Advanced Security and Large-scale Optimization

Advanced Security and Large-scale Optimization

- Up to this point, we have covered a shallow portion of the entirety of WebRTC.

- This has been limited to what we could build in a session inside our local computer without hooking up to any real services.

- This is great if it's only limited to you or a few of your friends but, unfortunately, this will not lead you to being able to connect thousands of people across the world.

Advanced Security and Large-scale Optimization

- The aim of this session is to provide a small amount of information on these topics so that you can research them in more depth at your leisure.
- Most of the information will be conceptual and we will not build a working demo in this session.
- By the end of the session, you should have a good idea of what concerns you will have when building a large-scale WebRTC-based service and where to go for more information.

Securing the signaling server

Using encryption

- The largest and most obvious upgrade is mandatory encryption of the signaling server.
- Encrypting the messages of the signaling server will ensure that no one can intercept a message to the server, thus figuring out which clients are talking to whom.
- This is easily the largest gap in security that the signaling server we built has right now.
- It is also the easiest to patch up since encryption is a highly standardized and widely used technology on the Web today.

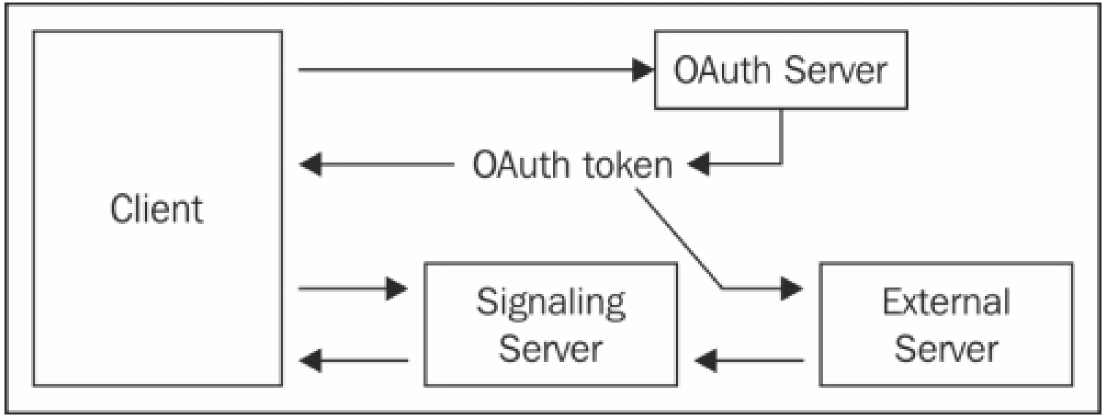
Securing the signaling server

Using an OAuth provider

- One of the other major upgrades to our example would be the integration of an identity provider.
- An identity provider is software that allows your application to keep track of users using another application's identity data.
- One good example of this is the Google login.
- This would provide each user with a unique user ID as well as their list of contacts to call.

Securing the signaling server

- This will also add to the security of our server by making sure only authenticated users are allowed to access our application:



Securing the signaling server

- To ensure that both users authenticate themselves before they make any calls, the authentication mechanism should be in place before the user can interact with the signaling server.
- If the user has to log in in order to do anything, this will reduce the exposed servers that a user can maliciously try to hack into.
- Once you choose an OAuth service, you can begin integrating it with your signaling server.

Supporting mobile devices

- With the meteoric rise of the Web on mobile devices, most developers will have to support WebRTC calling on phones at some point.
- It is the medium that is most often paired with the idea of web-based calling.
- The idea of ditching expensive minute plans and sticking with a data service is an alluring concept.
- Also, a web page can make use of the microphone and camera already attached to the device instead of needing to purchase these items for a desktop computer.

Supporting mobile devices

- To solve the data connection issue we need to simply reduce the amount of data that we send over the network.
- The data that we send is measured by the number and size of each frame of video that is captured.
- These are packaged and encrypted, then sent across the user's mobile network frame-by-frame to the other side.

Supporting mobile devices

- Here is the code required to do this in our application:

Refer to the file 8_1.txt

- We need to define a set of constraints for each type of device.

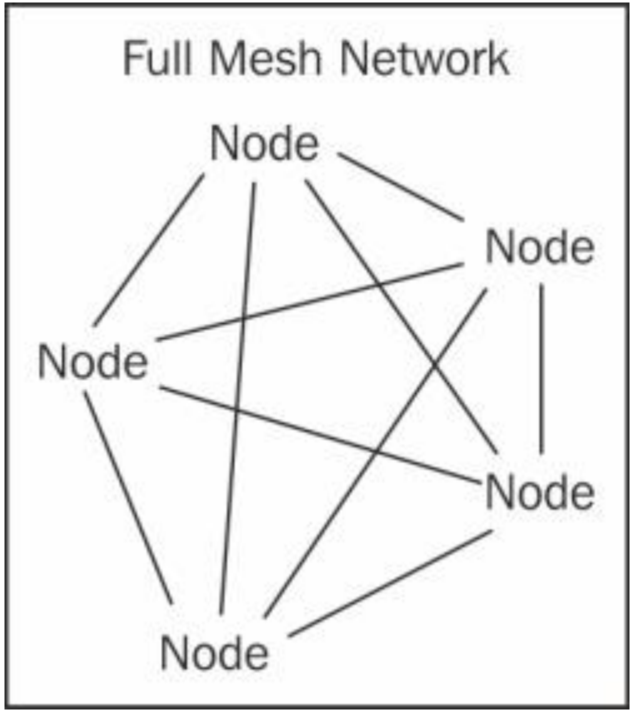
- In this case, we only define one for mobiles and the other for desktops.

Introduction to mesh networking

- Once your WebRTC application is working on multiple devices with a reasonable amount of security, the next question is: how do I scale this up to multiple users? One-to-one calls are great, but what if we can connect several users together inside one call? This is where we begin to dive into the world of mesh networking.

Introduction to mesh networking

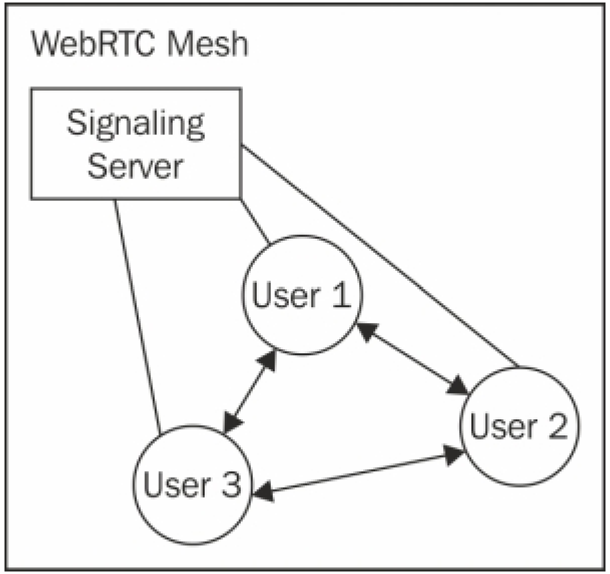
- Each node is connected with every other node in the network and no one node is responsible for the entire network:



Introduction to mesh networking

- Now we can expand this idea and add more nodes to our network. If two users can connect together then why not a third? There is no limit on the number of WebRTC connections you can have open in one browser at a time.

- Each user in our network can connect to multiple people at the same time, thus creating a mesh of users all in one call:



Introduction to mesh networking

Types of network meshes

Here is a set of simple criteria to judge each mesh type by:

- What is the minimum or maximum bandwidth a specific user might have?

- What is the maximum number of users I want to connect?

- How much data loss is acceptable for any given user?

Introduction to mesh networking

Everyone to everyone

- The first type of network we will cover is one where every node connects to every other node.
- This network, by far, is the easiest to implement since the logic is simple.
- The server just has to keep track of everyone that joins the call and each person should broker a WebRTC connection with every other user in the list.

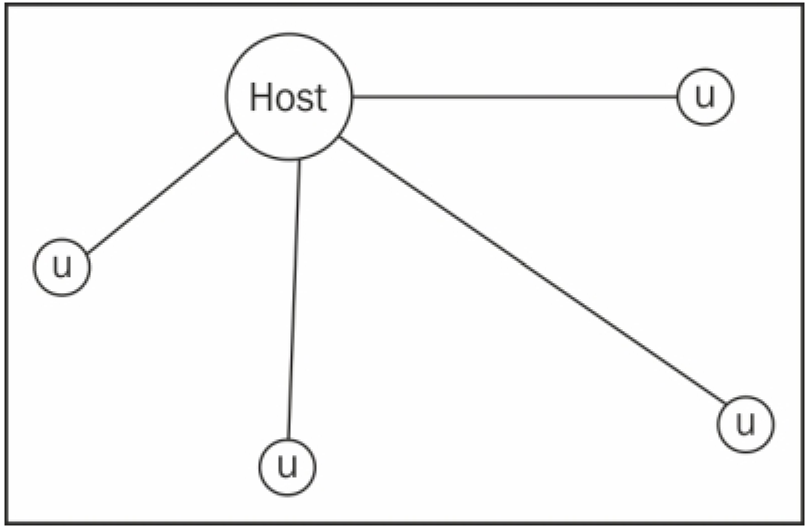
Introduction to mesh networking

Star network

- To support more users, we have to be smarter about the way we connect them.
- One simple way we can do this is to reduce the number of connections we have within the call.
- We can use a star topology to alleviate the issues we have with connecting everyone on the call.

Introduction to mesh networking

- The star network works by having everyone connect to one host node, and then this node is responsible for sending the correct video streams to all the other nodes on the network:



Introduction to mesh networking

- One issue that we will have to solve is the ability to actually select a user to be the host on the call.
- This is a feature that could be added to the signaling server.
- When the user logs in, they should collect a certain amount of information about the user and send it to the signaling server.
- We can test the user's bandwidth using a small file download to see what type of network the user is on.

Introduction to mesh networking

- The strategy to do this is to make an image download of a known size, and time how long it takes to download this file:

Refer to the file 8_2.txt

Introduction to mesh networking

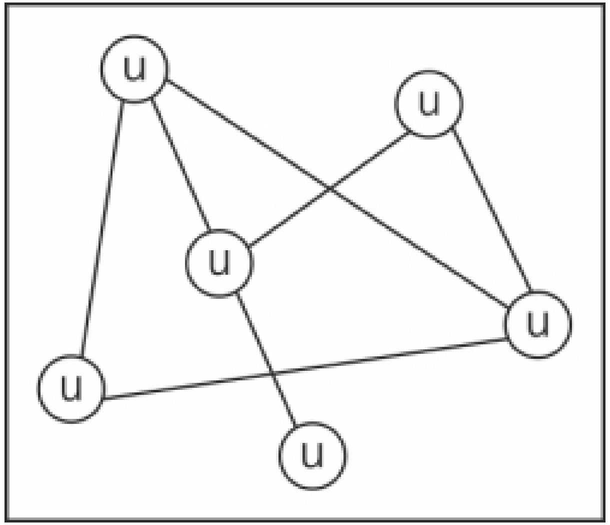
The logic here is fairly straightforward:

- We set up a new image along with defining the known size of the image beforehand
- Set up a load handler to determine the speed of the download once the image is loaded
- Set startTime of the download as the current date/time in JavaScript
- Set the source of the image that begins the download

Introduction to mesh networking

Partial mesh

- The last potential mesh we will cover is a partially-meshed network.
- A partial mesh is great in applications outside the realm of multiuser communication.
- This is especially helpful when using the data channel, as every node in the mesh network does not always have to be connected to every other node:



Introduction to mesh networking

Limits of mesh networking

- Although mesh networking is an easy path to enabling multiuser communication, it does have its drawbacks.
- In practice, mesh networking is a straightforward technique that is simple enough to understand.
- Implementing this technique in a production-ready environment is also fairly simple, since much of the logic can be implemented on the clients themselves.
- There is no need to add additional high-capacity servers other than the ones that select which nodes connect to each other.

Introduction to mesh networking

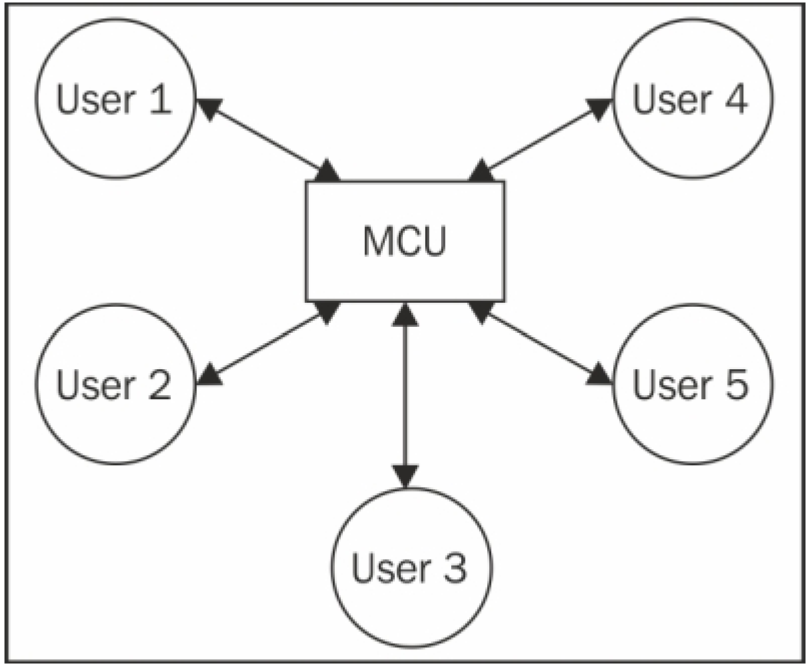
- The issue with mesh networking is that we are trading the ease of development and low cost for the user experience.
- A great example of this transition has been in video games over the years.
- Initially, many games used the star network model for the purpose of networking their users.
- One user would create the game and others would join the host's game, essentially creating the same pattern we reviewed earlier.

Video conferencing with more users

- Many large-scale communication companies have made the transition to server-client methods of networking vast number of users.
- There have been a wide number of different solutions that work in many ways, but they are all built on the principal of using servers as nodes in the network instead of clients.
- These MCUs give networks better stability, performance, and a better overall user experience at the cost of being expensive.

Video conferencing with more users

- It also allows the developers of the application to scale the bandwidth of the network, keeping in mind the needs of the user base:



Video conferencing with more users

- You may notice the major difference here is that the server can stitch together the other streams and only provide one stream back to each user.
- This is an extremely powerful feature since it cuts down the bandwidth needed to support many users in one call.
- Instead of needing to send multiple streams to each user, they only have to support the bandwidth for one video stream to the server and one back.

Video conferencing with more users

- For all the benefits mentioned here, you might wonder how you can actually start implementing an MCU infrastructure in production.
- This is the major trade-off when using this technology—it is expensive and hard to set up.
- As of writing this, there are a few open source MCU servers dedicated to WebRTC development, but the installation and compilation of these services are only for the most advanced users.

The future of conference calls

- This is just the tip of the iceberg when it comes to large-scale WebRTC applications.
- With the globalization of technology communication, platforms are not going to go out of style anytime soon.
- It is an industry that will continue to be invested in by many large corporations.
- This means that the technology presented here is going to become more accessible and powerful over time.

Summary

- By now, your head should be filled with dreams of the amazing things that WebRTC can do.
- We covered a large list of advanced techniques that can be used to greatly enhance the usability and performance of your application.
- Each of these topics has a wealth of information to read about and also discover on your own.
- It would be a great idea to spend time researching each one at your leisure if you plan on continuing your WebRTC education!

Summary

- All these topics are aimed at learning how to release a large-scale WebRTC application.
- When learning how to use WebRTC, developers may not think about what happens when you go from two users to two hundred or two thousand users at a time.
- There can be a lot of growing pains if you are not prepared for what may happen when adding more users to your application.