WebRTC beyond one-to-one communication (Gustavo Garcia Bernardo) - webrtcHacks

WebRTC and its peer-to-peer capabilities are great for one-to-one communications. However, when I discuss with customers use cases and services that go beyond one-to-one, namely one-to-many or many-to-many, the question arises: “OK, but what architecture shall I use for this?”. Some service providers want to reuse the multicast support they have in their networks (we are having fun doing some experiments with this), some are exploring simulcast-based solutions, others are considering centralised solutions like MCUs/mixers, and a bunch of them are simply willing to place the burden on the endpoint by using some variation of a mesh-based topology.

The folks at TokBox (a Telefónica Digital company) have great experience with multiparty conferencing solutions. I thought it would be great to have my friend Gustavo Garcia Bernardo (Cloud Architect at TokBox) to share here his take on the topic.

At TokBox, Gustavo is responsible for architecture, design, and development of cloud components. This includes Mantis, the cloud-scaling infrastructure for the OpenTok, which uses the WebRTC platform. Before joining TokBox, Gustavo spent more than 10 years building VoIP products at Telefónica and driving early adoption of WebRTC in telco products. In fact, I’ve known Gustavo for 8 years now and the first time I met him it was preparing a proposal for a European Commission-funded research project on P2PSIP. Since then we’ve been collaborating in the IETF doing some work in the context of P2PSIP, ALTO and SIP related activities. A couple of years ago, while I was working with Acme Packet (now Oracle), we worked together designing and launching Telefonica’s Digital TuMe and TuGo. Lately we have both shifted our focus towards WebRTC.

{“intro-by” : “victor“}

WebRTC beyond one-to-one (by Gustavo Garcia Bernardo)

In spite of limited specification of anything beyond one-to-one audio and video calls in WebRTC, one of the most popular usages of this technology today is multiparty video conference scenarios. Don’t think just about traditional meeting rooms. There are different use cases beyond meeting rooms, including e-learning, customer support, or real time broadcasting. In each case, the core capability is being able to distribute the media streams from multiple sources to multiple destinations. So… if you are a service provider how can you implement a multi-party topology with WebRTC endpoints?

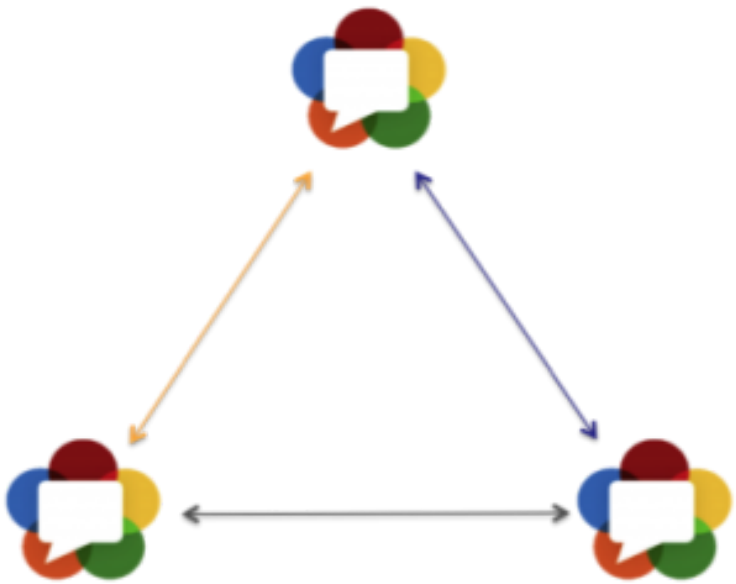
There are several different architectures that may be suitable depending on your requirements. These architectures basically they revolve around two axis:

- Centralized vs Peer-to-Peer (P2P) and
- Mixing vs Routing.

I will describe the most popular solutions here. If you need to go deeper into the protocol implications and implementation details the architectures, you can find all the relevant information in the RTP topologies IETF document.

Mesh solution

The Mesh approach is the simplest solution. It has been popular among new WebRTC servie providers because it requires no initial lack of infrastructure. The architecture is based on creating multiple one-to-one streams from every sender to every possible destination.



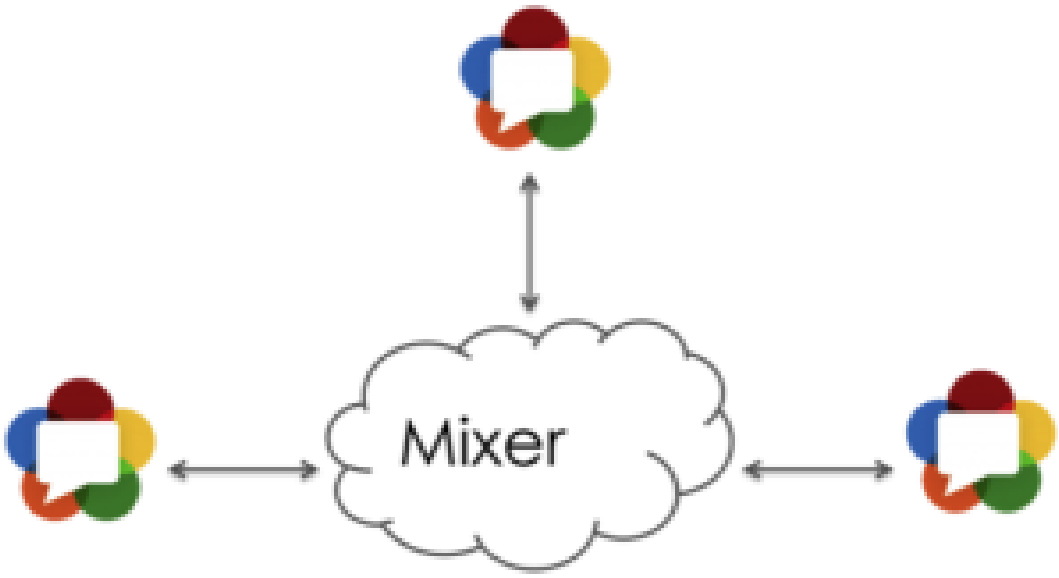
Mesh Solution

Even if it looks like an inefficient solution, in practice it is very effective and provides the lowest possible delay, with independent out-of-the-box bit rate adaptation for each receiver.

The “only” problem is that this solution requires lot of up-link bandwidth to send the media to all the destinations simultaneously, and existing browser implementations require a significant amount of CPU power to encode the video multiple times in parallel.

Mixer solution

This Mixer approach is the traditional solution for multi-conferencing, and has been used for years with great success. This success can be credited to the fact that it requires the least amount of intelligence in the endpoints. The architecture is based on having a central point maintain a single one-to-one stream with each participant. The central element then receives and mixes each incoming audio and video stream to generate a single stream out to every participant. One common term in the video conference industry for these centralized element is Multipoint Control Unit (MCU). In practice, use of an MCU usually refers to a mixer solution.



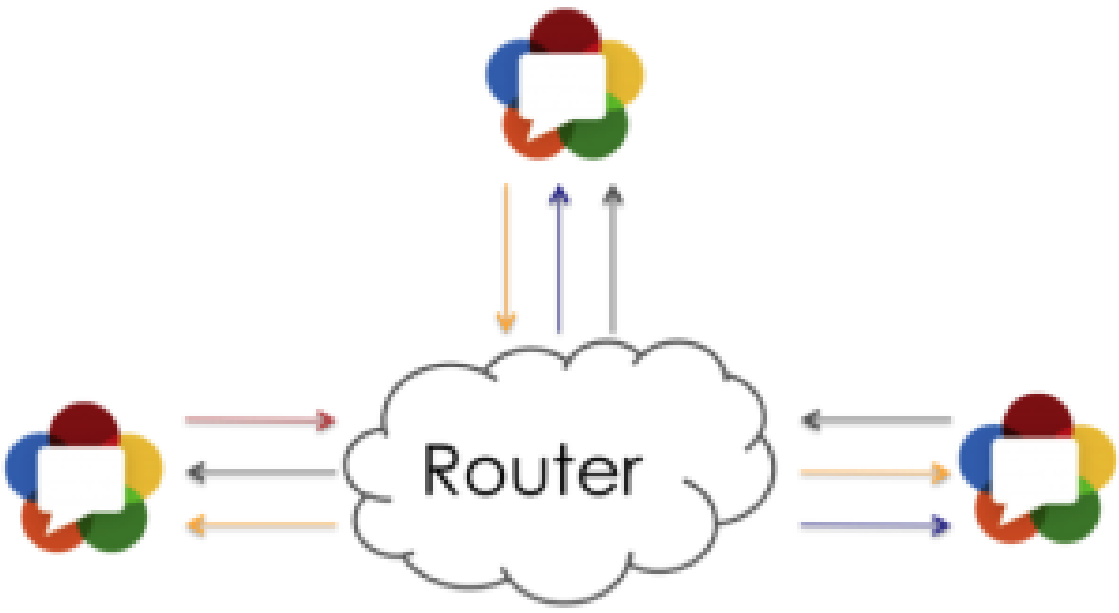
Mixer Solution

Mixers are very good solution for inter-operating with legacy devices. They also allows for full bit rate adaption because the mixer can generate different output streams, with different qualities for each receiver. Another advantage of a mixer solution is that it can utilize hardware decoding in the device, providing that many WebRTC devices would include the capability to decode a single video channel in the chipset.

The main drawbacks are the infrastructure costs in the MCU. Additionally, because mixing requires decoding and re-encoding, this introduces extra delay and loss of quality. Lastly, transcoding and composition may, in theory, result in less flexibility for the UI of the application (although there are workarounds for this issue).

Router solution

The Router (or relay) approach became popularized by H.264 SVC infrastructures, and it is the architecture being used by most of the new WebRTC platforms that have started without any legacy baggage. The architecture is based on having a central point receiving a stream from every sender and sending out a stream to every participant for each. This central point only does packet inspection and forwarding, but not expensive encoding and decoding of the actual media.



Router Solution

Routers provide a cheap and scalable multiparty solution, with better delays and no quality degradation compared with the traditional mixer solutions.

On the other hand, there is less experience in the industry building these infrastructures, and adaptation of streams to different receivers becomes tricky. It requires support in the endpoints to generate multiple versions of the stream (i.e. with simulcast or VP8 temporal scalability) that can later be selectively forwarded in the router, depending on the capabilities of each receiver.

Which architecture should I use?

Unfortunately, there is no simple answer. In fact some commercial solutions include support for all of them, in order to optimize different customers' use cases. However, there are some general rules of thumb that you can use.

If you are providing an audio only service, or need interoperability with legacy devices, then the Mixer architecture is likely the most appropriate for you. Also, in some cases where the cost of the infrastructure is not an issue, and the participants have very heterogeneous connectivity, this can be a good solution.

If you are building a service to be used by users with really good connections and powerful devices (i.e. an internal corporate service), and the number of participants is limited, then you may get good results with a Mesh architecture.

In general if you are providing a large scale service, preference should be given to the Router approach. At the end of the day, the router solution is closest to the Internet paradigm of putting the intelligence in the border of the network, to achieve better scalability and flexibility when building the end user applications.

What is missing in WebRTC?

Even if there are both commercial and free solutions that provide multiparty services on WebRTC, there are still issues that need to be addressed in the base technology to enable an even better user experience. These include:

1. Improved audio processing and coding, specially in the acoustic echo cancellation and noise suppression algorithms.
2. More advanced and flexible congestion control, allowing developers to modify on-the-fly, the parameters of streams, such as bitrate, quality, or resolution of video.
3. Simulcast and layered video coding support to adapt the original video stream to the capabilities of each receiver independently, without expensive transcoding. In case of simulcast, it can be done today with some workarounds. However, in the case of VP8 layered video coding, it is only possible today by hacking WebRTC codebase.

All in all, we are in a good position to start providing multiparty services to our customers based on WebRTC technology. As the standards evolve, as more APIs are provided, and as better implementations in more browsers are shipped, the future of web based video conferencing becomes even more promising.

Want to keep up on our latest posts? Please click here to subscribe to our mailing list if you have not already. We only email post updates. You can also follow us on twitter at @webrtcHacks for blog updates.