Tasks

- 1. Client
  - Modify client to have one end station
  - Change listener to exchange messages with server
- 2. Server
  - Very simple node server to implement signaling API
- API Spec – as follows…

webRTC Signaling Spec

- Two REST methods:
  - send
  - Poll
- Maintains a ‘directory’ of end points (similar to the client model)

```
var directory = {
  endpointName: {
    messages: [ {from:’endpointName’, message:’String’} ]
  }
}
```

- End point added to directory on poll…

Poll: https://server/poll/*myname*

- *myname* – only parameter and is the name of the sending endpoint
- Server Behaviour
  - If ‘myname’ is *not* in the directory then create it

```
var directoryEntry = getDirectoryEntry(myname);
```

  - Reply with JSON formatted structure:

```
{ directory: Object.keys(directory),
  messages: directory[endpointName].messages
}
```

  - Once this message has been sent:

```
directory[endpointName].messages.length = 0;
```

    (Delete messages)

Send: https://server/send/*myname*/*toname*

- HTTP POST request
- Send a message from *fromname* to *toname*
- The message is in the post data body and is JSON formatted text
- Server does *not* need to JSON.parse this text
- Behaviour:

```
 if (directory[toname]!=null) {
    var directoryEntry = directory[toname];
    directoryEntry.messages.push({from: myname, data:payload.data});
    send reply – 'success'
}
else sendReply - 'unknown destination'
```