Golang to build a real-time interactive SaaS Cloud

iTutorGroup

2019.4.23 董海冰

探探 Gopher China 2019

Agenda

- What's TutorMeet+
- Why use Golang
- WebRTC Example
- Problems encountered
- About SaaS
- Q&A

What‘s TutorMeet+ ?

Teacher

Desktop

PC

PAD

phone

Virtual Classroom

TM+

Student

Parents

Teaching Materials

TutorMeet+


vipJr
26:35:02
DO check the clients' understanding by asking for questions at the end of every 1 or 2 slides
结束课程
6/25
Vocabulary
Session Report
Lesson Plan Feedback
Length
Susan M
在线学员（3）
讨论区
Amily
Tom
Edward
Ada
Linda
Ciore
Planet

Feature

- Langugage：**Golang**、C++、ReactJs；
- Video：**VP8**、VP9、**H264**、AV1；
- Audio：**OPUS**、AAC；
- Deployment：**Docker**、Kubernetes、Hybird cloud；
- Support：1v1、**1vN**、NvN（连麦）

Why use Golang？

- History ：

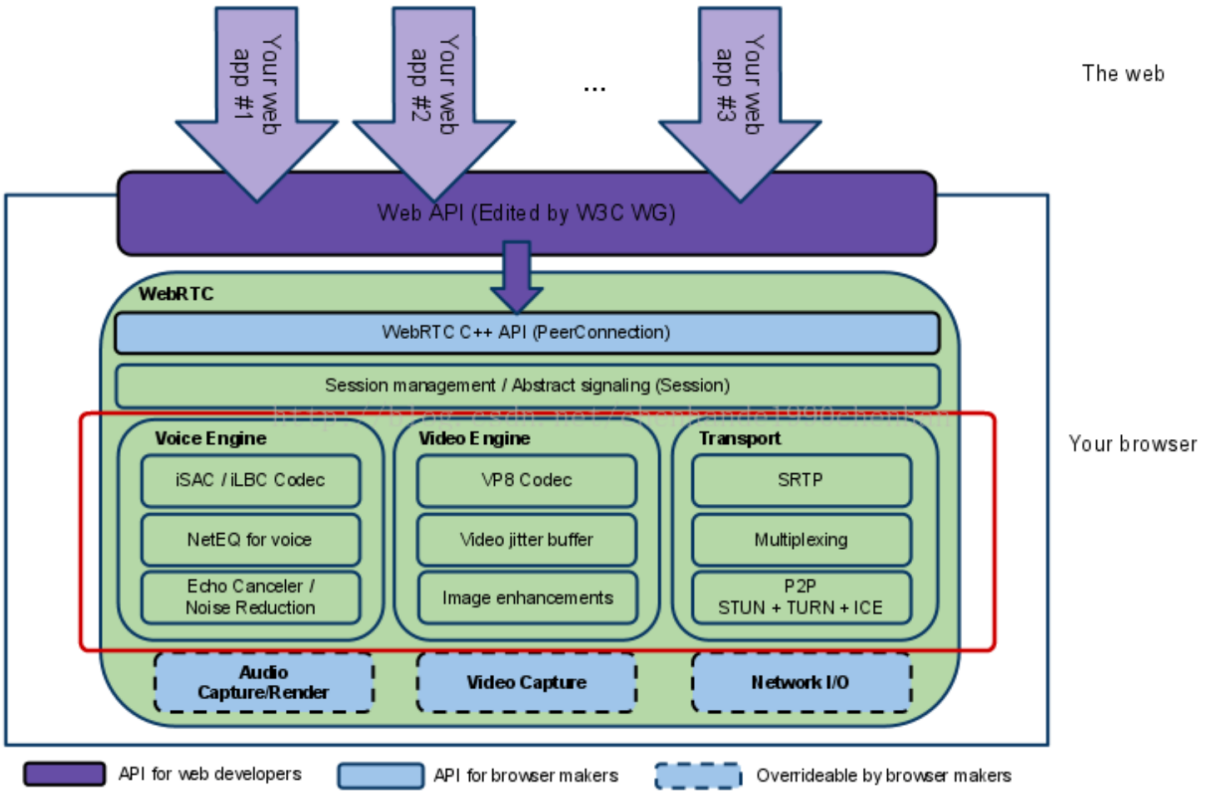
  Python or C（C++） ？

Golang advantage

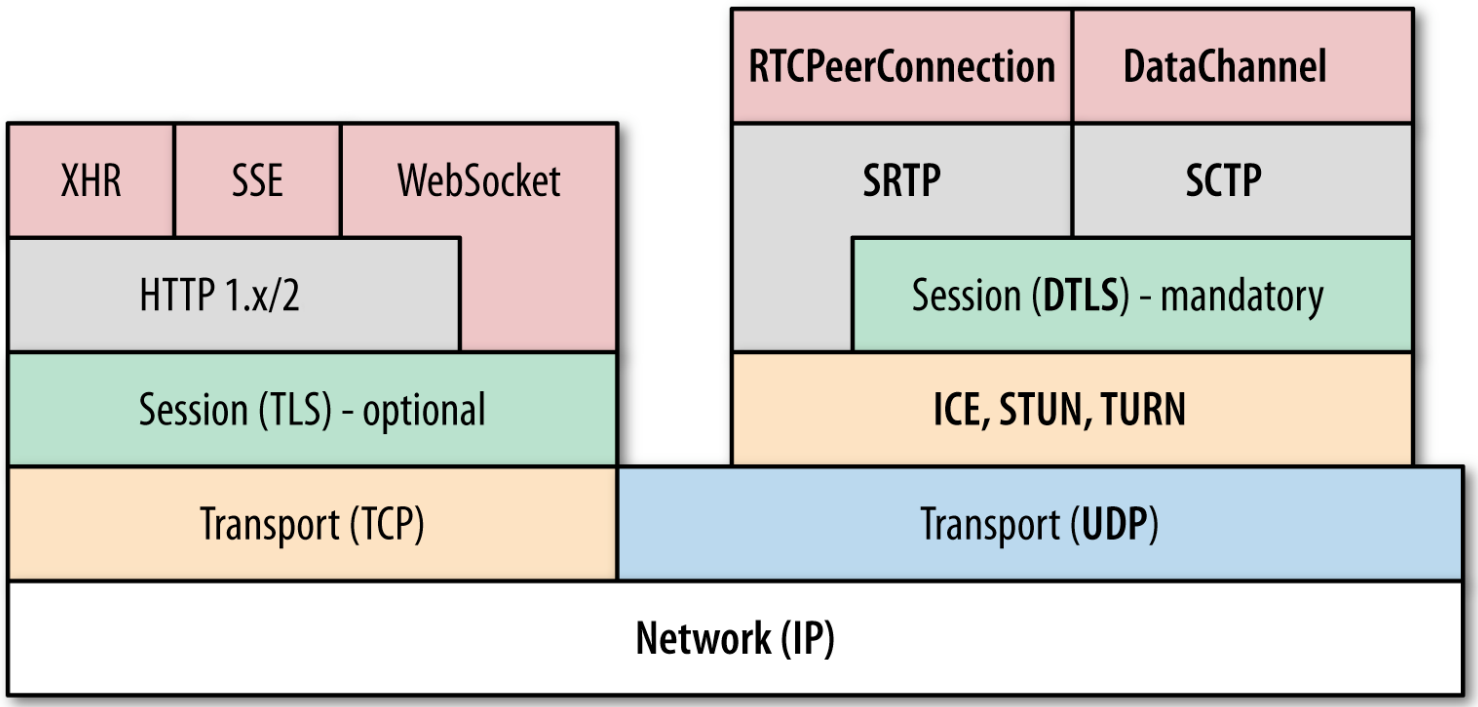
- **Efficiency（多）**
- **Performance（快）**
- **Engineering（好）**
- **Less is more（省）**

WebRTC Overview

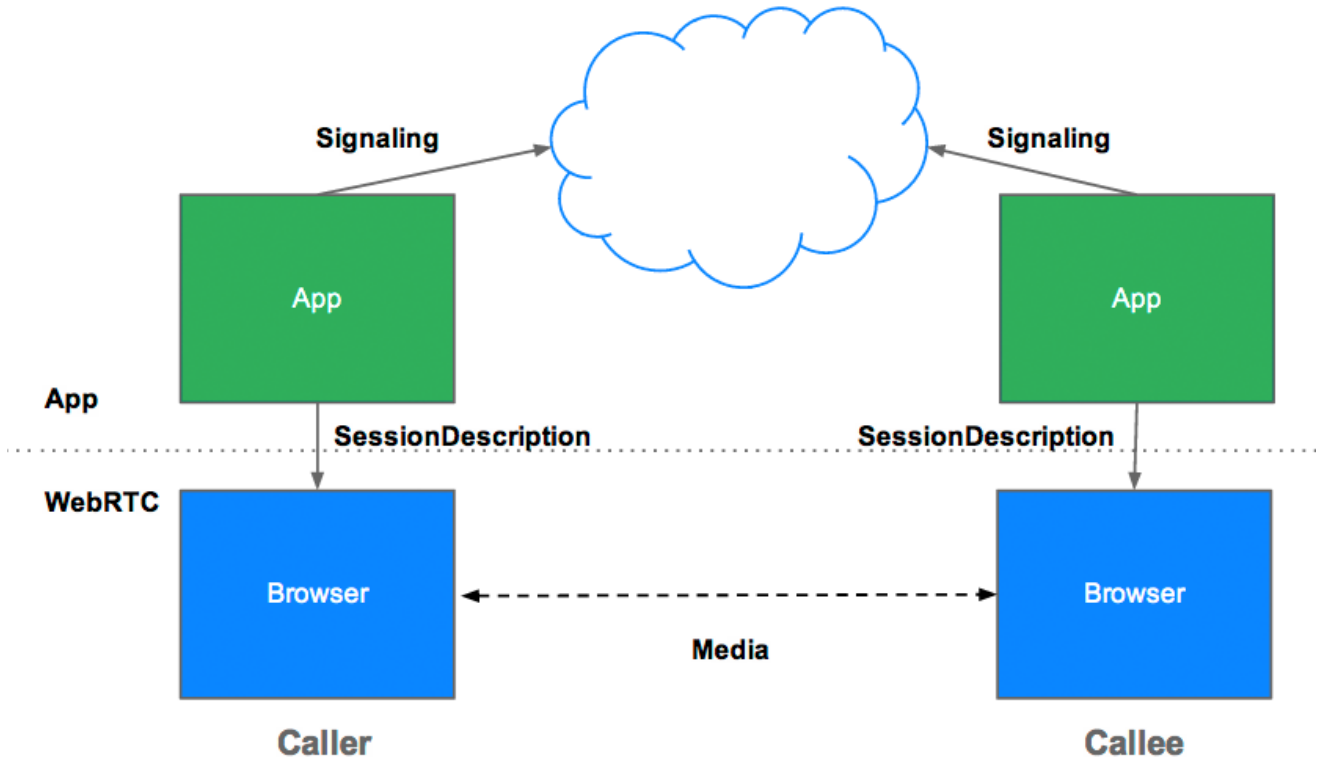
WebRTC


Your web app #1
Your web app #2
...
Your web app #3
The web
Web API (Edited by W3C WG)
WebRTC
WebRTC C++ API (PeerConnection)
Session management / Abstract signaling (Session)
Voice Engine
iSAC / iLBC Codec
NetEQ for voice
Echo Canceler / Noise Reduction
Video Engine
VP8 Codec
Video jitter buffer
Image enhancements
Transport
SRTP
Multiplexing
P2P STUN + TURN + ICE
Audio Capture/Render
Video Capture
Network I/O
Your browser
API for web developers
API for browser makers
Overrideable by browser makers

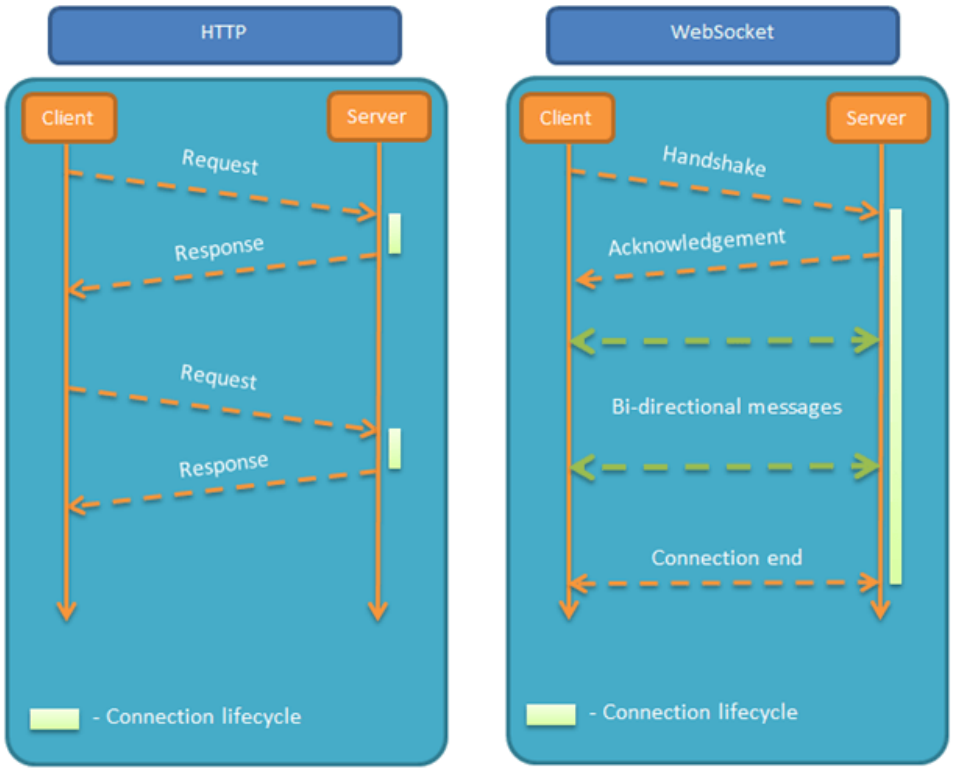
WebRTC Protocol Stack



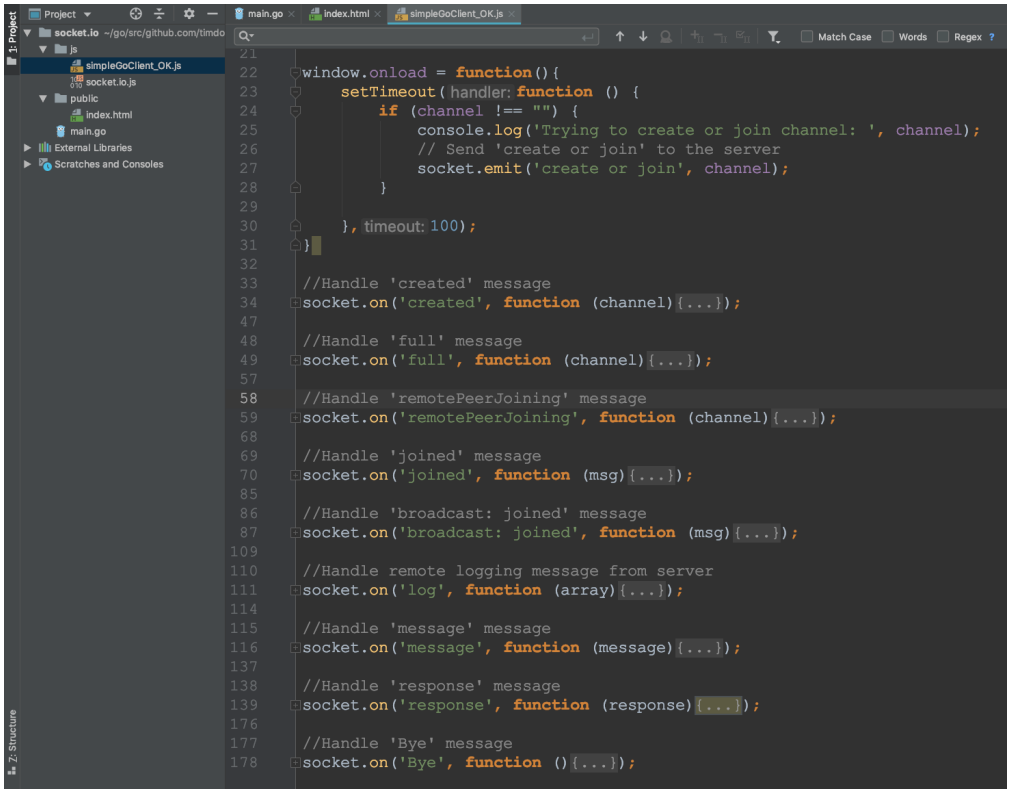
WebRTC


Signaling
Signaling
App
App
App
SessionDescription
SessionDescription
WebRTC
Browser
Browser
Media
Caller
Callee

WebSocket

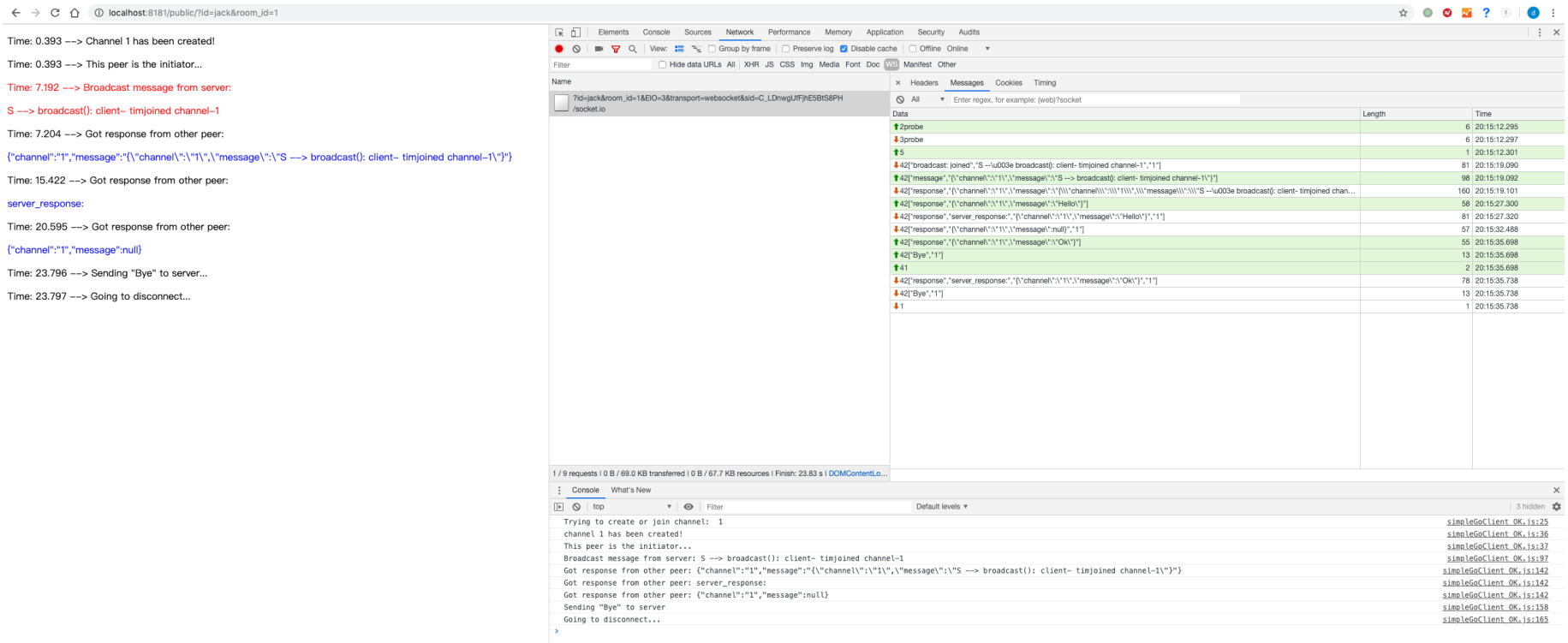

HTTP
Client
Server
Request
Response
Request
Response
- Connection lifecycle
WebSocket
Client
Server
Handshake
Acknowledgement
Bi-directional messages
Connection end
- Connection lifecycle

WebSocket Example

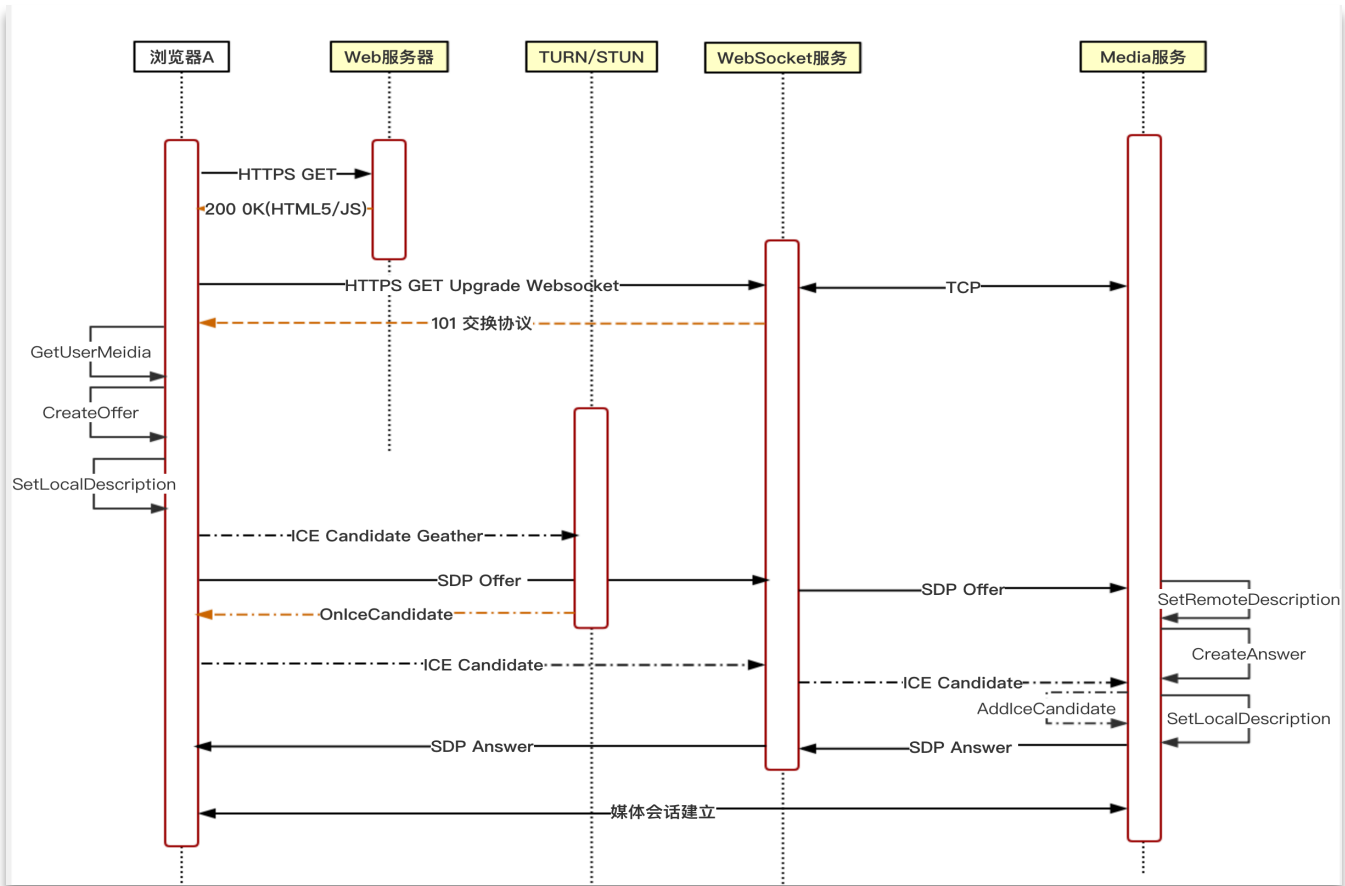



```
server, err := socketio.NewServer( transportNames: nil)
if err != nil {
    log.Fatal(err)
}
_ = server.On( event: "connection", func(so socketio.Socket) {
    //fmt.Println(" === connection === ")
    id := so.Request().FormValue( key: "id")
    channelID := so.Request().FormValue( key: "room_id")

    log.Println("on connection:" + id + "|room_id:" + channelID)

    // Handle 'message' messages
    _ = so.On( event: "message", func(message string) {
        log.Printf( format: "S --> Got message: %v", message)
        _= so.BroadcastTo(channelID,  event: "message", message,channelID)
    })

    // Handle 'create or join' messages
    err := so.On( event: "create or join", func(channel string) {
        numClients := findClientsSocket(id, namespaces)
        log.Printf( format: "numClients: %v", numClients)
        if numClients == 0 {
            so.Join(channel)
            err = so.Emit( event: "created", channel)
            if err != nil{
                log.Printf( format: "so.Emit(created) is failed! err: %v",err)
            }
            log.Printf( format: "so.Emit -> channel[%v]  success!",channel)
        } else if numClients == 1 {

            err = so.Emit( event: "remotePeerJoining", channel)
            if err != nil {
                log.Printf( format: "so.Emit(remotePeerJoining) is failed! roo
            }
            so.Join(channel)
            err = so.BroadcastTo(channel,  event: "broadcast: joined", "S -->
            if err != nil {
                log.Printf( format: "so.BroadcastTo(broadcast: joined) is fail
            }
        }else {
            log.Println( v...: "Channel full!")
            so.Emit( event: "full", channel)
        }
```

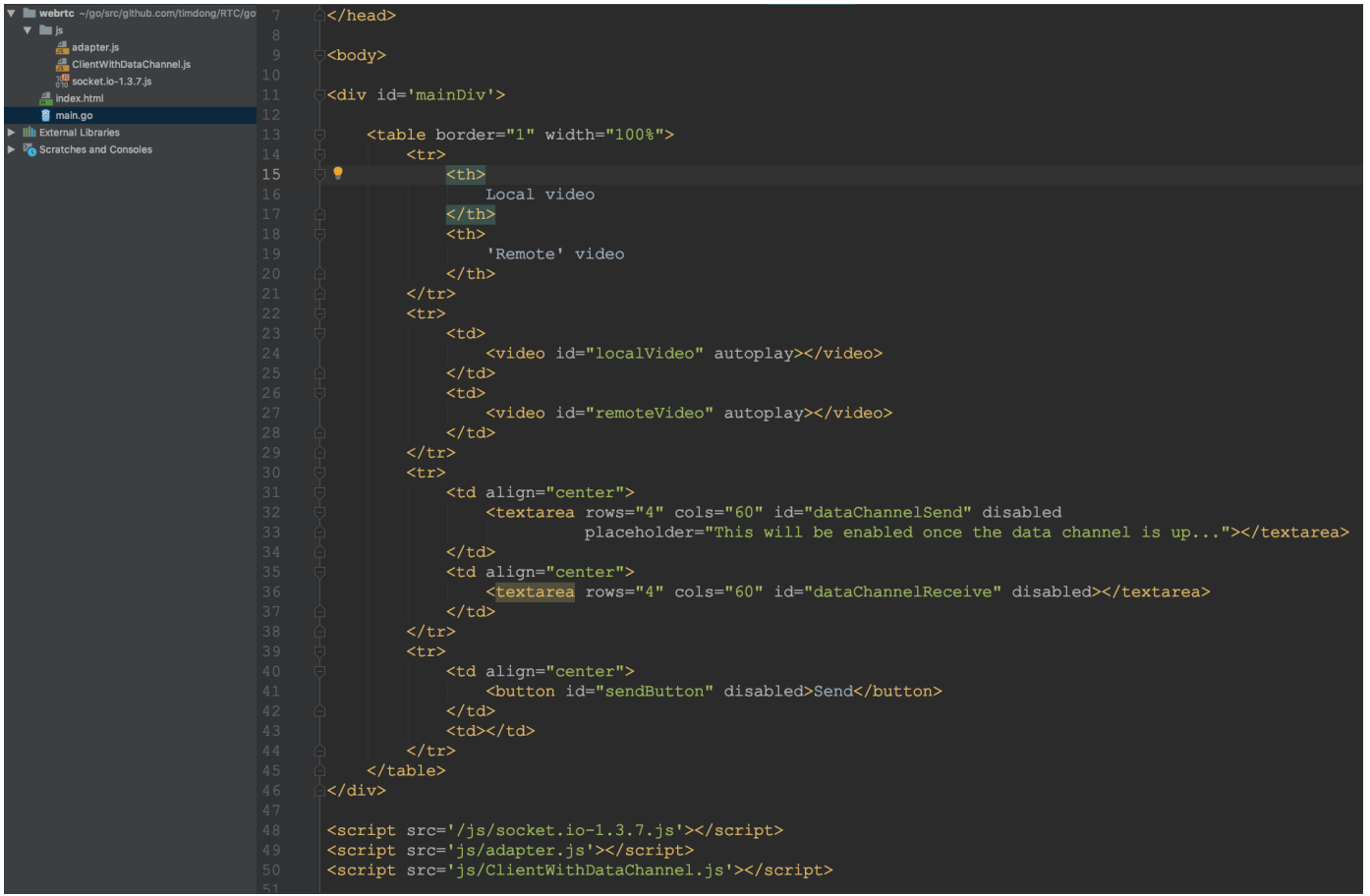

WebSocket Example



WebRTC Flow


浏览器A
Web服务器
TURN/STUN
WebSocket服务
Media服务
HTTPS GET
200 0K(HTML5/JS)
HTTPS GET Upgrade Websocket
TCP
101 交换协议
GetUserMeidia
CreateOffer
SetLocalDescription
ICE Candidate Geather
SDP Offer
SDP Offer
SetRemoteDescription
OnIceCandidate
CreateAnswer
ICE Candidate
ICE Candidate
AddIceCandidate
SetLocalDescription
SDP Answer
SDP Answer
媒体会话建立

WebRTC Example:Front end




```
</head>

<body>

<div id='mainDiv'>

    <table border="1" width="100%">
        <tr>
            <th>
                Local video
            </th>
            <th>
                'Remote' video
            </th>
        </tr>
        <tr>
            <td>
                <video id="localVideo" autoplay></video>
            </td>
            <td>
                <video id="remoteVideo" autoplay></video>
            </td>
        </tr>
        <tr>
            <td align="center">
                <textarea rows="4" cols="60" id="dataChannelSend" disabled
                          placeholder="This will be enabled once the data channel is up..."></textarea>
            </td>
            <td align="center">
                <textarea rows="4" cols="60" id="dataChannelReceive" disabled></textarea>
            </td>
        </tr>
        <tr>
            <td align="center">
                <button id="sendButton" disabled>Send</button>
            </td>
            <td></td>
        </tr>
    </table>
</div>

<script src='/js/socket.io-1.3.7.js'></script>
<script src='js/adapter.js'></script>
<script src='js/ClientWithDataChannel.js'></script>
```


WebRTC Example:Front end

```
// 1. Server-->Client...
////////////////////////////////////////////

// Handle 'created' message coming back from server:
// this peer is the initiator
socket.on( event: 'created',  fn: function (room){
    console.log('Created room :: ' + room);
    isInitiator = true;
});

// Handle 'full' message coming back from server:
// this peer arrived too late :-(
socket.on( event: 'full',  fn: function (room){
    console.log('Room ' + room + ' is full');
});

// Handle 'join' message coming back from server:
// another peer is joining the channel
socket.on( event: 'join',  fn: function (room){
    console.log('Another peer made a request to join room ' + room);
    console.log('This peer is the initiator of room ' + room + '!');
    isChannelReady = true;
});

// Handle 'joined' message coming back from server:...
socket.on( event: 'joined',  fn: function (room){
    console.log('This peer has joined room ' + room);
    isChannelReady = true;
});

// Server-sent log message...
socket.on( event: 'log',  fn: function (array){
    console.log.apply(console, array);
});

// Receive message from the other peer via the signalling server
socket.on( event: 'message',  fn: function (message){...});
////////////////////////////////////////////////

// 2. Client-->Server
////////////////////////////////////////////////
```

```
// Receive message from the other peer via the signalling server
socket.on('message', function (message){
   console.log('Received message:', message);

   if (isJsonString(message)){
      message = JSON.parse(message)
   }else{
      message =  message.replace(/^\"|\"$/g,'');
   }

   if (message === 'got user media') {
      checkAndStart();
   } else if (message.type === 'offer') {
      if (!isInitiator && !isStarted) {
         checkAndStart();
      }
      pc.setRemoteDescription(new RTCSessionDescription(message));
      doAnswer();
   } else if (message.type === 'answer' && isStarted) {
      pc.setRemoteDescription(new RTCSessionDescription(message));
   } else if (message.type === 'candidate' && isStarted) {
      var candidate = new RTCIceCandidate({sdpMLineIndex:message.label,
         candidate:message.candidate});
      pc.addIceCandidate(candidate);
   } else if (message === 'bye' && isStarted) {
      handleRemoteHangup();
   }
});
```


WebRTC Example:Back end

```go
package main

import (
    "github.com/googollee/go-socket.io"
    "log"
    "net/http"
)

func main() {
    namespaces := make(map[string]int)
    server, err := socketio.NewServer(transportNames: nil)
    if err != nil {
        log.Fatal(err)
    }
    _ = server.On(event: "connection", func(so socketio.Socket) {...})

    //http.Handle("/socket.io/", server)
    http.HandleFunc(pattern: "/socket.io/", func(w http.ResponseWriter, r *http.Request) {
        server.ServeHTTP(w, r)
    })
    http.Handle(pattern: "/", http.FileServer(http.Dir("./")))
    http.Handle(pattern: "/js", http.FileServer(http.Dir("./js")))
    log.Println(v...: "Serving at localhost:8282...")
    log.Fatal(http.ListenAndServe(addr: ":8282", handler: nil))

}

func findClientsSocket(cID string, namespace map[string]int) int {...}
```

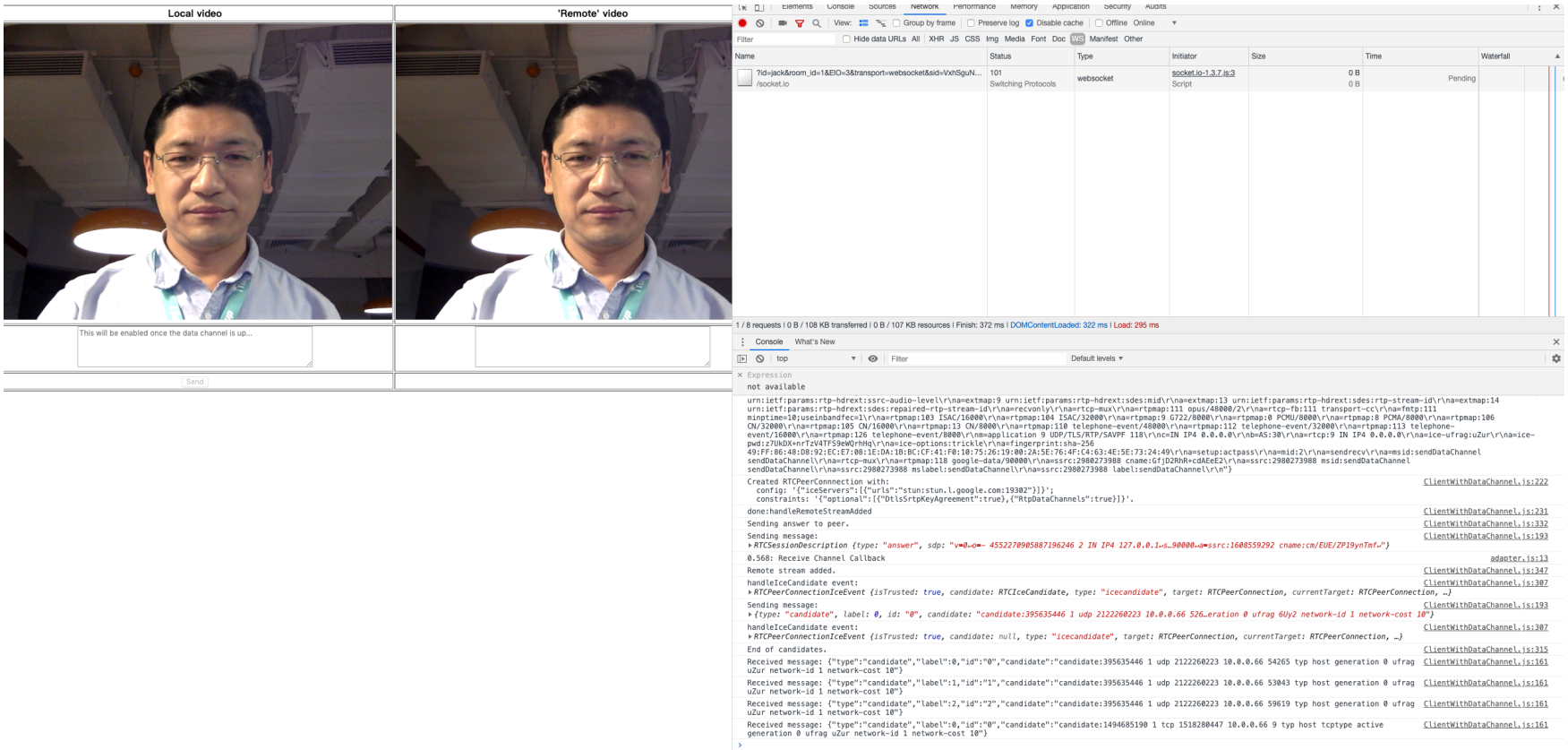
```go
    _ = server.On(event: "connection", func(so socketio.Socket) {
        id := so.Request().FormValue(key: "id")
        // Handle 'message' messages
        _ = so.On(event: "message", func(message string) {
            log.Printf("S --> Got message: #{message}")
            so.BroadcastTo(room: "1", event: "message", message)
        })
        // Handle 'create or join' messages
        err := so.On(event: "create or join", func(room string) {
            numClients := findClientsSocket(id, namespaces)
            log.Printf(format: "numClients: %v", numClients)
            log.Printf(format: "S --> Room : %v  has  %v client(s)", room, numClients)
            log.Printf(format: "S --> Request to create or join room: %v ", room)
            // First client joining...
            if numClients == 0 {
                log.Printf(format: "First client joining...: %v", numClients)
                err = so.Join(room)
                if err != nil {
                    log.Printf(format: "so.Join[0] failed! err: %v", err)
                }
                log.Println(v...: "First client joined")

                err = so.Emit(event: "created", room)
                if err != nil {
                    log.Printf(format: "so.Emit[0] created failed.  err: %v", err)
                }

                log.Println(v...: "First client :Created")
                // Second client joining...
            } else if numClients == 1 {
                err = so.Emit(event: "join", room)
                if err != nil {
                    log.Printf(format: "so.Emit[1] send 'join' failed!  err:%v", err)
                }
                err = so.Join(room)
                if err != nil {
                    log.Printf(format: "so.Join[1] join failed! err: %v", err)
                }
                err = so.BroadcastTo(room, event: "join", room)
                if err != nil {
                    log.Println(v...: "so.BroadcastTo[1] room join failed!")
                }
                err = so.Emit(event: "joined", room)
```

main() › func(so socketio.Socket) › func(message string)

Debug Tools

chrome://webrtc-internals/

Demo Result


Local video
'Remote' video
This will be enabled once the data channel is up...
Send
Elements Console Sources Network Performance Memory Application Security Audits
Name Status Type Initiator Size Time Waterfall
?id=jack&room_id=1&EIO=3&transport=websocket&sid=VxhSguN... 101 Switching Protocols websocket socket.io-1.3.7.js:3 Script 0 B 0 B Pending
1 / 8 requests | 0 B / 108 KB transferred | 0 B / 107 KB resources | Finish: 372 ms | DOMContentLoaded: 322 ms | Load: 295 ms
Console What's New
not available
Created RTCPeerConnnection with:
done:handleRemoteStreamAdded
Sending answer to peer.
Sending message:
0.568: Receive Channel Callback
Remote stream added.
handleIceCandidate event:
Sending message:
handleIceCandidate event:
End of candidates.
Received message: {"type":"candidate","label":0,"id":"0","candidate":"candidate:395635446 1 udp 2122260223 10.0.0.66 54265 typ host generation 0 ufrag uZur network-id 1 network-cost 10"}
Received message: {"type":"candidate","label":1,"id":"1","candidate":"candidate:395635446 1 udp 2122260223 10.0.0.66 53043 typ host generation 0 ufrag uZur network-id 1 network-cost 10"}
Received message: {"type":"candidate","label":2,"id":"2","candidate":"candidate:395635446 1 udp 2122260223 10.0.0.66 59619 typ host generation 0 ufrag uZur network-id 1 network-cost 10"}
Received message: {"type":"candidate","label":0,"id":"0","candidate":"candidate:1494685190 1 tcp 1518280447 10.0.0.66 9 typ host tcptype active generation 0 ufrag uZur network-id 1 network-cost 10"}

SDP

```
....
o=- 20518 0 IN IP4 0.0.0.0
……
a=ice-options:trickle
a=group:BUNDLE audio video

m=audio 54609 UDP/TLS/RTP/SAVPF 109 0 8
a=mid:audio
a=sendrecv
a=rtpmap:109 opus/48000/2
…….
a=candidate:0 1 UDP 2122194687 192.168.1.4 61665 typ host
a=candidate:1 1 UDP 1685987071 24.23.204.141 54609 typ srflx raddr
192.168.1.4 rport 61665
……
```

来源信息 用户 会话id 版本 id 网络类型 ip类型 ip

采用trickle方式
媒体数据包含音频和视频

媒体信息描述

[RFC3264]can send and recv audio
[I-D.ietf-payload-rtp-opus] - Opus Codec 48khz, 2 channels

[RFC5245] - ICE Host Candidate
[RFC5245] - ICE Server Reflexive Candidate

会话描述

```
1  v = (协议版本)
2  o = (发起者和会话标识符)
3  s = (会话名称)
4  i = * (会话信息)
5  u = * (描述的URI)
6  e = * (电子邮件地址)
7  p = * (电话号码)
8  c = * (连接信息 – 如果包含在内，则不需要所有媒体)
9  b = * (零个或多个带宽信息行)
10
11 一个或多个时间描述 ("t ="和"r ="行;见下文)
12 z = * (时区调整)
13 k = * (加密密钥)
14 a = * (零个或多个会话属性行) 零个或多个媒体描述
```

时间描述

```
1  t = (会话活动时间)
2  r = * (零个或多个重复次数)
```

媒体描述，如果存在

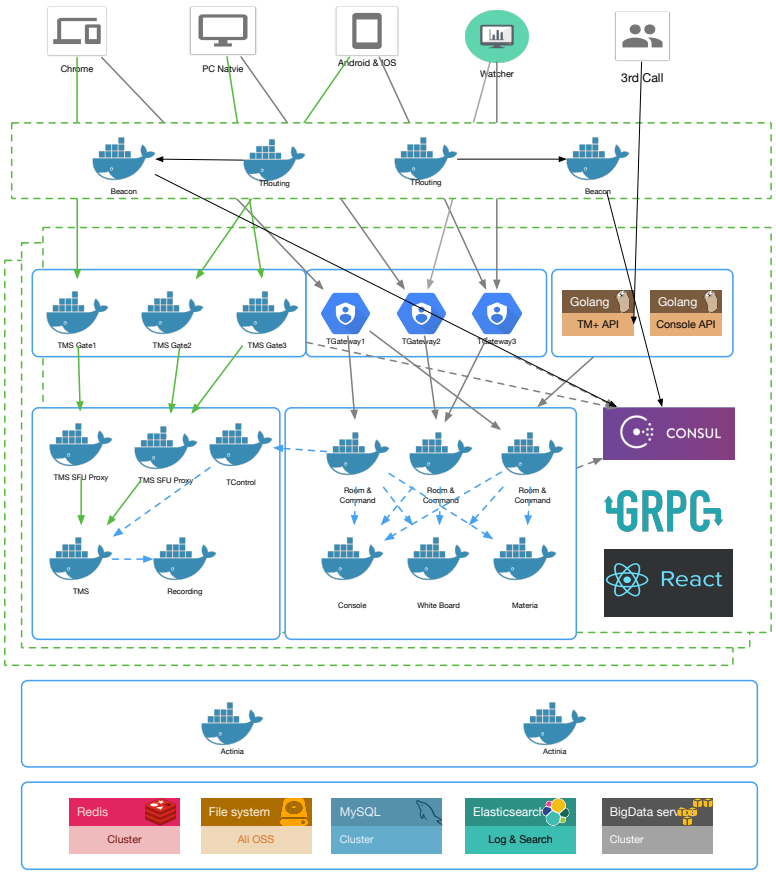
```
1  m = (媒体名称和传输地址)
2  i = * (媒体标题)
3  c = * (连接信息 – 如果包含在内，则为可选项会话级别)
4  b = * (零个或多个带宽信息行)
5  k = * (加密密钥)
6  a = * (零个或多个媒体属性行)
```


WebRTC Server

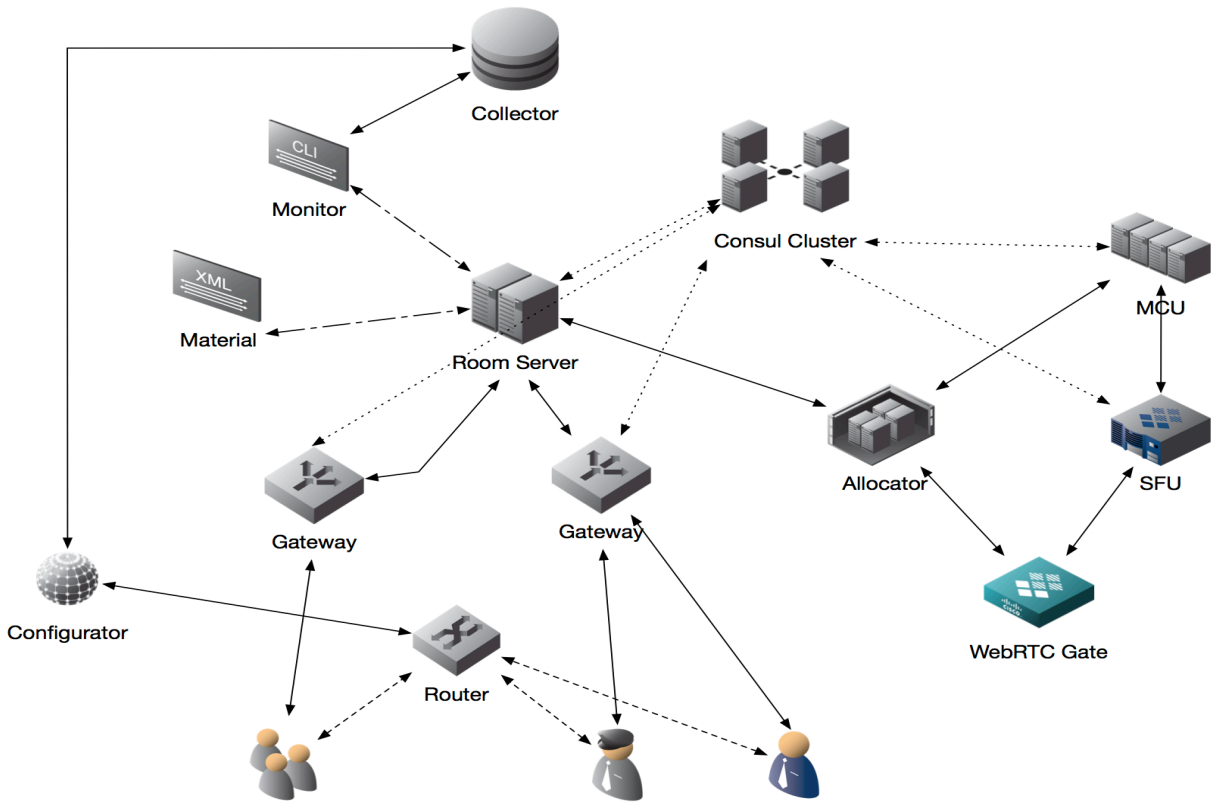
Open Source Media Server :

- lynckia/licode : github.com/lynckia/licode/ 【C++】
- meetecho (janus) : github.com/meetecho/janus-gateway【C】
- jitsi (meetme) : github.com/jitsi/jitsi 【Java】
- Kurento : github.com/Kurento/kurento-media-server【C++】

TM+ Architecture I

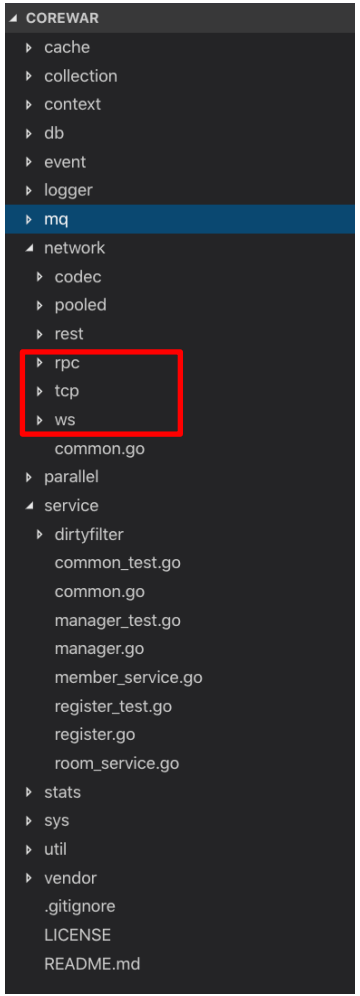


TM+ Architecture II



Go Lib

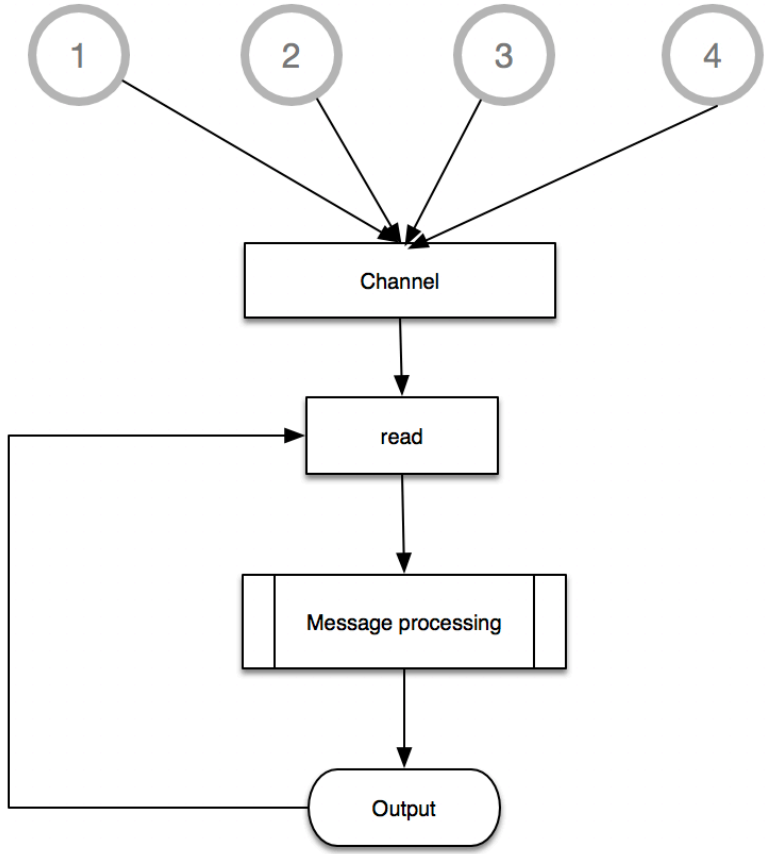
- gRPC 、TCP
- Websocket 、 Socket.IO
- Gin
- Redigo(Redis)
- Mgo(MongoDB)
- Consul
- ……



Problem I

loop issue

```
for {
    select {
    case msg := <- stream:
        handle(msg)  // message处理
    }
}
```



Problem I

loop issue

Solution A :

```
for {
    select {
    case msg := <-stream:
       messages := []Message{msg}
       for i := 0; i < len(stream); i++ {
           messages = append(messages, <-
stream)
       }
       handles(messages) // 批处理
    }
}
```


Problem I

loop issue

Solution B :

```
sema := make(chan *Message, 16) //
for {
    select {
    case msg := <-stream:
        sema <- msg
        go func(sem <-chan *Message) {
            handle(<-sem)
        }(sema)
    }
}
```


Problem II

- **Go Time :**

- p := fmt.Println
  thisday,_ := time.Parse(time.*RFC3339*,"2018-11-30T14:00:00+08:00")
  tomorrow,_ := time.Parse(time.*RFC3339*,"2018-12-01T14:00:00+08:00")
  time2 := thisday.Format("2018-01-01 00:00:00")
  p(time2)
  time3 := tomorrow.Format("2018-01-01 00:00:00")
  p(time3)

- Result:
- time2 =30118-11-11 00:00:001128-12-12 00:00:00
- time3 =1128-12-12 00:00:00

Problem II

Go Time :

Why “2006-01-02 15:04:05”

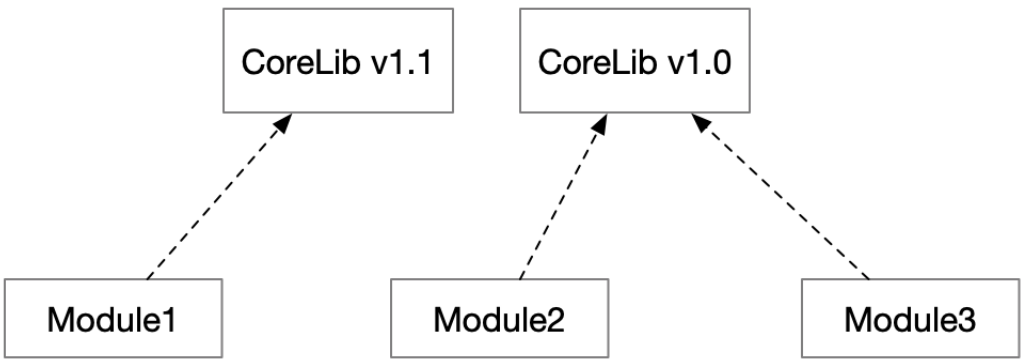
2006年1月2日3点4分5秒~

Format:

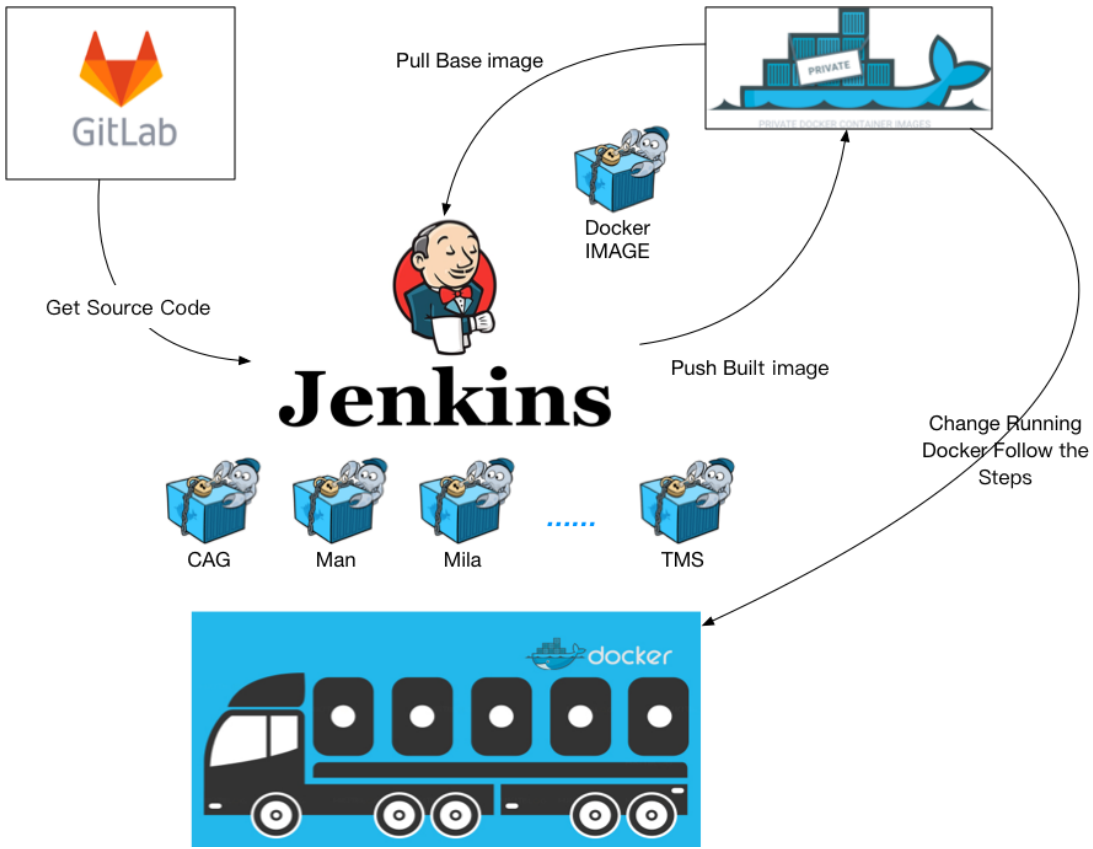
2006-01-02 15:04:05.999999999 -0700 MST

Problem III

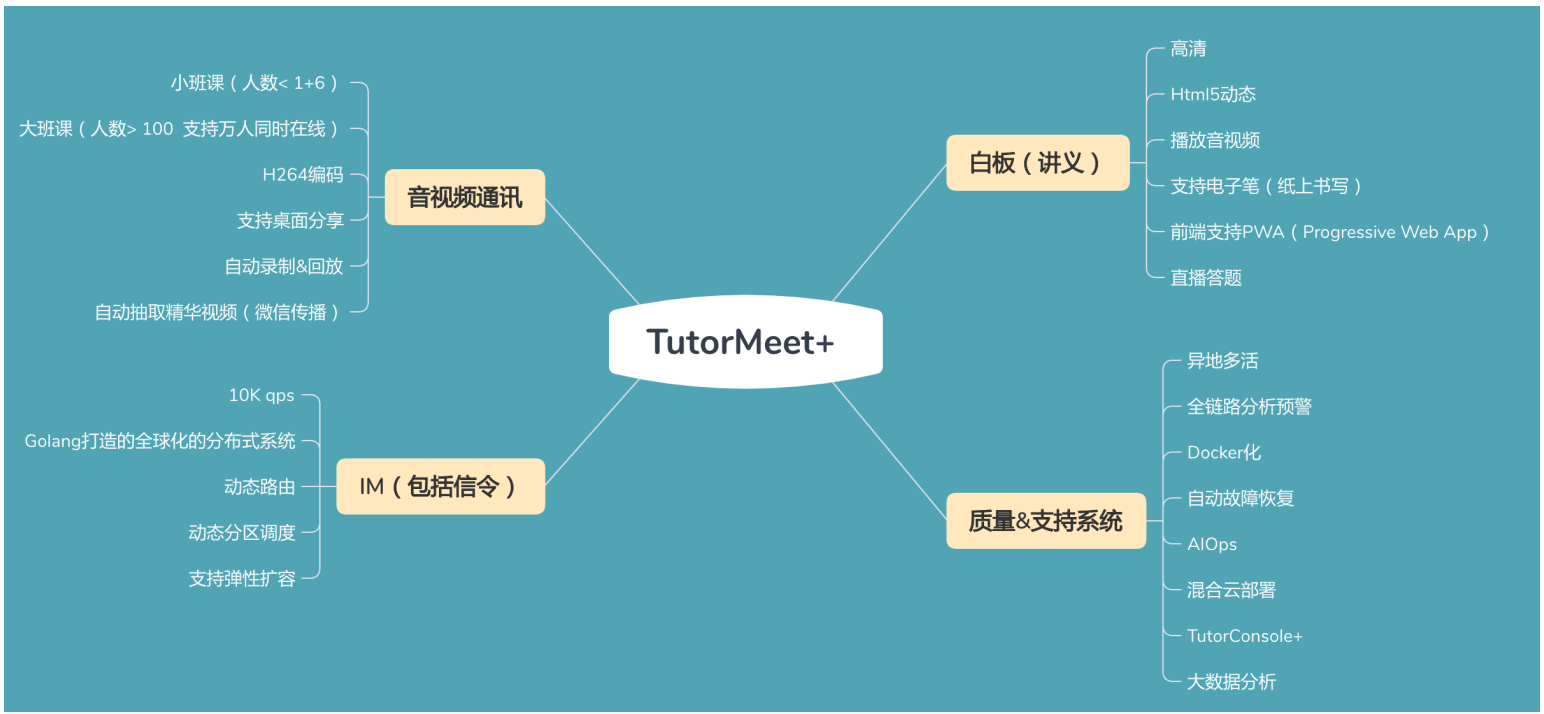
- Go module:



TM+ Deployment



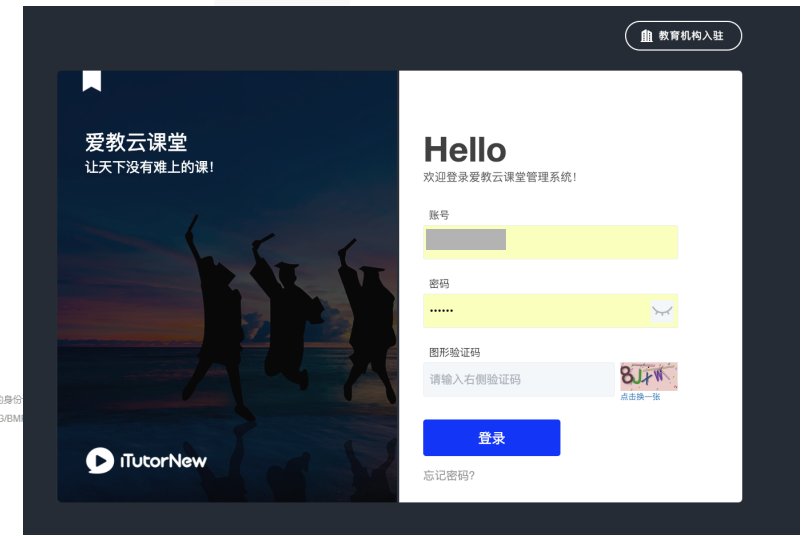
TM+ Module


TutorMeet+
音视频通讯
小班课（人数< 1+6）
大班课（人数> 100 支持万人同时在线）
H264编码
支持桌面分享
自动录制&回放
自动抽取精华视频（微信传播）
白板（讲义）
高清
Html5动态
播放音视频
支持电子笔（纸上书写）
前端支持PWA（Progressive Web App）
直播答题
IM（包括信令）
10K qps
Golang打造的全球化的分布式系统
动态路由
动态分区调度
支持弹性扩容
质量&支持系统
异地多活
全链路分析预警
Docker化
自动故障恢复
AIOps
混合云部署
TutorConsole+
大数据分析

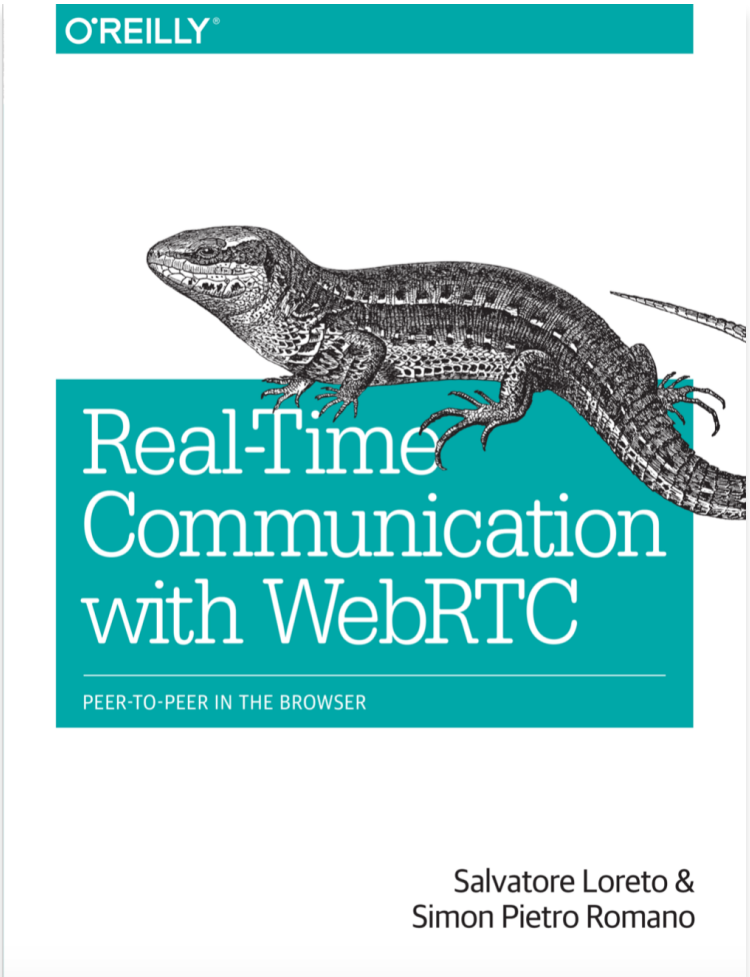
SaaS

Real-Time Interaction

SaaS



机构端：https://open.cntutormeet.com/hlogin.html
教师端：https://open.cntutormeet.com/tlogin.html
学生端：微信小程序，在微信中搜索小程序“爱教云课堂”
或微信扫描下方二维码：


O'REILLY
Real-Time Communication with WebRTC
PEER-TO-PEER IN THE BROWSER
Salvatore Loreto &
Simon Pietro Romano

Thanks !

Q&A