Mobile based languages learning application

Le Ngoc Hieu – Nguyen Nhat Quang- Do Hoang Nguyen- Truong Thanh Lam – Nguyen Huy Hung
FPT University, Software Engineering
Ho Chi Minh city, Vietnam
Nguyen Huy Hung (Instructor)
hungnh3@fe.fpt.edu.vn
Le Ngoc Hieu (Leader)
hieulnse61723@fpt.edu.vn
Nguyen Nhat Quang (Contributor)
quangnnse61472@fpt.edu.vn
Do Hoang Nguyen (Contributor)
nguyendhse61556@fpt.edu.vn
Truong Thanh Lam (Contributor)
lamttse61257@fpt.edu.vn

Abstract

The world's largest taxi company, Uber, owns no cars. The world's most popular media company, Facebook, creates no content. The world's most valuable retailer, Alibaba, carries no stock. And the world's largest accommodation provider, Airbnb, owns no property. The app for learning languages, MOLA, own no schools, teachers or students, every single user is a source of knowledge. There are almost 3.2 billions people around the world connected to the internet (data in 2015), so the needs to learn, to study, and to be teach are very large. MOLA have focused on provide a system help users can learn and teach each other in any languages by stream video one to one.

MOLA provided tools help teacher can construct their courses (follow by structured or follow by topic), manage learner, manage schedule and earn money from knowledge they are selling.

For example, a foreigner want to travel in Vietnam, so he/she want to learn a few of sentences to talk with Vietnamese. MOLA will find the best native Vietnamese teacher for the foreigner to practice. Teachers and learner communicate and use tools MOLA provided for select the suitable time and practice by video call one to one.

Keywords

WebRTC, Firewall[1], NATs [2], STUN [3], TURN [4], Collective Intelligence, Collaborative Filtering, Redis

I. INTRODUCTION

MOLA aims to provide people around the world with a supported environment for learning languages. To achieve that goal, the application has to not only perform its business activities such as course management, scheduling and course registering, but also assist users in certain unique activities. On learning time, MOLA concentrates on connecting the teacher and the learner via one to one video call. There are more problems around this function, as many features affect it, especially on mobile application environment.

II. PROBLEM AND SOLUTION PLAN

When implement the MOLA system, we faded two big problems

1. Buy or build?

Video call is the mandatory function, so the system must be implement these things:

- Getting access input devices like microphone, webcam, camera to communicate audio and video.
- Not just audio and video, application should communicating arbitrary data in real-time.
- Connect to another endpoint across the internet.
- Bypass firewall and NAT.

At beginning, we find some platform as a services provide technology helps to communicate via video and audio in real-time. We did integrated to system and everything worked well. But soon we found the cost per minute used that services is not free, it cost $0.003 per minutes not much, but we do some calculate when the system reach up to 1000 user, and the cost is significant.

After that, the team develop found WebRTC, a free open-source is supported by Google, Mozilla and Opera. WebRTC is which "enable rich, high-quality RTC applications to be developed for the browser, mobile platforms, and IoT devices, and allow them all to communicate via a common set of protocols". We spent two weeks to study WebRTC and build a demo, it's worked.

WebRTC provide peer to peer communication mechanism, which mean when A want to communicate with B, then A must be know the B's address. To create a peer connection throws network between A and B, there are some limitations by firewall and NAT. STUN servers are used to get an external network address and to pass firewalls, so both A and B to determine their IP address as visible by the global

Internet. If both the peers are behind the same NAT, STUN settings are not needed since they are anyways findable each other. STUN is effective when the A and B are on different networks. In the MOLA system, because the time restricted so we not implement own STUN server yet, currently we are using a free STUN server from Google.

2. Reliable Recommendation

Like other item-related businesses, MOLA needs to implement a mechanism to suggest its items – courses – to users. Collective Intelligence and Collaborative Filtering [10] can be applied to solve the problem. However, the following problems occur when the development team implements Recommendation algorithms:

- There are many factors which have influence on giving score to courses, not just plain rating. A formula is needed for balancing those factors.
- For users who do not have many interacts with the system's courses (mostly new users), an alternative method for recommendation is needed.
- It is difficult for newly created courses to approach users as they do not have many interactions. The formula needs to have some factors that create opportunities for those new courses.
- Recommending methods needs a re-organized dataset to work on. Building the dataset consumes a noticeable amount of resources. Therefore, if the dataset can be stored for later access instead of rebuild every time there is request, performance will be improved.

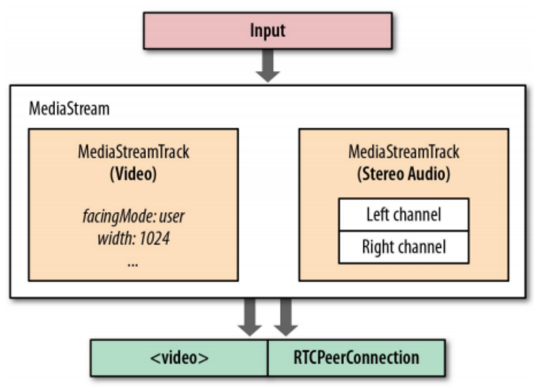
III. PLAN IMPLEMENT

1. Integrating WebRTC technology

MOLA system used WebRTC implemented in JavaScript, WebRTC provided 3 main APIs:

- MediaStream (or getUserMedia).
- RTCPeerConnection.
- RTCDataChannel.

MediaStream API enables get a stream taken from camera and microphone input has synchronized video and audio tracks. Output could be passed to an html video element <*video*> or an RTCPeerConnection.



1.1 getUserMedia method have 3 arguments:

- A Constraints object: such as aspect ratio, facing mode (front or back camera), frame rate, height and width video stream.
- A callback when success
- A callback when failure

When get stream success, an stream result object passed in callback function, it's contains field 'label' such as 'XUW7CLhsuHKKL2RWkW4yYKFJ5wONsgwIW9sP', and two MediaStremTracks; each has kind 'video' or 'audio' represented for either video stream or audio stream got from camera and microphone of input device.

1.2 RTCPeerConnection

RTCPeerConnection API enables to create a connection between peers and communicate audio and video by get local media, such as resolution and codec capabilities. This is the metadata used for the offer and answer mechanism. All public API can find at Mozilla docs [5].

The RTCPeerConnection instance pc represents a WebRTC connection between the local computer and a remote peer.

Once this local data has been establish, it must be exchanged via a signaling service with the remote peer (mention as Signaling server section).

1.3 RTCDataChannel:

RTCDataChannel allow exchange data peer to peer, in MOLA system the data is video, audio or arbitrary data.

2. Signaling server

The discovery and negotiation process of WebRTC peers is called signaling. Our proposed is to build a module called 'signaling server'. For two devices in different networks to find each other they need to use a central service called a signaling server. Using the signaling server two devices can discover each other and exchange metadata (describe as below). WebRTC does not specify signaling; different technologies such as WebSocket, Socket.io, XMPP or just simple done by using copy/paste.

In order for a WebRTC application to set up a 'call', its clients need to exchange information:

- Session control messages used to open or close communication.
- Media metadata such as codecs and codec settings, bandwidth and media types.
- Network data, such as a host's IP address and port.

The JavaScript Session Establishment Protocol [6] requires exchange between peers of offer and answer the media metadata mentioned above. Offers and answers are communicated in Session Description Protocol format [7], which look like this.

```
  v=0
  o=- 7537702454031608290 2 IN IP4 127.0.0.1
  s=-
  t=0 0
  a=group:BUNDLE audio video
  a=msid-semantic: WMS d9bb264d-9482-4a65-
9668-6f12e66f2944
  m=audio 9 UDP/TLS/RTP/SAVPF 111 103 9 102 0
8 105 13 110 113 126
```

```
  c=IN IP4 0.0.0.0
  a=rtcp:9 IN IP4 0.0.0.0
  a=ice-ufrag:niwZ
  a=ice-pwd:mOd9DTtBygiNr02004Usw8jx
  a=ice-options:renomination
  a=fingerprint:sha-256
19:4C:5F:24:C0:81:44:AD:E4:7E:4D:B1:19:1B:FE:C
0:01:27:80:D8:70:03:3C:E9:AC:71:59:62:41:EA:7C
:6D
  a=setup:actpass
  a=mid:audio
  a=extmap:1   urn:ietf:params:rtp-hdrext:ssrc-
audio-level
  a=sendrecv
  a=rtcp-mux
  ...
```

Best solution signaling service for MOLA system is built with Socket.io on NodeJS. The design of Socket.io makes it simple to build a service to exchange messages, and Socket.io is particularly suited to WebRTC signaling because of its built-in concept of namespace and rooms.

For example, HieuLN represents the local peer (caller) and QuangNN represents the remote peer (callee). We can show the role of signaling server as **diagram 1**.

In **diagram 1** the token is the metadata mention as above. It is an object contains SDP and some properties to recognize who are caller and callee. When step 6 finished. Signaling server's responsibility is done. WebRTC generated a connection to exchange data peer to peer.

3. Recommendation

3.1 The scoring formula

The development team consider the following factors in determining the score of a learner to a course:

- Percentage of completed lesson by the learner on the course (called A in the formula). This will be the main factor which makes differences among users on the same course.
- Teacher rating, registration per week and price factors (called B, C and D consecutively in the formula) are stable for each course. They are also factors for new user recommendation method.

Moreover, existed time of the course (called E in the formula) will be the denominator in the formula. This not only makes an advantage for newer courses, but also motivates older courses to change for adaptation.

The standard formula to calculate the score of a learner to a course is:

$$\frac{40\% \times A + 30\% \times B + 20\% \times C + 10\% \times D}{E}$$

The formula to calculate the score of a course to suggest for new users is:

$$\frac{50\% \times B + 30\% \times C + 10\% \times D}{E}$$

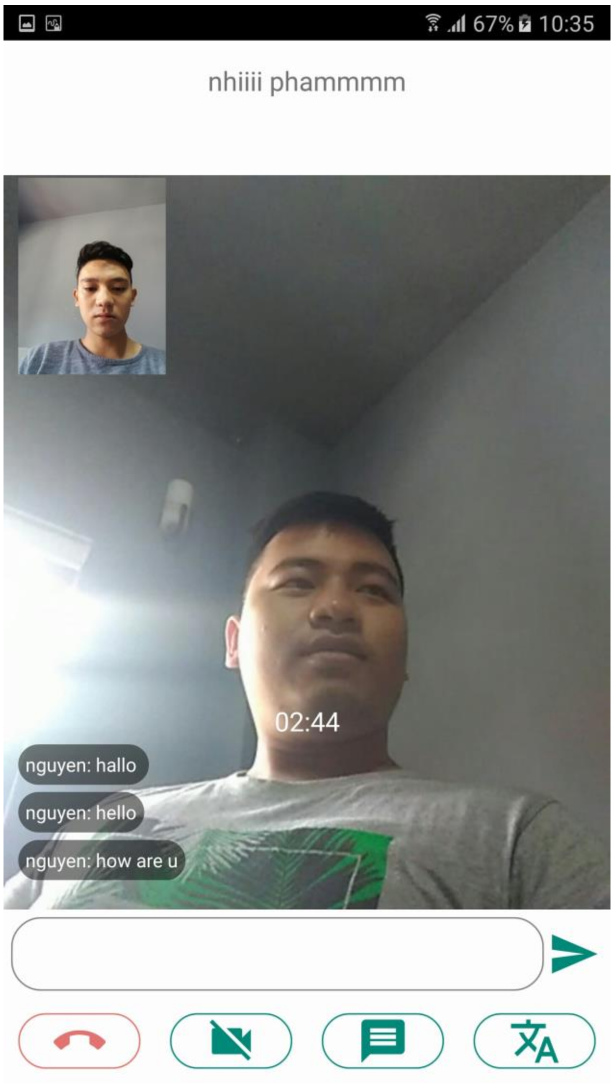
3.2 Dataset Storage

Although dataset for recommendation can be processed and calculated from database, storing it would have certain benefits. For large amount of data, accessing from storage would be faster than re-calculate for every request. Therefore, the development team decided to use temporary storage for that performance purpose. Redis [11] is selected because it is suitable for storing structured data and not difficult to be integrated.

The dataset will be re-calculated and stored once each day.

IV. EXPERIMENTAL RESULT AND CONCLUSION

We have executed some testing on 2 mobile devices



nguyen: hallo
nguyen: hello
nguyen: how are u

Conclusion:

After experiment many time, we conclude that WebRTC is suitable and worth for building real-time communicate applications. However, the quality of video and audio can be depend on network speed, so we not always have the highest quality and can be delay 500 - 2000 miliseconds.

Strengths:

- WebRTC has been widely used in several real-time services such as gaming, remote desktop applications, file transfer…
- Low cost for application, preinstall into browser, no need to install other plugins, add-ons…
- Peer to peer communication, no need intermediate server to transfer data.
- Encryption is mandatory of WebRTC, it's implementations use secure protocols such as DTLS[8] and SRTP[9].

Weakness:

- Still under development.
- Not fully implemented for Safari browser.

Acknowledgment

This paper owes massive thanks to Mr. Nguyen Huy Hung at FPT University. He was always support us whenever we ran into a trouble or had a question about our project. He consistently allowed this project to be our own work, but steered us in the right direction whenever he thought we needed it.

References

[1] Firewall:
https://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html
[2] Network address translation:
https://en.wikipedia.org/wiki/Network_address_translation
[3] Session Traversal Utilities for NAT:
https://en.wikipedia.org/wiki/STUN
[4] Traversal Using Relays around NAT:
https://en.wikipedia.org/wiki/Traversal_Using_Relays_around_NAT
[5] RTCPeerConnection API: ttps://developer.mozilla.org/en-US/docs/Web/API/RTCPeerConnection
[6] Javascript Session Establishment Protocol:
http://tools.ietf.org/html/draft-ietf-rtcweb-jsep-03#section-1.1
[7] Session Description Protocol:
https://en.wikipedia.org/wiki/Session_Description_Protocol
[8] Datagram Transport Layer Security:
https://en.wikipedia.org/wiki/Datagram_Transport_Layer_Security
[9] Secure Real-time Transport Protocol:
https://en.wikipedia.org/wiki/Secure_Real-time_Transport_Protocol
[10] Toby Segaran, *Programming Collective Intelligence* (2007)
[11] Redis:
https://redis.io/

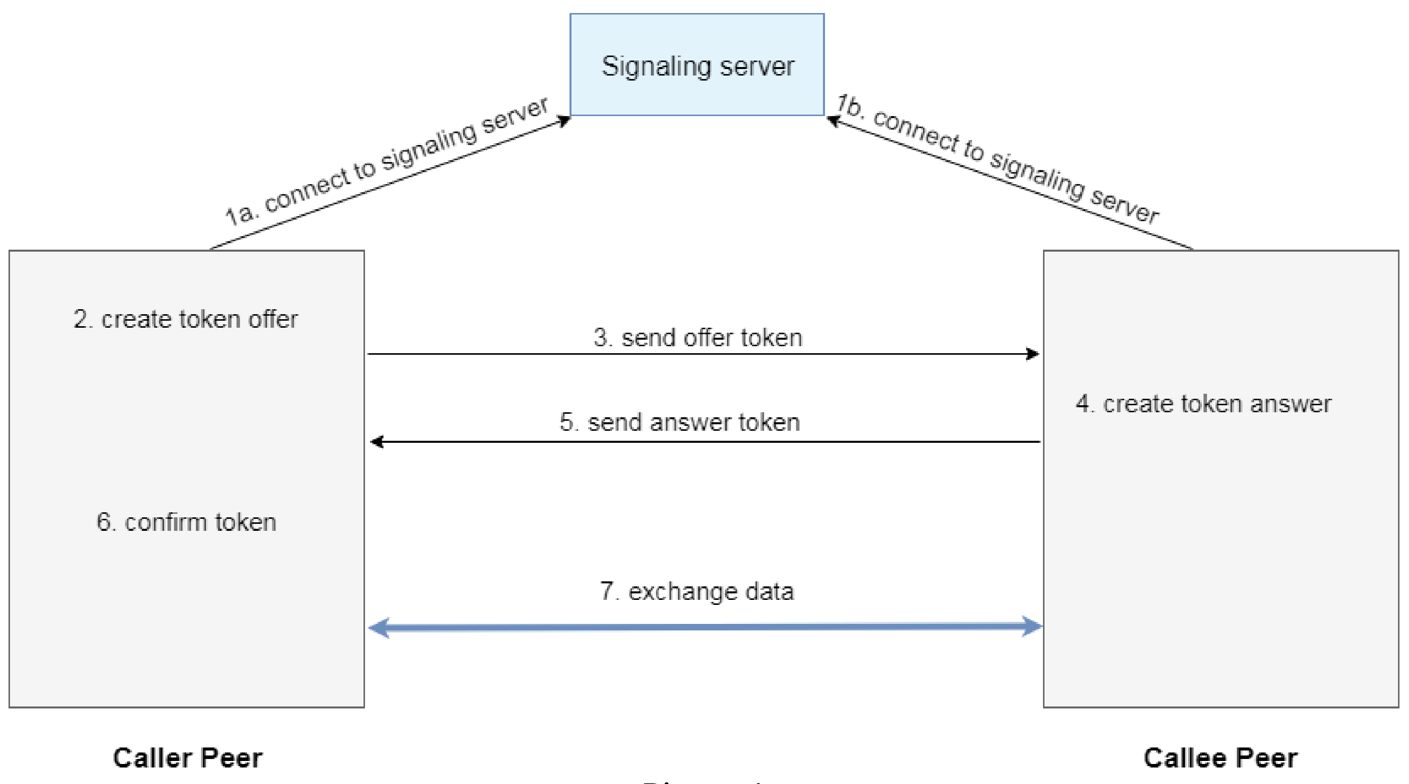

Signaling server
1a. connect to signaling server
1b. connect to signaling server
2. create token offer
3. send offer token
4. create token answer
5. send answer token
6. confirm token
7. exchange data
Caller Peer
Callee Peer

Diagram 1