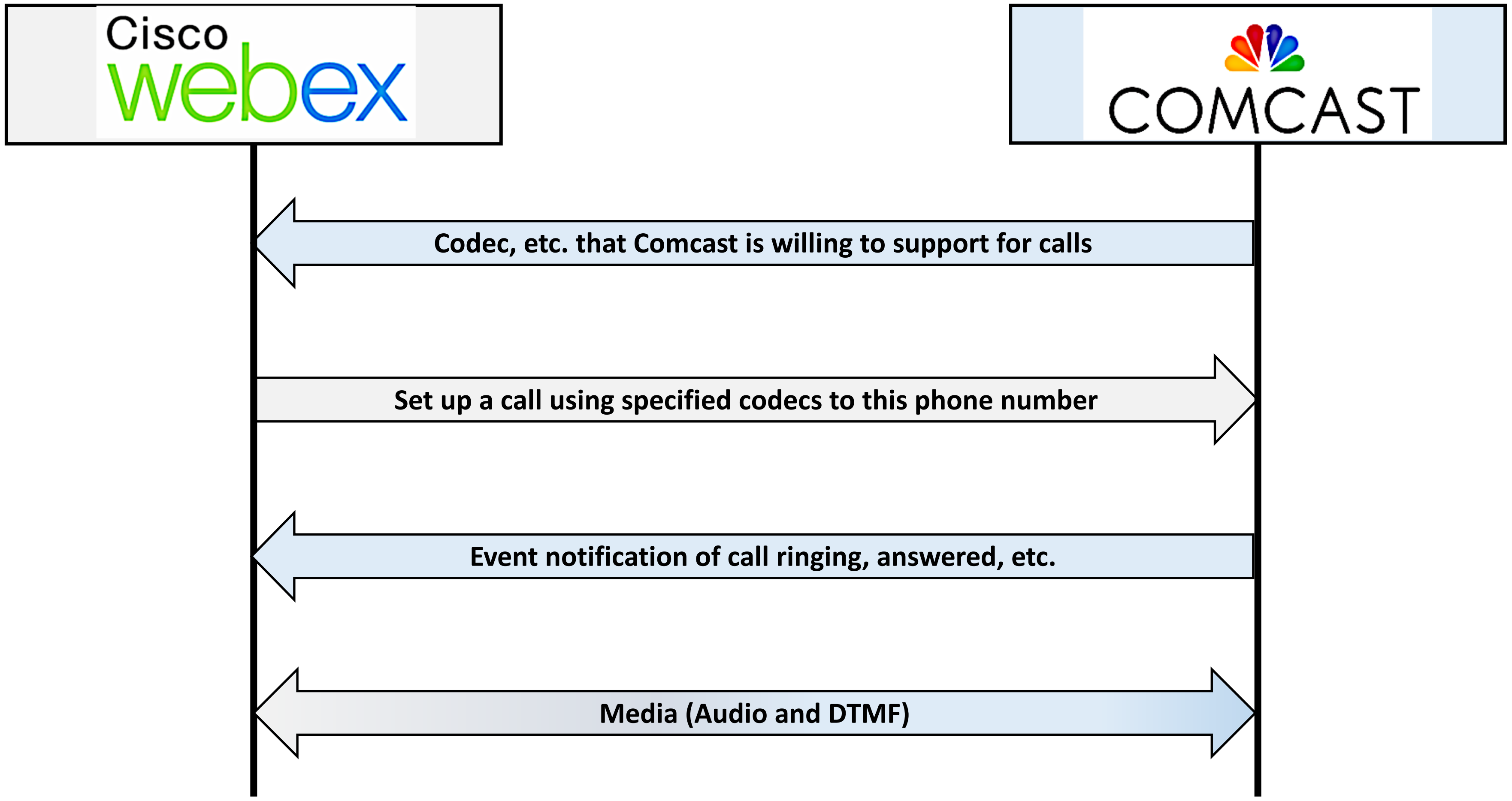
- Enterprise Voice Trunking
  - Today, SIP trunking from enterprises to trunking providers is very popular, but not for the faint of heart IT guy
  - Doesn’t work for shops that are otherwise all-cloud based
  - Specialized skill sets
  - Slow provisioning, frequent inter-op issues
  - Shouldn’t it be that It takes a few clicks and a few seconds to OAuth enterprise apps into cloud web apps? Why can’t we do the same for calling?

Technical Details

Really just a starting point to help folks understand what we had in mind

General Information Flow

Ignore what protocols are used and just look at data for a call from Webex to PSTN…


Cisco webex
COMCAST
Codec, etc. that Comcast is willing to support for calls
Set up a call using specified codecs to this phone number
Event notification of call ringing, answered, etc.
Media (Audio and DTMF)

Capabilities

- Capabilities allow Comcast to indicate what codecs it supports and other relatively static information about system capabilities
  - These do not change frequently – reasonable to think of as updating on perhaps a daily basis, but they do *not* contain per call information
  - Capabilities could include:
    - Supported codecs
    - Max bitrate & max sample rate
    - Force use of CBR (constant bit rate for the security crazy)
    - A few more weird things for advanced use-cases

Advertisement

- The calling side (Webex in this case) has the capabilities of the other side (in this case, Comcast). The calling side creates an Advertisement of all the exact details for setting up the call. These details must be consistent with the capabilities from the other side. The advertisement is sent to Comcast and they can **only** accept or reject it. There is no offer / answer like negation.
  - Information in advertisement:
    - Number to call
    - Caller ID of caller

Caller ID

We use (drum roll please)…

Passport

Protocol Mapping

- All the authors are big believers that:
  - In the near future, it will be possible to do unreliable data over QUIC
  - HTTP/3 over QUIC is a good idea
- Currently mapped to multiple HTTP requests with simple API

Get Auth token: https://ripp.example.com/trunks/{trunkID/consumerTrunk

Get Advertisement of capabilities for this trunk: https://ripp.example.com/trunks/{trunkID/capAdv

Get ongoing calls on this trunk: https://ripp.example.com/trunks/{trunkID/calls

Get the previous event (i.e., ringing, answer ….): https://ripp.example.com/calls/{callID/prevEvent

Wait for next event and return it: https://ripp.example.com/calls/{callID/event

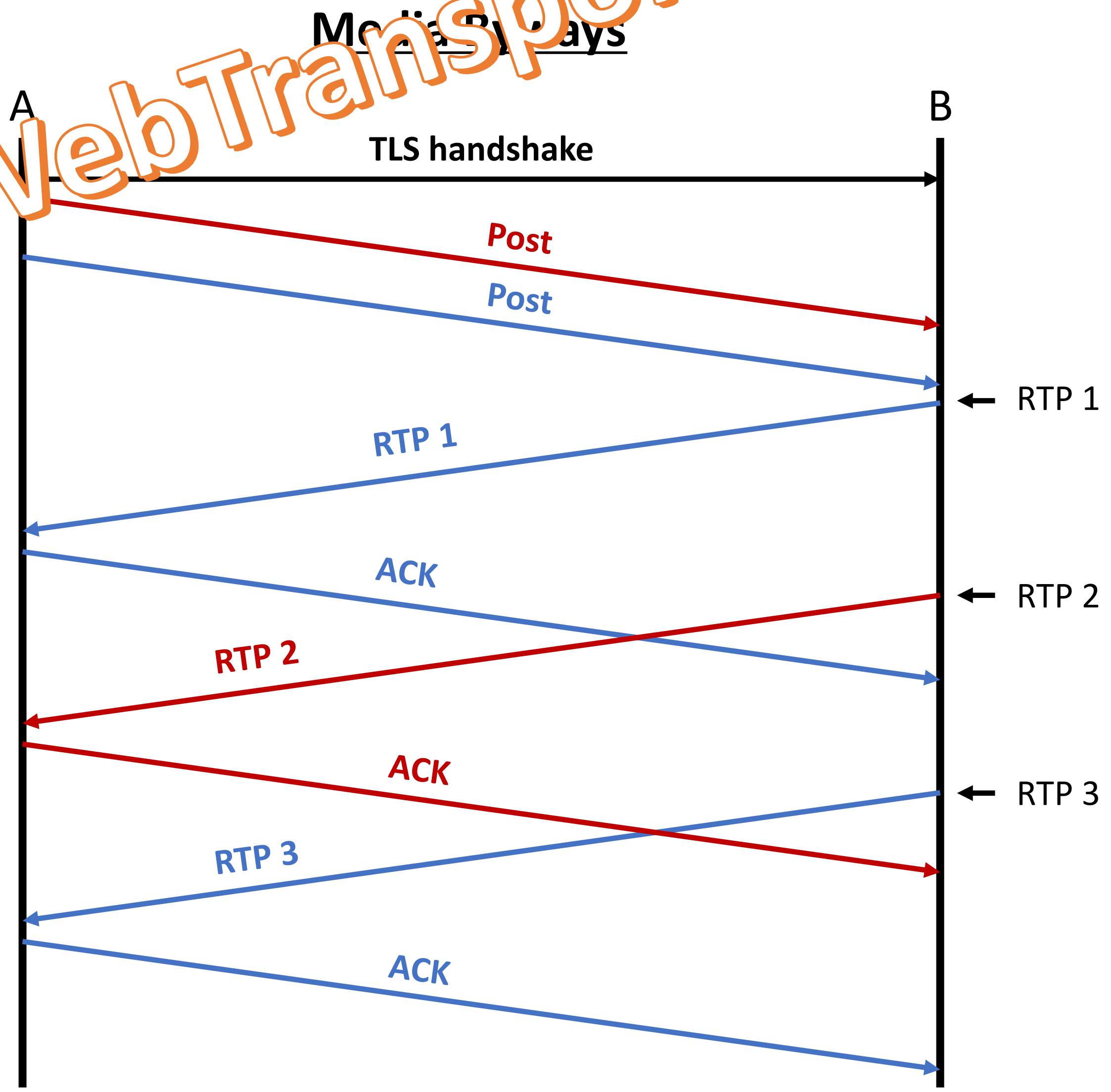
Flow of media chunks in forward direction: https://ripp.example.com/calls/{callID}/media-forward

Flow of media chunks in reverse direction: https://ripp.example.com/calls/{callID/media-reverse

Media Byways

Replace with WebTransports ?

- We are sending media over the HTTP/3 connection
  - Many ways to do this— still need to figure out wh[illegible] works best
- Draft currently purposes se[illegible] simultaneous HTTP long poll type req[illegible]ests that are multiplex in one HTTP/3 session. The sender sends media on whichever one is currently “empty” and waiting for data. Each packet is ACK’d so that the sender knows more data can be sent on that HTTP transaction. These are called byways.
- There are multiple byways each direction



Events

The events that flow both directions on a call are:

- **Ringing**: phone is ringing
- **Accepted**: user (human or bot) answered the call
- **Rejected**: the user (human or bot) declined the call
- **Failed**: something went wrong
- **Ended**: user (human or bot) ended the call

Moving Calls

- Servers can tell client to migrate media connection gracefully to new server
- Clients can do the same

Security

TLS

- Signaling encryption and integrity
- Media encryption and integrity
- Server authentication

OAuth

- Client authentication

End-to-End Security

- To the PSTN? Seriously? No.
- MLS if serious solution to identity/MITM ?

What’s the Scope

Help us get the input to write a charter

Problem Scope

- Just between data center servers or also between endpoints?
- Just audio or audio & video?

Transport Solution Scope

- Strong desire for some form of media over non-reliable QUIC
- Current draft stripes multiple media transaction over HTTP/3
- Looks like web-transports (BOF this week) would be an excellent match for this

Encoding Solution Scope

- gRPC is look very appealing or a bunch of this

Events Scope

- Simple JSON Events over push notification both directions
- DTMF is media - not an event

Capabilities & Negotiation Scope

- Could use SDP Offer / Answer
- Could use SDP O/A semantics with JSON like syntax (ORTC style)
- Could use Advertisement / Proposal
  - Endpoint says what is is capable of; Controller tells it what to do
  - Trivial for audio, but remains to be proven if easily scalable for video
- What we do: it is about supporting new systems, not old ones

Media Solutions Scope

1. Could put the media codec payload + timing/sequence info over new packets
2. Could just send RTP packets over new transport
   - Could do something in between these two options that removes redundant RTP info and leverages transport info, but semantically maps to and from RTP?

Security Scope

- Hop-by-Hop is effectively what we have today
  - For PSTN Trunking use cases, anything beyond this seems unlikely to be deployed
- Could expand to End-to-End, but need a feasible way to know you are encrypting to the right person / group

Next Steps

Delay, Delay, Delay

Preferred Approach

- ❑ Create a mailing list this week
- ❑ Get a Charter to IESG in the next 4 weeks
- ❑ Charter WG before next year
- ❑ Begin open source implementation before next year
- ❑ Update drafts and have a virtual interim in Feb
- ❑ Be at IETF 107 with good drafts & running code

Backup Slides

Inbound Calls

- Pretty much the opposite of the outbound case with an automated pre-configuration process
- Before any calls take place, the Webex side creates a HTTP/3 connection to Comcast and uses a specific API to pass Comcast an auth token and base URI for Webex endpoint
  - Later, when Comcast has an incoming call for Webex, it uses that URI and auth token to make the call happen

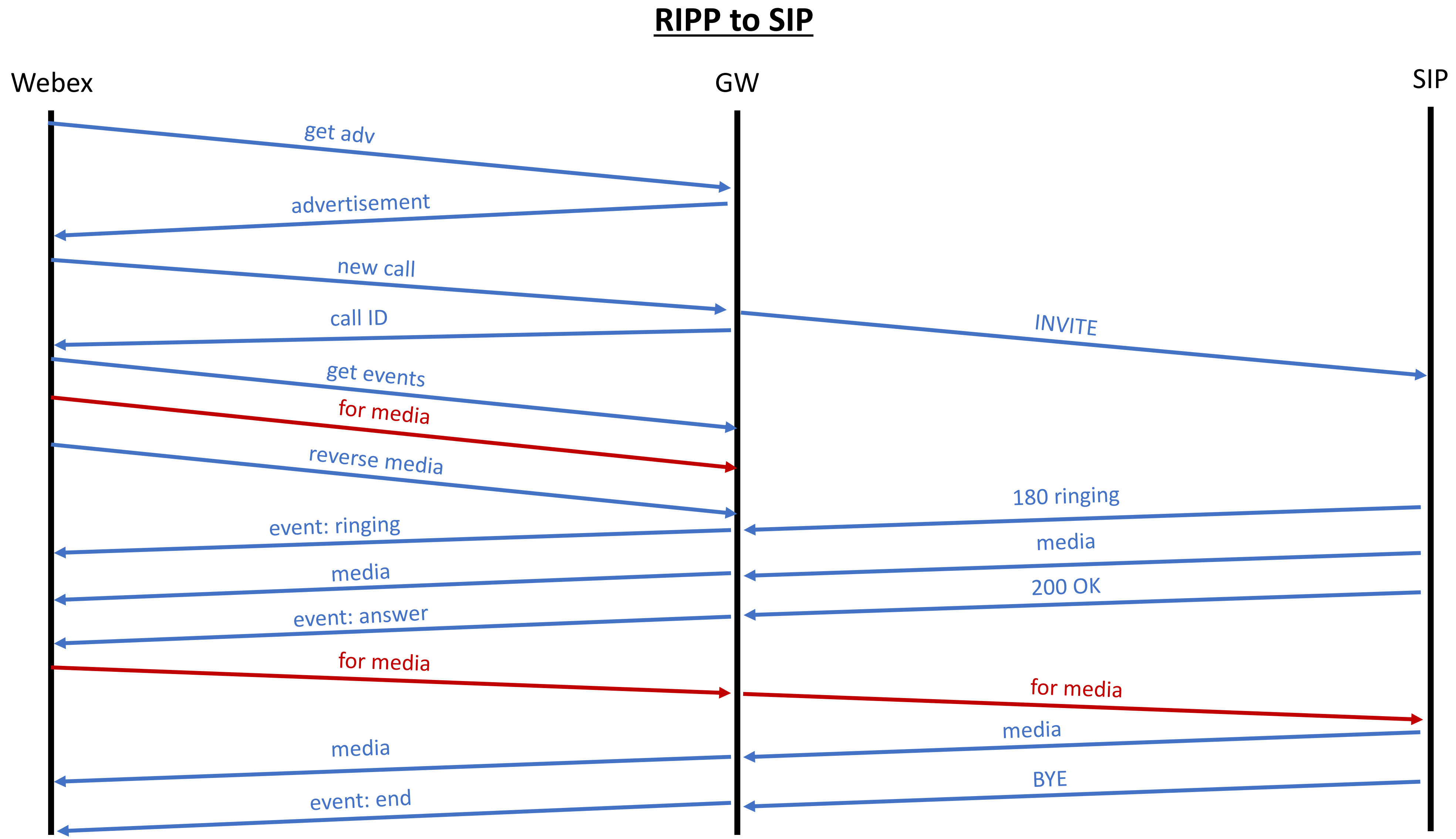
Recover from Failed Server

- If the client finds the connection failed, closed, or just dead (no ACKs), it simply forms a new HTTP connection using the URLs from before and picks up the old call
  - 5 second window to do this before the far side terminates the call

The API

- Get Auth token: https://ripp.example.com/trunks/{trunkID/consumerTrunk
- Get Advertisement of capabilities for this trunk: https://ripp.example.com/trunks/{trunkID/capAdv
- Get ongoing calls on this trunk: https://ripp.example.com/trunks/{trunkID/calls
- Get the previous event (i.e., ringing, answer ….): https://ripp.example.com/calls/{callID/prevEvent
- Wait for next event and return it: https://ripp.example.com/calls/{callID/event
- Flow of media chunks in forward direction: https://ripp.example.com/calls/{callID}/media-forward
- Flow of media chunks in reverse direction: https://ripp.example.com/calls/{callID/media-reverse

Gateway to SIP


RIPP to SIP
Webex
GW
SIP
get adv
advertisement
new call
call ID
INVITE
get events
for media
reverse media
180 ringing
event: ringing
media
media
200 OK
event: answer
for media
for media
media
media
BYE
event: end