Iñaki Baz
@ibc_tw

José Luis Millán
@jomivi

BUILDING MULTI-PARTY VIDEO APPS WITH

MEDIASOUP

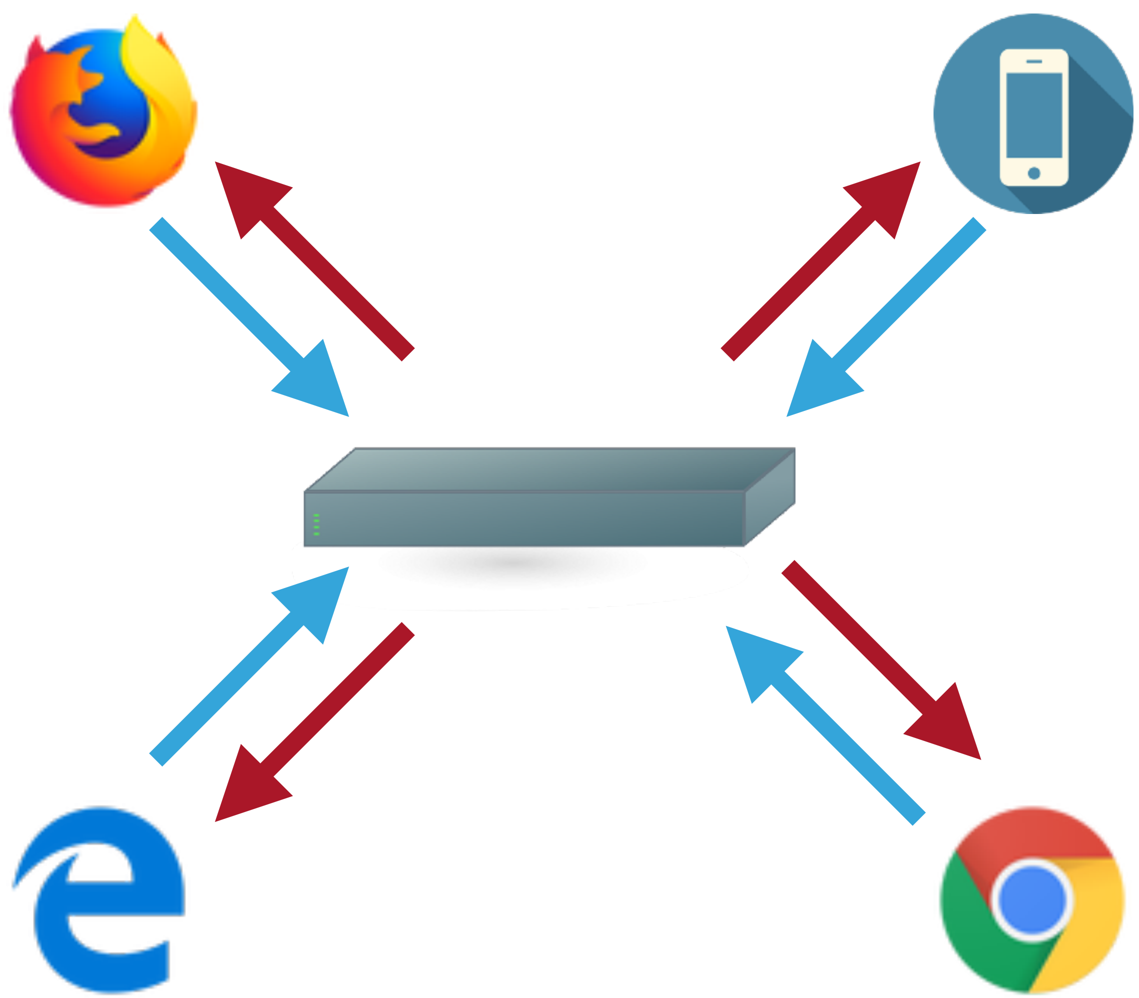
WHAT WE DO

- RFC 7118 “The WebSocket protocol as a Transport for SIP”
- JsSIP “The JavaScript SIP library”
- OverSIP (first SIP proxy with WebSocket support)
- mediasoup “Cutting Edge WebRTC Video Conferencing”

MEDIASOUP…

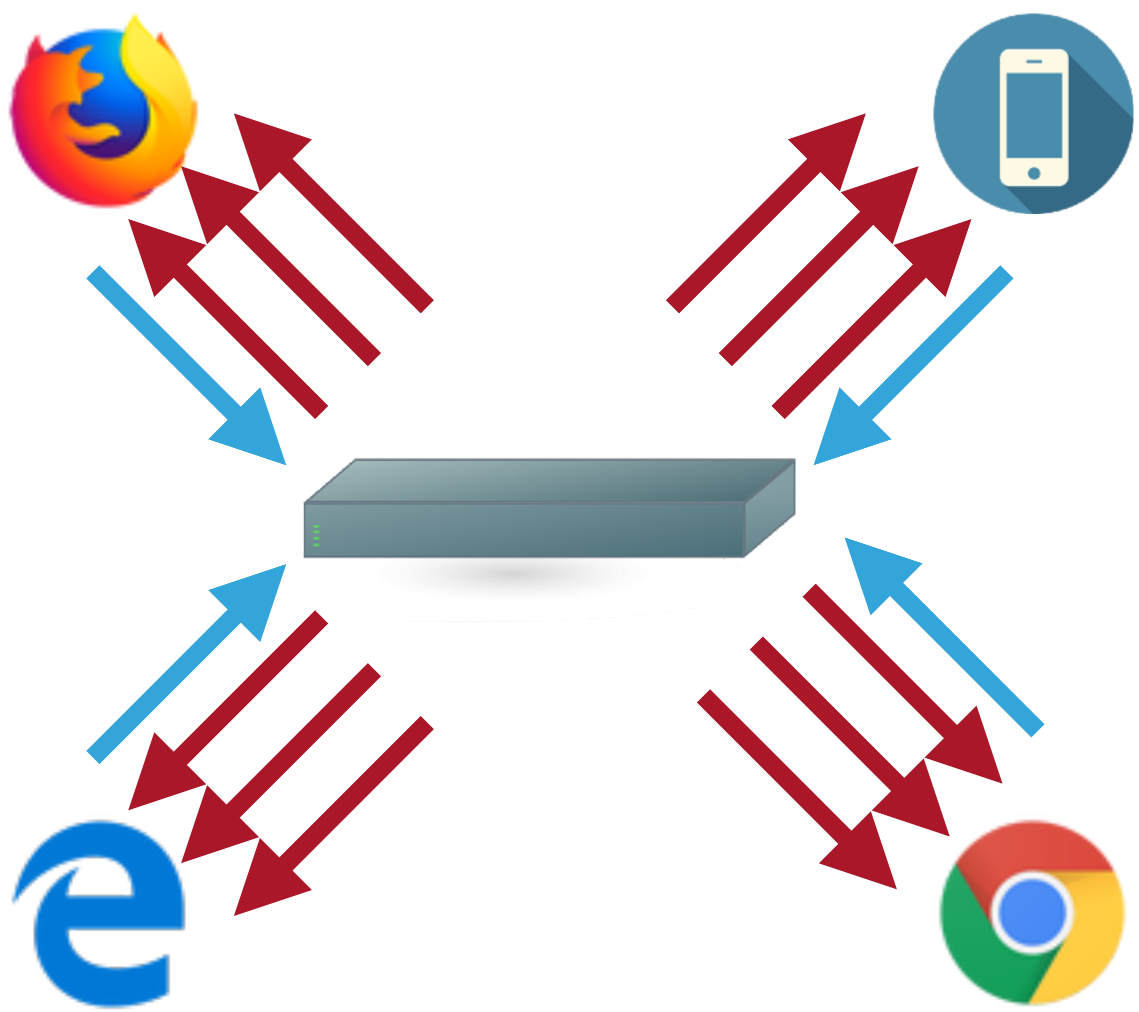
- is:
  - a client and server side library
  - a muti-party video component
- is not:
  - an application itself
  - an end product

MULTIPOINT CONTROL UNIT (MCU)



- Participants send their media to the server
- Participants receive others media in a single stream, mixed by the server
- ✓ Clients need to handle a single remote stream
- ✓ Server performs transcoding
- ✓ Low download link required
- ⦿ CPU intensive in server side
- ⦿ High latency
- ⦿ Fixed remote participants representation
- ⦿ Low flexibility in client side

SELECTIVE FORWARDING UNIT (SFU)



- Participants send their media to the server
- Participants receive others media in separate streams, one each
- ✓ Server simply routes. High throughput, low latency
- ✓ Low CPU usage in server side
- ✓ Client can decide what streams to receive
- ✓ Client/Server can choose quality for each stream
- ◉ Higher download link required
- ◉ No transcoding

```
protoo-server protoo-server version 3.0.0 +0ms
mediasoup mediasoup version 2.1.0 +0ms
mediasoup Server() +48ms
mediasoup:Server constructor() [options:{ numWorkers: 1, logLe
, rtcMaxPort: 39999 }] +0ms
mediasoup:Worker constructor() [id:ugouolln#1, parameters:"--l
mediasoup:Channel constructor() +0ms
opensips-summit-2018:INFO mediasoup Server created +0ms
opensips-summit-2018:Room constructor() +1ms
protoo-server:Room constructor() +0ms
mediasoup:Server Room() +10ms
mediasoup:Worker Room() +9ms
mediasoup:Room constructor() +0ms
mediasoup:Channel request() [method:worker.createRouter, id:80
protoo-server:WebSocketServer constructor() [option:{ maxRecei
opensips-summit-2018:INFO protoo WebSocketServer created +5ms
opensips-summit-2018:INFO protoo plain WebSocket listening [ip
mediasoup:Channel request succeeded [id:80075850] +28ms
mediasoup:Worker "worker.createRouter" request succeeded +29ms
protoo-server:WebSocketServer onRequest() [origin:https://firs
opensips-summit-2018:INFO connection request [playerId:iñaki_Q
protoo-server:WebSocketTransport constructor() +0ms
protoo-server:WebSocketServer _onRequest() | accept() called +
opensips-summit-2018:INFO:Room handleNewPlayerConnection() [pl
protoo-server:Room createPeer() [peerId:"iñaki_Q2KFwW", transp
protoo-server:Peer constructor() +0ms
opensips-summit-2018:INFO:Player#iñaki_Q2KFwW constructor() +2
opensips-summit-2018:Player#iñaki_Q2KFwW protoo "request" even
opensips-summit-2018:Player#iñaki_Q2KFwW mediasoup-client requ
mediasoup:Room receiveRequest() [method:queryRoom] +5s
opensips-summit-2018:Player#iñaki_Q2KFwW protoo "request" even
opensips-summit-2018:Player#iñaki_Q2KFwW mediasoup-client requ
mediasoup:Room receiveRequest() [method:join] +136ms
mediasoup:Room _createPeer() [peerName:"iñaki_Q2KFwW] +3ms
mediasoup:Peer constructor() [internal:{ routerId: 52943765, p
opensips-summit-2018:INFO:Room player joined [player:iñaki_Q2K
opensips-summit-2018:Player#iñaki_Q2KFwW protoo "request" even
opensips-summit-2018:Player#iñaki_Q2KFwW mediasoup-client requ
mediasoup:Peer receiveRequest() [method:createTransport] +192m
mediasoup:Peer _createWebRtcTransport() [id:18675929, directio
mediasoup:Channel request() [method:router.createWebRtcTranspo
mediasoup:Channel request succeeded [id:36164864] +2ms
mediasoup:Peer "router.createWebRtcTransport" request succeede
mediasoup:WebRtcTransport constructor() +0ms
mediasoup:WebRtcTransport setMaxBitrate() [bitrate:1000000] +4
```

MEDIASOUP SERVER

- Programmable WebRTC Selective Forwarding Unit (SFU)
- Written in C++ in its core, using libuv for asynchronous IO
- Written in JavaScript ES6 in the surface
- Offers a ORTC like API (no SDP but RTC Objects)
- Presented as a Node.js module

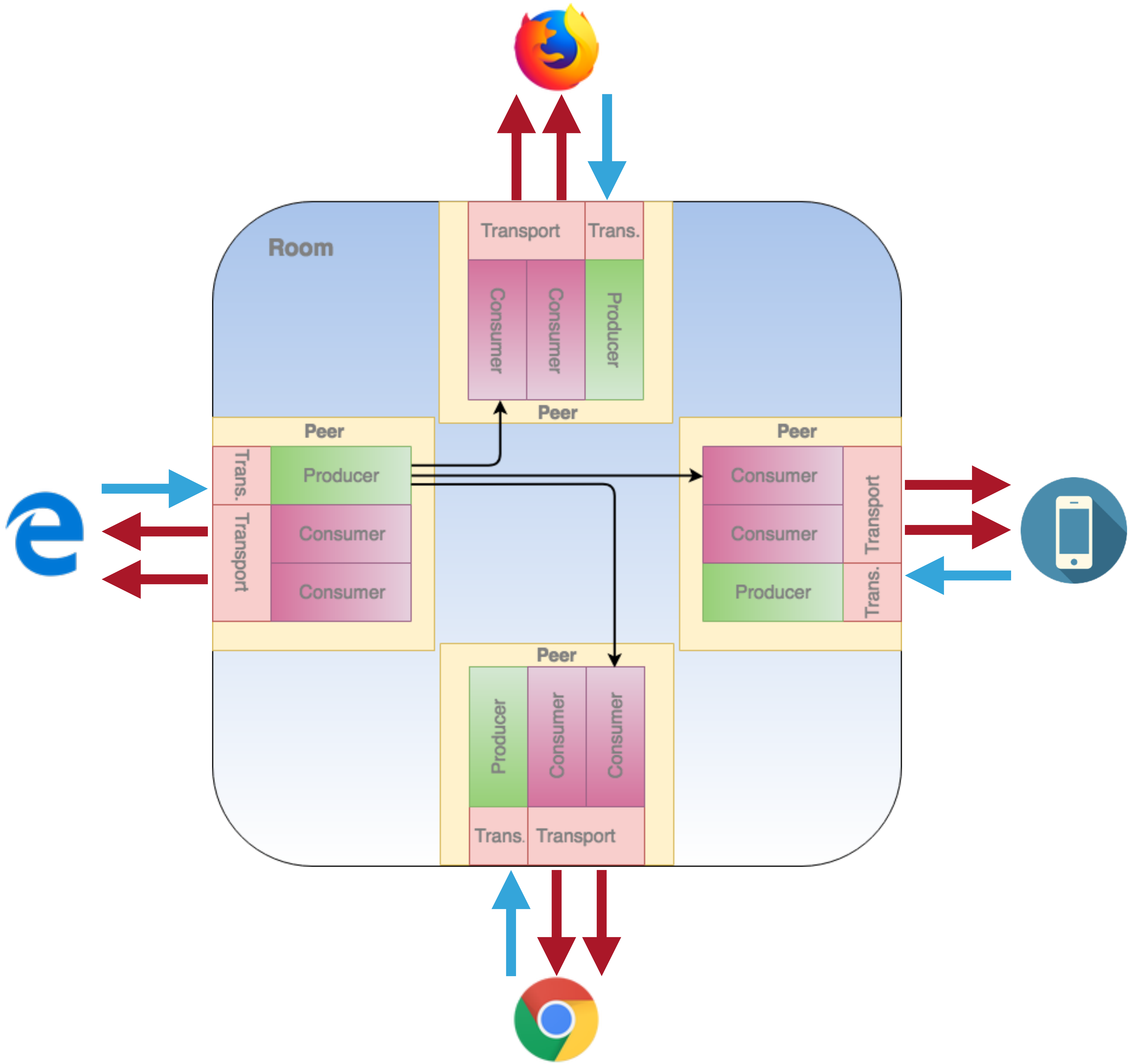
```
$ npm install mediasoup
```

NODE.JS MODULE ARCHITECTURE

- **Server** instance launches the C++ workers
- **Rooms** are created within a server
- **Peers** are created within a room

MEDIASOUP PEER

- WebRTC endpoint in the server side
- Interacts with a remote endpoint (browser, native client)
- Handles transports, producers and consumers
- A **Transport** represents the channel for ICE, DTLS, SRTP
- A **Producer** represents a media track produced by the remote peer
- A **Consumer** represents a media track produced by other peer and consumed by this one



Room
Transport
Trans.
Consumer
Consumer
Producer
Peer
Peer
Trans.
Producer
Transport
Consumer
Consumer
Peer
Consumer
Consumer
Producer
Transport
Trans.
Peer
Producer
Consumer
Consumer
Trans.
Transport

MEDIASOUP CLIENT

```
opensips-summit-2018:NetClient constructor() [player:▶Player {_
protoo-client:WebSocketTransport constructor() [url:"
wss://firstsight.local.mediasoup.org:3000/p/iñaki_KgITbK", optic
protoo-client:WebSocketTransport _setWebSocket() [currentAttempt
protoo-client:Peer constructor() +0ms
mediasoup-client:Room constructor() [options:
▶ {requestTimeout: 10000, transportOptions: {…}}] +0ms
opensips-summit-2018:NetClient protoo Peer "open" event +87ms
mediasoup-client:Room join() [peerName:"iñaki_KgITbK"] +76ms
mediasoup-client:Room _sendRequest() [method:queryRoom, request:
▶ {method: "queryRoom", target: "room"}] +2ms
opensips-summit-2018:NetClient sending mediasoup request [methoc
▶ {method: "queryRoom", target: "room"} +3ms
mediasoup-client:Room request succeeded [method:queryRoom, respo
▶ {rtpCapabilities: {…}, mandatoryCodecPayloadTypes: Array(0)}]
mediasoup-client:Room join() | got Room settings:
▶ {rtpCapabilities: {…}, mandatoryCodecPayloadTypes: Array(0)} +
mediasoup-client:Chrome55 getNativeRtpCapabilities() +0ms
mediasoup-client:Room join() | native RTP capabilities:
▶ {codecs: Array(28), headerExtensions: Array(14), fecMechanisms
mediasoup-client:Room join() | extended RTP capabilities:
▶ {codecs: Array(2), headerExtensions: Array(4), fecMechanisms:
mediasoup-client:Room join() | effective local RTP capabilities
▶ {codecs: Array(3), headerExtensions: Array(4), fecMechanisms:
mediasoup-client:Room _sendRequest() [method:join, request:
▶ {method: "join", target: "room", peerName: "iñaki_KgITbK", rtp
  }
] +1ms
opensips-summit-2018:NetClient sending mediasoup request [methoc
▶ {method: "join", target: "room", peerName: "iñaki_KgITbK", rtp
  }
 +58ms
mediasoup-client:Room request succeeded [method:join, response:
▶ {peers: Array(0)}] +6ms
mediasoup-client:Room join() | joined the Room +2ms
mediasoup-client:Room createTransport() [direction:send] +1ms
mediasoup-client:Transport constructor() [direction:send
, extendedRtpCapabilities:
▶ {codecs: Array(2), headerExtensions: Array(4), fecMechanisms:
mediasoup-client:Chrome55 constructor() [direction:send, extende
▶ {codecs: Array(2), headerExtensions: Array(4), fecMechanisms:
mediasoup-client:RemotePlanBSdp constructor() [direction:send
, rtpParametersByKind:▶ {audio: {…}, video: {…}}] +0ms
mediasoup-client:Room createTransport() [direction:recv] +22ms
mediasoup-client:Transport constructor() [direction:recv
```

- client-side javascript SDK

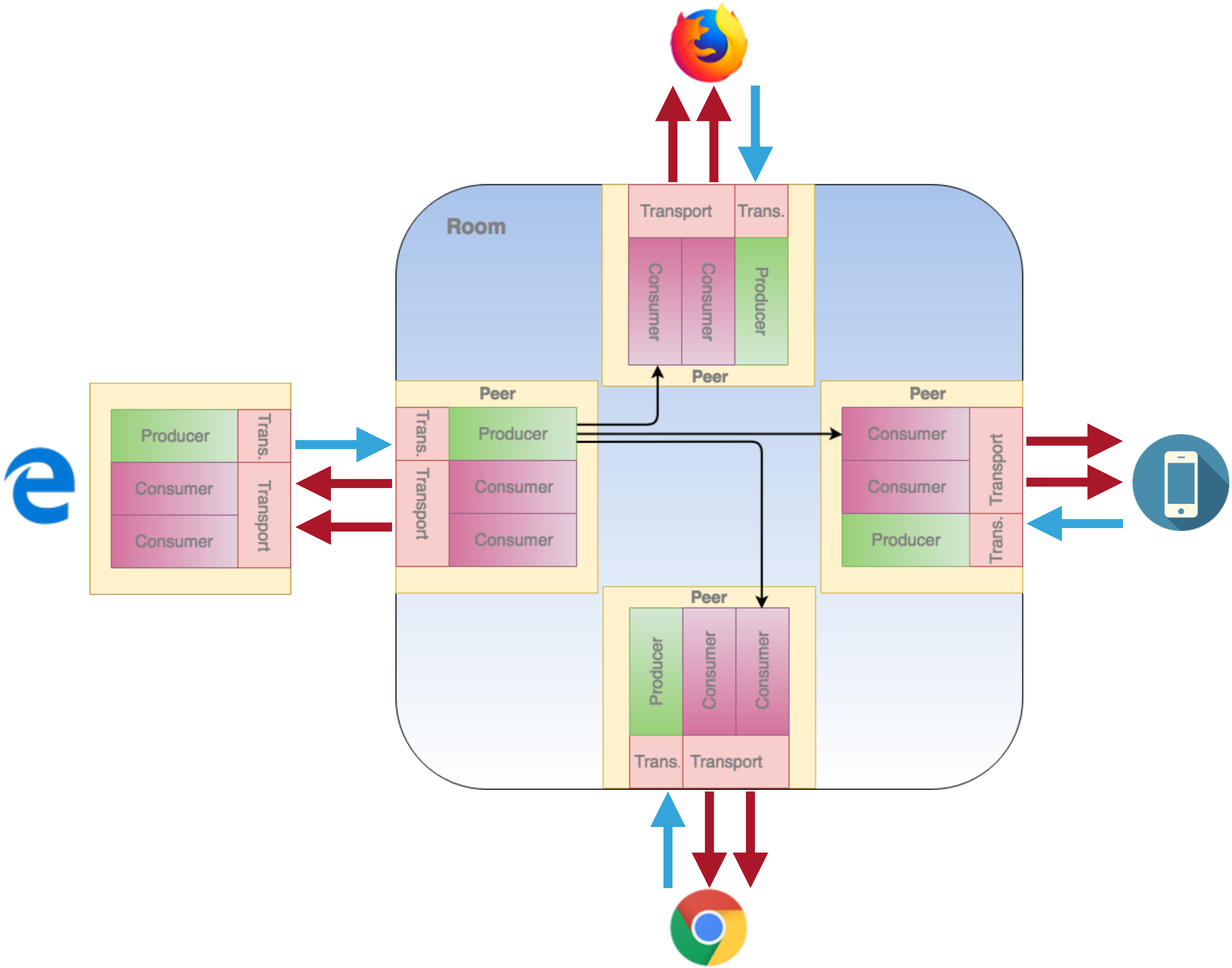
```
$ npm install mediasoup-client

$ bower install mediasoup-client
```

- Abstracts the app from the underlaying WebRTC device
  - SDP specifics, WebRTC API, ORTC API
- Handles message exchange with mediasoup server

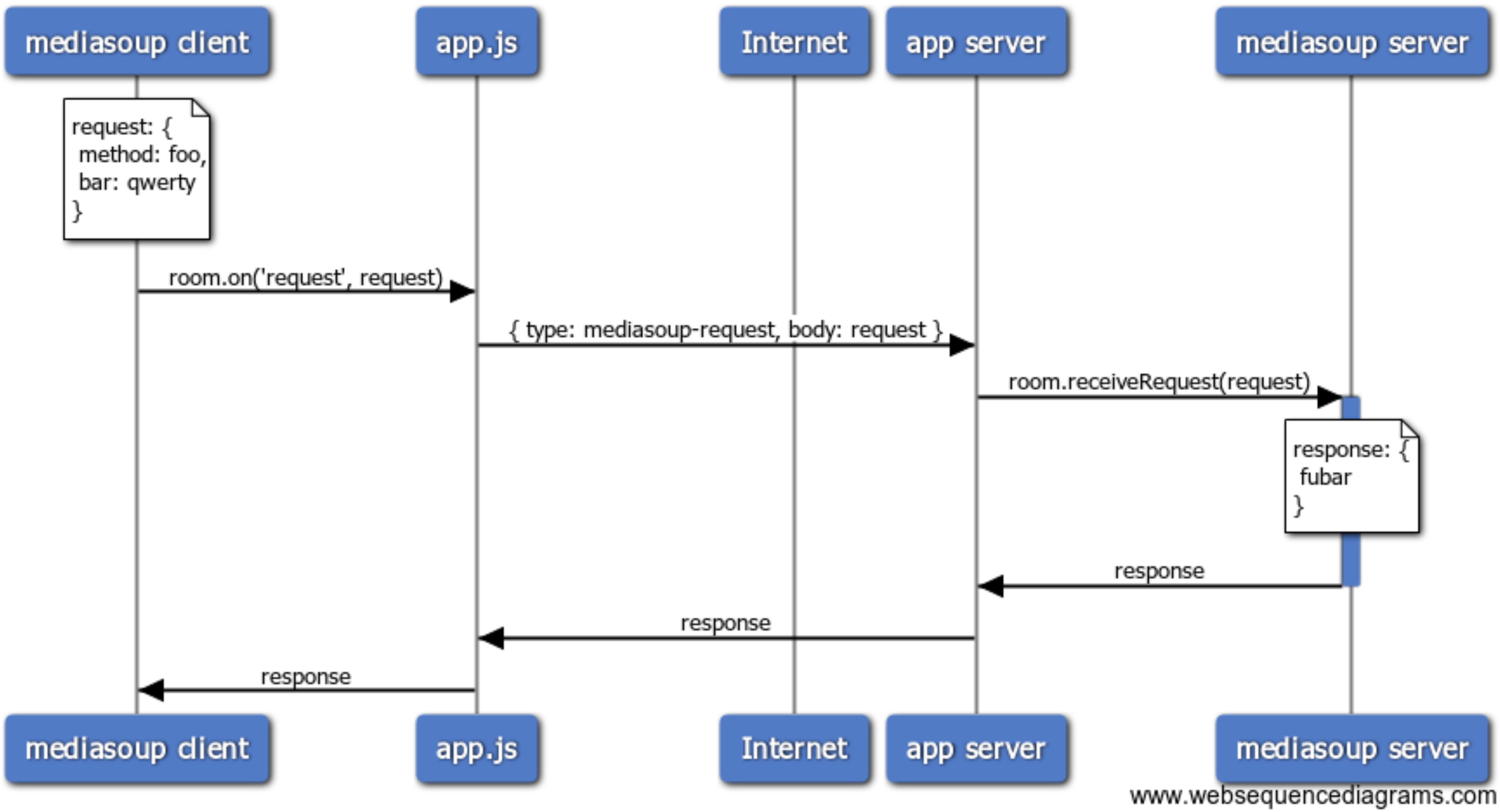
MEDIASOUP CLIENT SDK ARCHITECTURE

- **Room** representing the room in mediasoup server
- Local peer representing the local WebRTC endpoint
  - It consists of **Transports** and **Producers**
- Remote **Peers** are added to the room as they join
  - They consist of **Consumers**

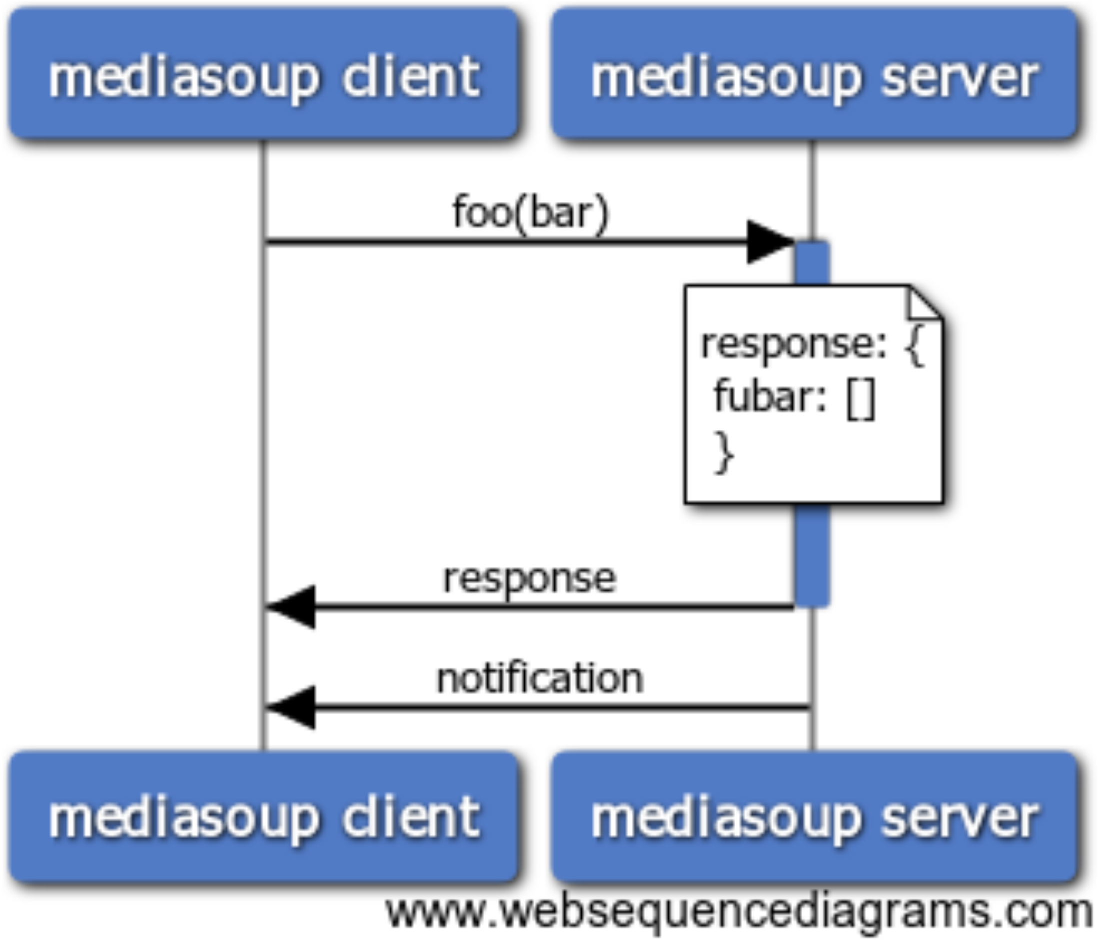

Room
Transport
Trans.
Consumer
Consumer
Producer
Peer
Peer
Trans.
Transport
Producer
Consumer
Consumer
Producer
Consumer
Consumer
Trans.
Transport
Peer
Consumer
Consumer
Producer
Transport
Trans.
Peer
Producer
Consumer
Consumer
Trans.
Transport

MEDIASOUP CLIENT AND SERVER INTERACTION

MEDIASOUP CLIENT AND SERVER MESSAGE EXCHANGE



MEDIASOUP CLIENT AND SERVER MESSAGE EXCHANGE (SIMPLIFIED)

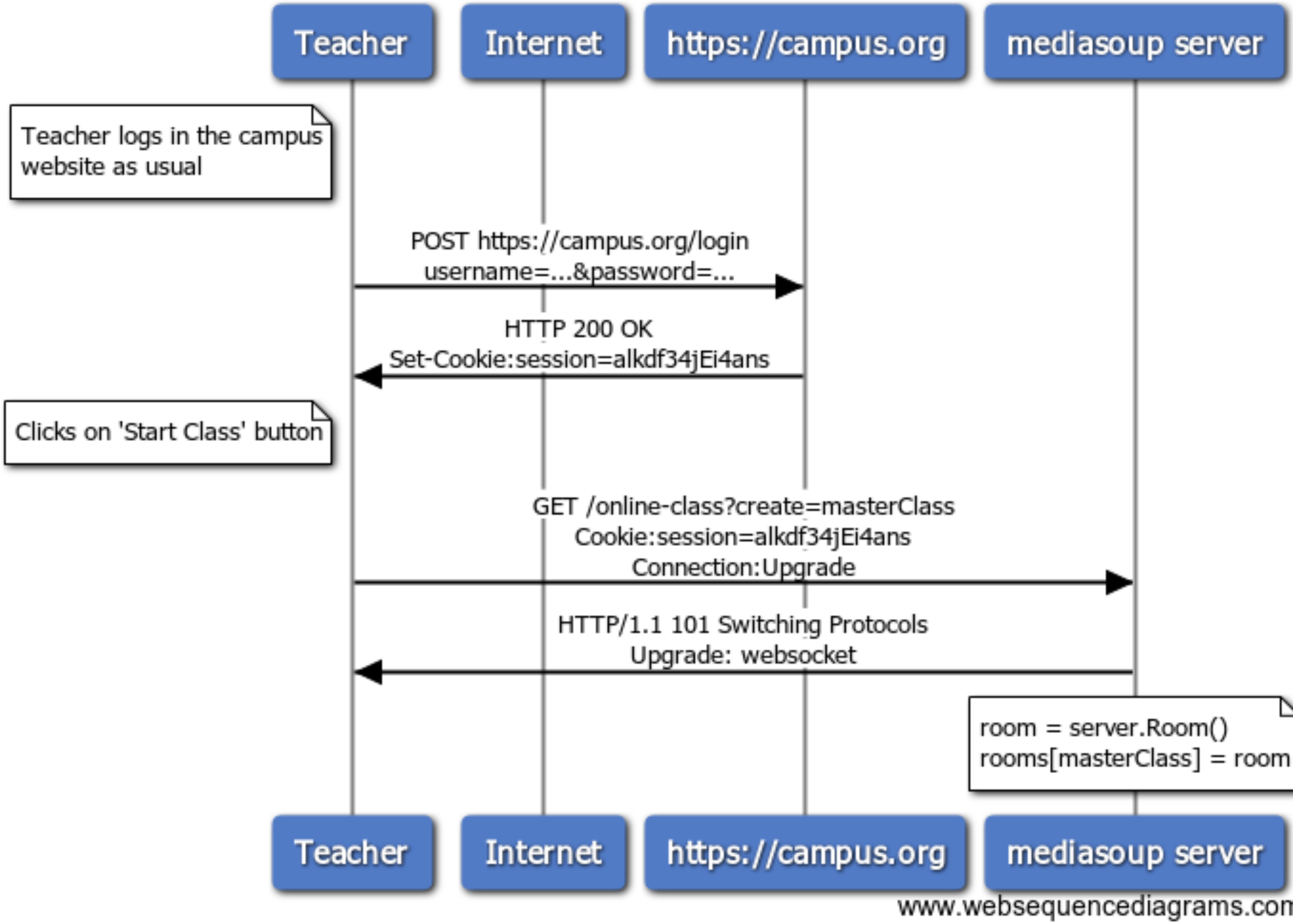


BUILDING THE APPLICATION

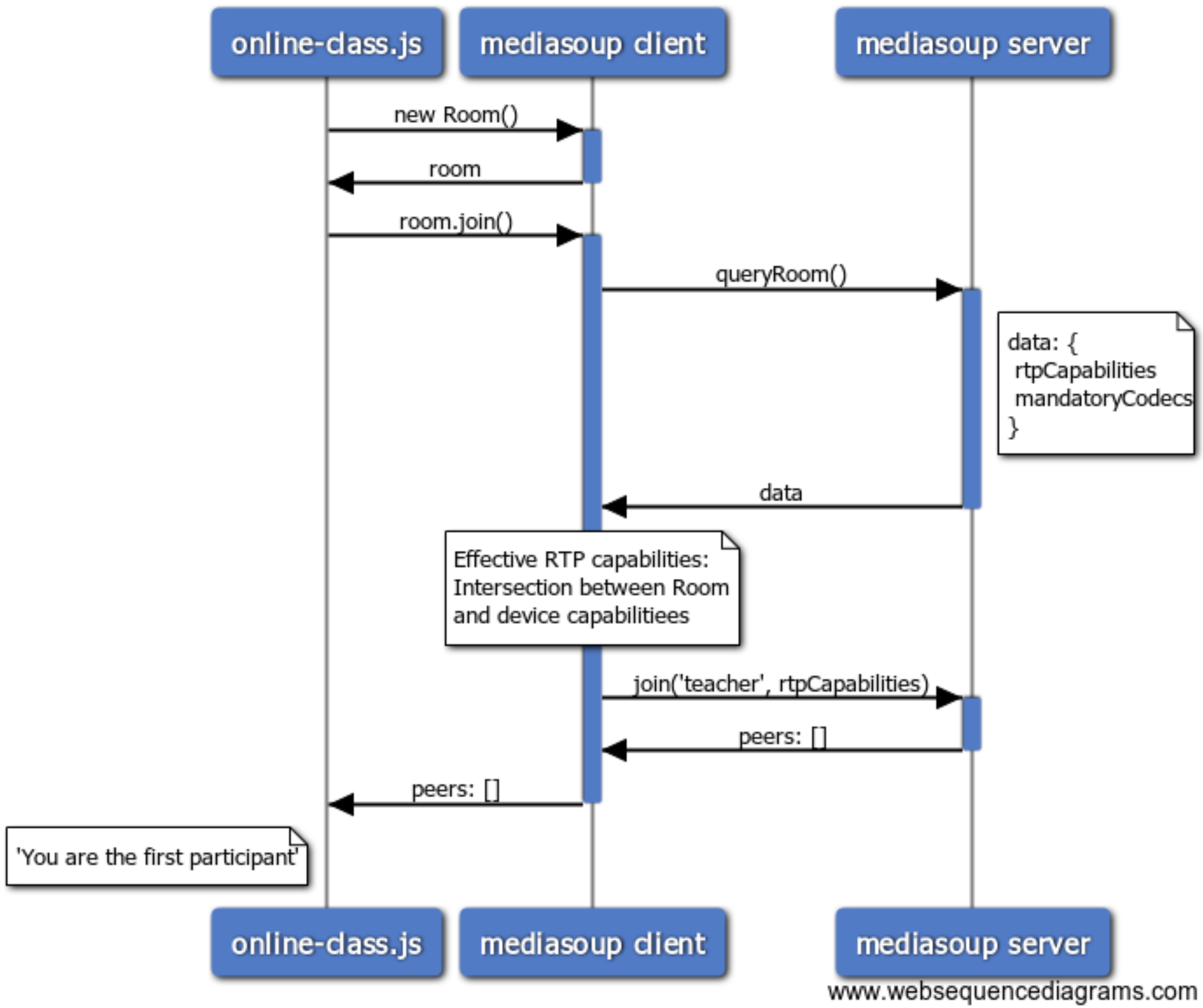
APPLICATION EXAMPLE

- Campus X has decided to offer online live classes
- Teacher talks, students listen and see the teacher’s webcam
- Students can “raise the hand” when they want to talk
  - If they are granted permission, they talk and are seen

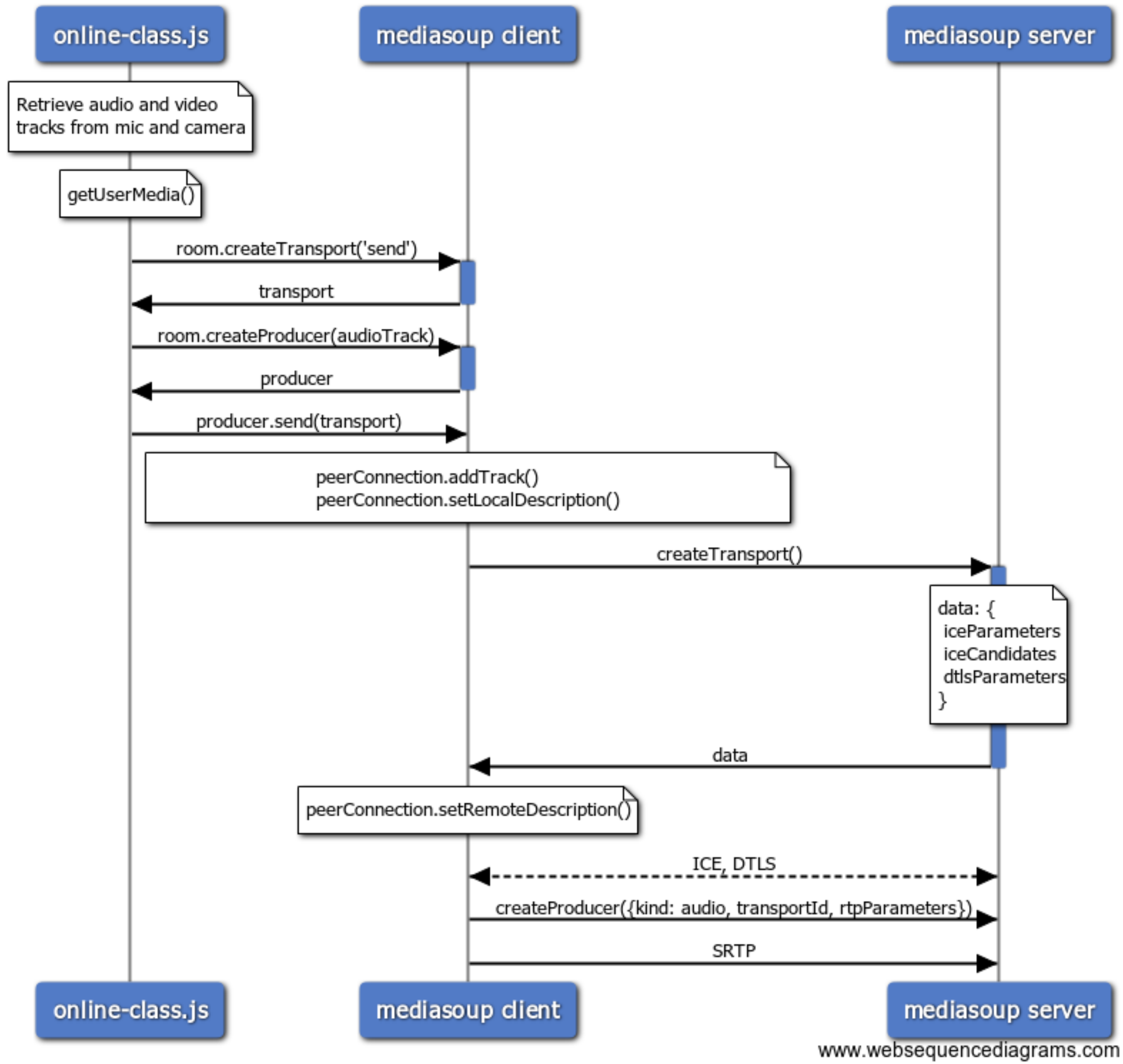
TEACHER LOGS IN THE CAMPUS AND CREATES THE 'MASTERCLASS' ROOM



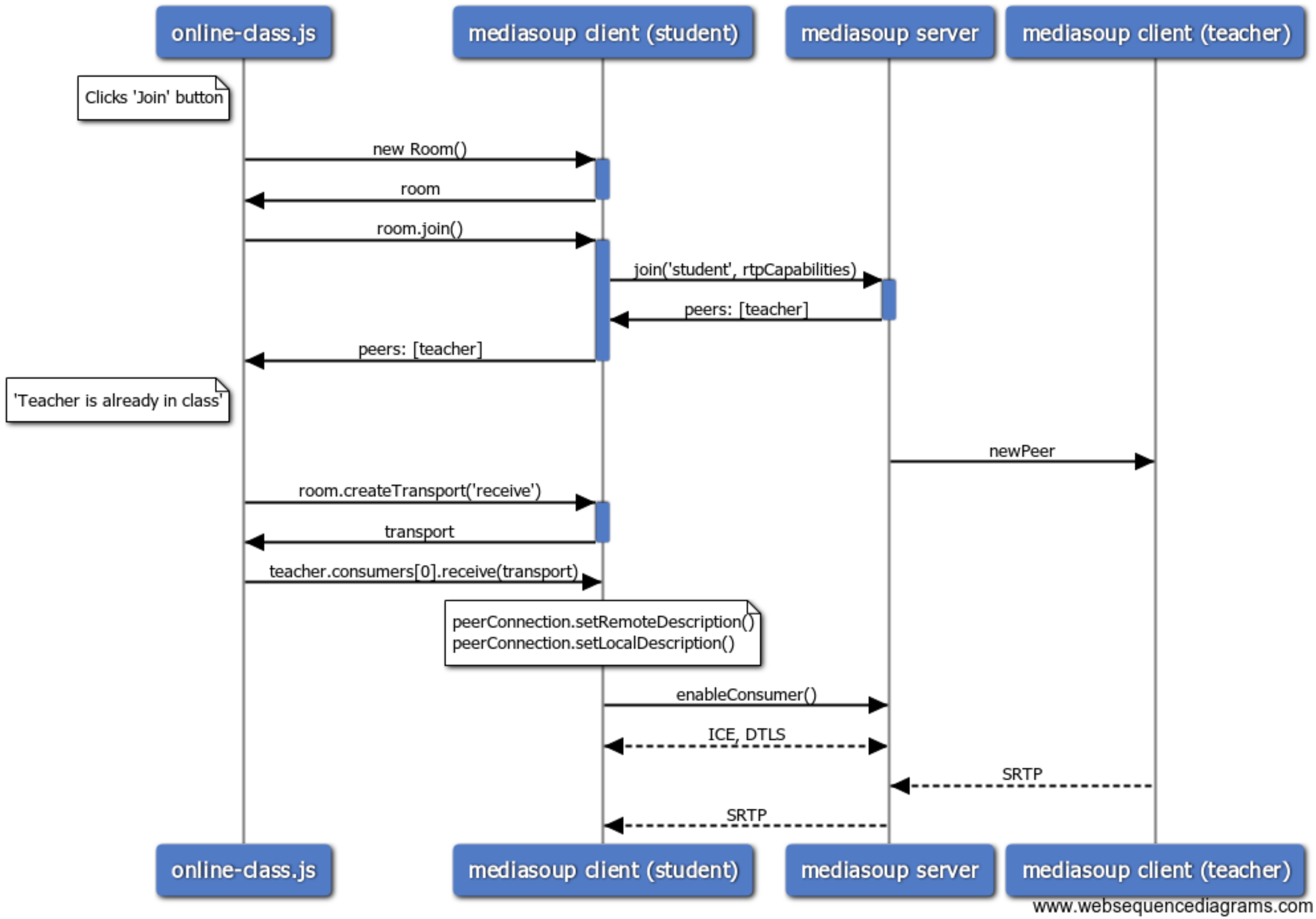
TEACHER JOINS THE ROOM



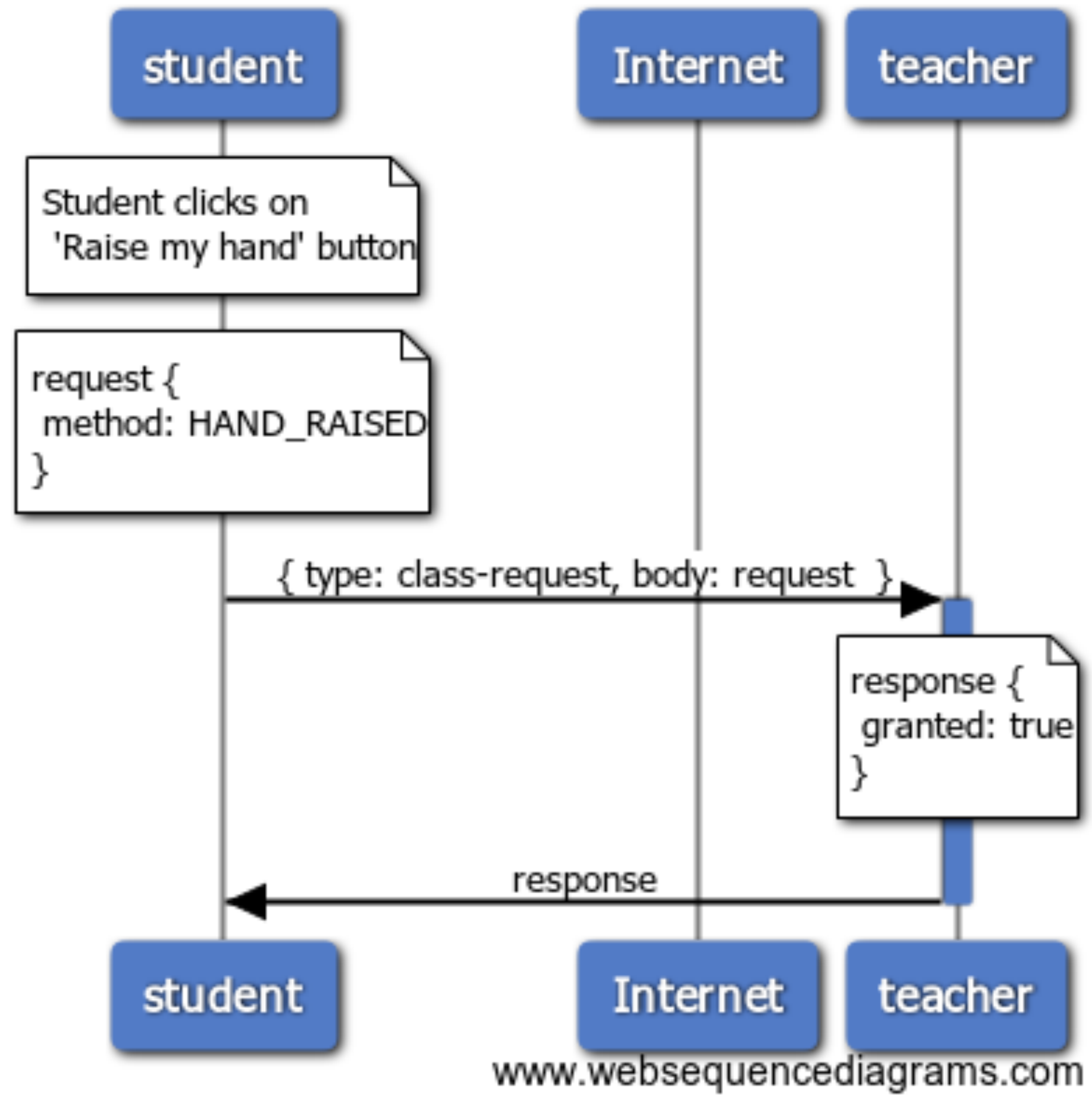
TEACHER STARTS SENDING MEDIA



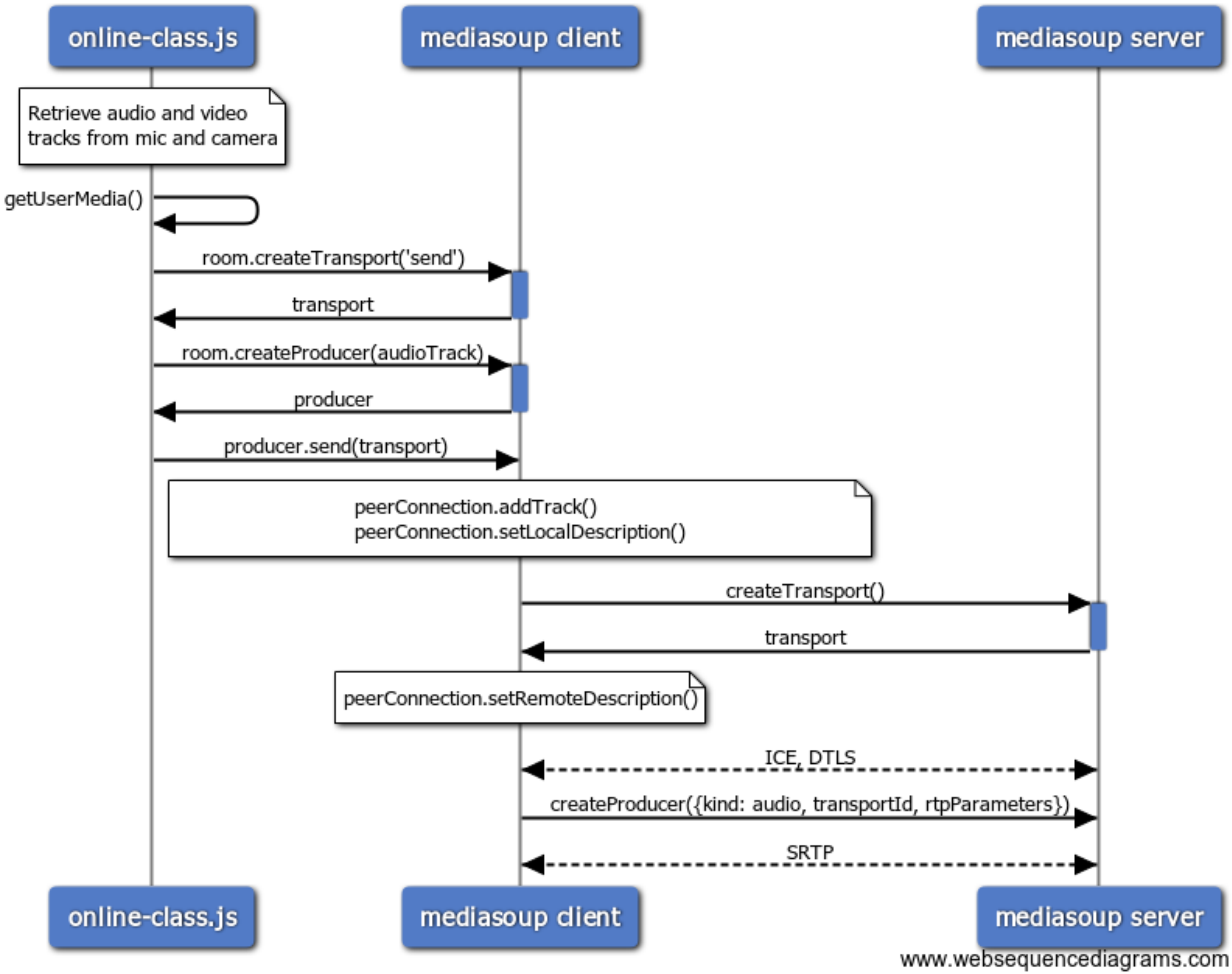
STUDENT JOINS THE ROOM AND STARTS RECEIVING TEACHER'S MEDIA



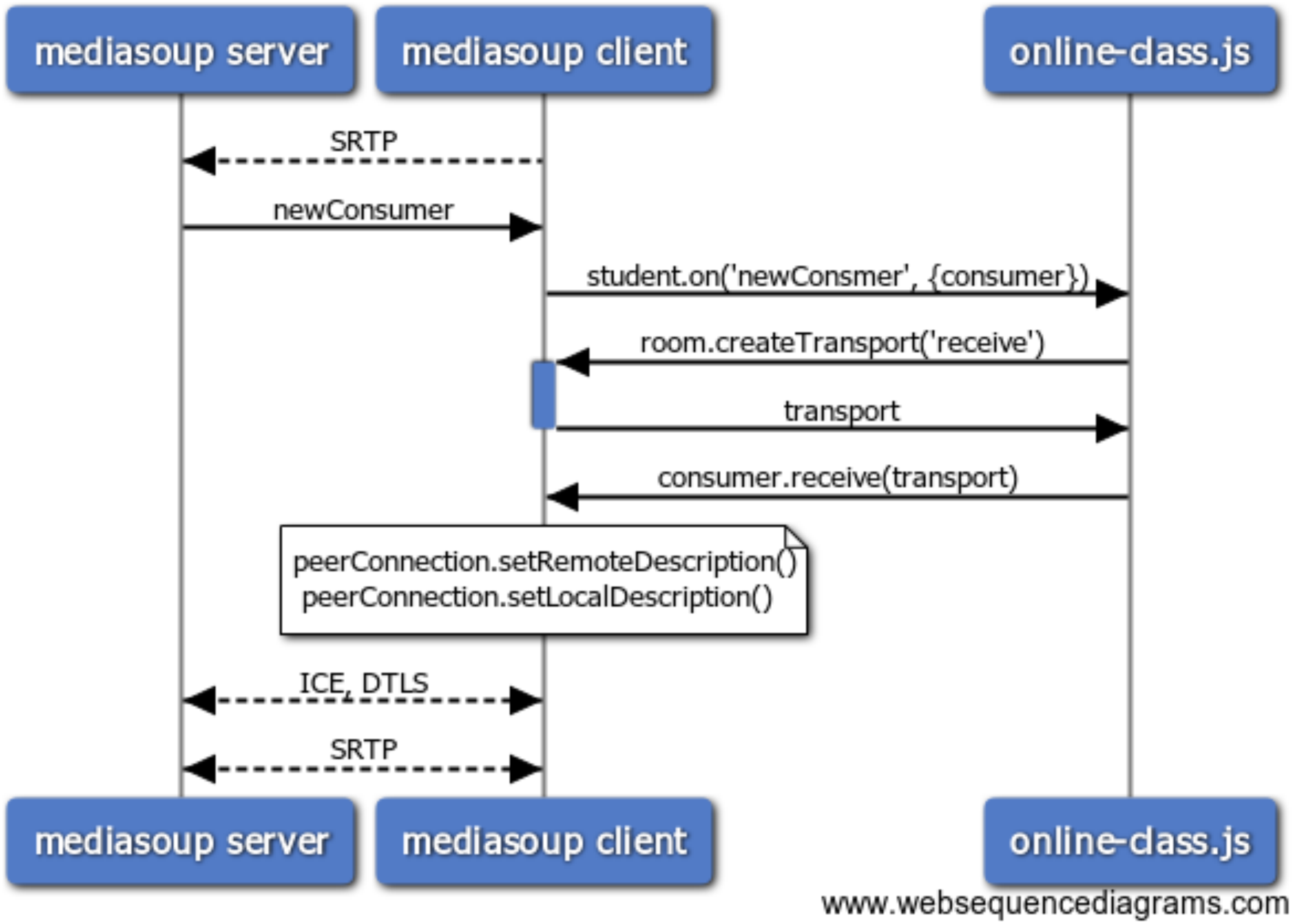
STUDENT REQUESTS PERMISSION FOR TALKING



STUDENT STARTS SENDING MEDIA



TEACHER STARTS RECEIVING STUDENT'S MEDIA



DEMO

FIRSTSIGHT

- https://firstsight.mediasoup.org
- Join using desktop or Android Chrome/Firefox