Comunicações Hiperligadas: Novo Paradigma de Comunicação e Colaboração, potenciado pela Tecnologia WebRTC

Henrique Lopes Rocha
email1: henrique.rocha@ist.utl.pt

Instituto Superior Técnico

Abstract. In here put your abstract. Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem.

Keywords: keyworkd1, keyworkd2, keyworkd3

1 Introduction

The need to build a global comunications network in an era when almost nobody had access to it, caused that some protocols weren't suitable for a huge increase on the amount of publicly known users. Internet Protocol Version 4 (IPv4) limits the number of public addresses in such a way that today is scarse [5]. One way to overcome this problem was the development of a mechanism that groups multiple address into a single one, the machine that is assigned that address is then responsible to redirect messages to members of its group through their private addresses, each member of the private network is identified publicly by the same Internet Protocol (IP) address but different port, this technique is also known as Network Address Translation (NAT).

Initially NAT offered an alternative for address exhaustion and a minimal sensation of security. There are four types of NAT, "Full Cone NAT", "Restricted Cone NAT", "Port Restricted Cone NAT", "Symmetric NAT".

Full Cone NAT maps public each IP address and port to a private IP address and port, any external host can communicate with private hosts throught their mapped public address and port. This represents the least restrictive type of NAT and as we will later, the unique type of NAT that enables real time communications from point to point.

Restricted Cone NAT requires that a private client must first send a message to an external host before it can receive messages from the same host. With

this type of NAT, the private client can be contacted from any port of the same external host.

Port Restricted Cone NAT works in the same way as Restricted Cone NAT but only allows communications from the same external host's IP address and port, ignoring all messages from other applications within the same external host.

Symmetric NAT maps different ports for each connection, as we will see later, this represents a problem on real time communications.

Asymmetric NAT became a vulgar configuration on the web. As a direct result, problems started to appear, the amount of ports that IP disponibilizes is also small compared to our current needs, worse than that, NAT also difficult end-to-end communication, forcing most of applications that follows this model to be implemented ineffectively.

Applications based on multimedia and file sharing were one of the most strained by NAT. Those kind applications require real time communication in order to achieve the best performance. Session Traversal Utilities for NAT (STUN) and Traversal Using Relays around NAT (TURN) [6] servers are a possible solution to overpass NAT, although, none of those can establish direct connections on multiple level NATs.

STUN servers are quite simple, they receive requests from NATed clients, the source address of a request is the public address that NAT mapped to the client, STUN servers will reply the mapped public address to the client, so it knows its public IP address and port. Because Symmetric NAT changes IP port for each different connection, STUN servers will reply the IP address and port of their connection, which will be useless to clients connections, that's why Symmetric NAT represents a problem for real time communications.

On the other hand, TURN uses public servers to redirect traffic between private endpoints, it may use a Peer-to-peer (P2P) network relay to find the best peer but, after that, the behavior is much like client-server. Direct communication is only achieved by STUN when NAT is a type *full cone*. Interactive Connectivity Establishment (ICE) uses STUN when it's possible and TURN otherwise.

Most of client-server applications aren't affected by NAT when the servers are public, but they're inadequate for real time communication between two private endpoints. Clearly this type of communication requires a more expensive infrastructure and, in most cases, more network usage, leading to a worse quality of service. The requirements of video communication makes this kind of model unsuitable.

When connection is established, either in a direct or indirect way (via TURN servers), Web Real-Time Communication (WebRTC) cames to simplify how audio and video are transmited through web browsers.

WebRTC is an open source technology that defines a collection of standard protocols and JavaScript Application Programming Interface (API)s for web browser real time communications without installing any aditional application or plugin.

WebRTC defines three main APIs: GetUserMedia, PeerConnection and DataChannel.

- **GetUserMedia** allows from the browser to access to camera, microphone and device screen.
- **PeerConnection** aquires connection data and negotiates with peers.
- **DataChannel** allows to send whatever type of data to other peers.

WebRTC uses User Datagram Protocol (UDP) for transporting data, which provides lower latencies than Transmission Control Protocol (TCP), but is not reliable and packet order and integrity are not assured. Stream Control Transmission Protocol (SCTP) and Secure Real-time Transport Protocol (SRTP) are used for streaming data, providing a mechanism for congestion control and partial reliable delivery over UDP. All transferred audio, data and video must be encrypted with Datagram Transport Layer Security (DTLS) which provides the same security guarantess as Transport Layer Security (TLS).

Skype is an application that allows video, voice and instant messaging communication over proprietary protocols, its main strength is the amount of users that are using it nowadays. But compared to Skype, WebRTC applications don't need to be pre-installed.

Google Hangouts is video conference web application, in the past in order to use hangouts on a web browser a plugin was needed to be installed, nowadays hangouts is using WebRTC.

Jitsi Meet[1] is a WebRTC collaborative application that uses Jitsi Videobridge for high quality and scalable video conferences and supports shared document editing. Jitsi Videobridge is a server that enables multiparty video calls.

2 Signaling

Signaling is the most important proccess for applications to exchange connection information about peers and servers, their capabilities and metadata.

WebRTC doesn't implement signaling, different applications may require different protocols, there is no single answer that fits all problems. Amongst multiple options, signaling can be done by using Session Initiation Protocol (SIP), Extensible Messaging and Presence Protocol (XMPP), WebSockets, Socket.io or by implementing a custom protocol.

WebRTC uses Session Description Protocol (SDP) (rfc4566[2]) to define peer connection properties such as types of supported media, codecs, protocols used and network information. An SDP offer describes to other peers the expected type of communication and its details, such as used transport protocols, codecs, security and other.

One of signaling requisites is bi-directional communication over Hypertext Transfer Protocol (HTTP). HTTP works on a request followed by a server response, by other words, follows a unidirectional communication. Sometimes it's

[1] jitsi meet

[2] `http://tools.ietf.org/html/rfc4566`

required that some informations are obtained in real time, as we saw, some NAT's don't suport callbacks from servers, one technique to overcome this problem is polling.

Polling consists on sending periodic messages that the server responds imediately with empty content or fresh information. Because real time communications are unpredictable, if the time between periodic requests is short, most of time the server will return empty results wasting network bandwith and energy. On the other hand, if the time between periodic requests is large, newer messages may arrive later.

A technique called long polling consists on making the server hold the request until there is fresh information or expiring after some time, after the receival, the client makes another request. This results on a better network usage and a faster server response, but both simple polling and long polling requests are sent with HTTP headers, which adds data overhead, specialy for small messages.

The WebSocket protocol (rfc6455[3]) provides bidirectional communications trough a full-duplex socket channel. WebSocket handshake phase specifies an HTTP header in order to upgrade to websocket type of communication, the remainder messages are done without HTTP headers, which leads to much smaller messages and better network usage. WebSockets may not be avalailable on every web browser, frameworks like *socket.io*[4] and *SockJS*[5] uses HTTP when there is no support for WebSockets.

Bidirectional-streams Over Synchronous HTTP (BOSH) is a technique based on long polling, that uses two socket connections and allow sending client messages to server while a previous request is holded.

SIP (rfc3261[6]) is protocol used for negotiation, creation, modification and finalization of communication sessions between users. SIP follows a client/server architecture with HTTP like messages and it can be used as signaling protocol. The advantage of SIP is the ability to make video and voice calls applications over IP networks.

The work group SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE) proposed the creation of SIP extensions, namely presence information (rfc5263[7]) and instant messaging (rfc3428[8]).

SIP is used in Voice Over IP (VoIP) applications due to its compatibility with Public Switched Telephone Network (PSTN). Service providers are disponibilizing access to their SIP infrastrucures through WebSockets. Frameworks like *jsSIP*[9], *QoffeeSIP*[10] and *sipML5*[11] are used on client side to parse and encode SIP messages, making SIP accessible to web based applications.

[3] `http://tools.ietf.org/html/rfc6455`
[4] `http://socket.io/`
[5] `http://github.com/sockjs`
[6] `http://tools.ietf.org/html/rfc3261`
[7] `http://tools.ietf.org/html/rfc5263`
[8] `http://tools.ietf.org/html/rfc3428`
[9] `http://jssip.net/`
[10] `http://qoffeesip.quobis.com/`
[11] `http://sipml5.org/`

SIP with WebSockets can be used as a signaling method for WebRTC applications, it allows web browsers to have audio, video and Short Message Service (SMS) capabilities like mobile phones. For instance, it's possible to interoperate web communications with SIP networks, mobile and fixed phones.

[XMPP sig and jingle]

XMPP was initially developed for instant messaging (Jabber[12]). It is nowadays an open technology for real-time communications.

XMPP messages are eXtensible Markup Language (XML) based, which are attractive for applications that need structured messages and rich hypermedia. XMPP advantage is the addition of extensions, for example XEP-0096[13], which adds file transfer capabilities between two entities and XEP-0045[14] which enables multi-user chat.

XMPP's bi-directional communication is achieved through BOSH (XEP-0206[15]), which basically consists on long polling. This kind of communication is also possible through WebSockets (rfc7395[16]).

Amongst multiple XMPP servers softwares are: ejabberd[17], Metronome[18], Openfire[19] and Prosody[20]. Ejabberd is the one that implements more Request For Comments (RFC) specifications and XMPP Extensions (XEP)s[21].

3 Hypermedia

Since the early days of video technology, one of the problems that raised with it consisted of how to add more information onto it without generating multiple versions. Some implementations like [4] added hypermedia information to empty space present on Moving Picture Experts Group (MPEG) frames in order to provide interactive television, the MPEG coder and decoder were changed in order to handle hypermedia content.

The need to translate movies, raised the problem whether it is apropriate to change the original video or audio. For example subtitles should be an entity independent from the video, in order to be personalized or replaced easily.

Amongst multiple formats for subtitles, Synchronized Accessible Media Interchange (SAMI), and SubRip Text (SRT) are used by video players that support them. Although those formats have styling available, they are quite limited to text.

[12] http://jabber.org/
[13] http://xmpp.org/extensions/xep-0096.html
[14] http://xmpp.org/extensions/xep-0045.html
[15] http://xmpp.org/extensions/xep-0206.html
[16] http://tools.ietf.org/html/rfc7395
[17] http://jabberd.im/
[18] http://lightwitch.org/metronome
[19] http://igniterealtime.org/projects/openfire/
[20] http://prosody.im/
[21] http://en.wikipedia.org/wiki/Comparison_of_XMPP_server_software

Hypervideo is a kind of video that contains links to any kind of hypermedia, including links to skip part of it. An example of hypermedia application could be a search engine over hypermedia content, like subtitles, in order to jump to a specific point in time. HyperCafe [8] was an experimental project to expose hypervideo concepts that consisted on an interactive film by switching between conversations inside a cafe.

Detail-on-demand is a subset o hypervideo that allow us to obtain aditional information about something that apears along the video, like obtaining information about a painting that appears in a particular segment. Hyper-Hitchcock[9] is an editor and player of detail-on-demand video.

In order to navigate through a dynamic video, one must be aware of time synchronization and the multiple time flows, it's important that all time, causality and behavior rules are well defined.

HyVAL[11] is an XML based language that was proposed for modeling composition, synchronization and interaction of hypermedia. HyVAL defines defines video structure, internal video and external media objects.

HyVAL video structure objects defines a structure derived from traditional video, which divides video into segments, scenes, shots and frames hierarchically, this approach is quite limitative if we want to apply hypervideo concepts to videos that don't follow this structure. External media objects are linked by primary video, those objects can represent other videos, images, text, animation and sound.

Synchronized Multimedia Integration Language (SMIL)[1] was introduced to describe temporal behavior of multimedia content, in particular, it could be used to overlay subtitles on films. With SMIL it's possible to synchronize multiple sections of video, either in parallel or in sequence, reproduce a different audio track, overlay user interface elements with hyperlinks, amongst multiple other functionalities.

SMIL is XML based and defines twelve modules: *Animation*, *Content Control*, *Layout*, *Linking*, *Media Objects*, *SmilText*, *Metainformation*, *Structure*, *Timing*, *Time Manipulations*, *State* and *Transitions*.

- **Animation** module contains elements and attributes that define a time based mechanism for composing the effects of animations. For example, this module can changes on XML or Cascading Style Sheets (CSS) attributes like colour and dimensions.
- **Content Control** module contains elements and attributes that provide optimized alternatives for content delivery. For example, it could be used to change audio language in function of user's nationality, for videos with multiple audio channels.
- **Layout** module contains elements and attributes for colouring and positioning media content, another layout mechanisms are also possible, such as CSS.
- **Linking** module contains elements and attributes for navigatonal hyperlinking. Navigation can be triggered by events or user interaction.

- **Media Object** module contains elements and attributes for referencing rendering behaviour of external multimedia or control objects.
- **SmilText** module contains elements and attributes that defines and controls timed text. For example this module could be used to create labels and captions.
- **Metainformation** module contains elements and attributes that allows describing the SMIL document. For example, this module could be used to define a movie details such as category, director, writters and cast.
- **Structure** module defines the basic elements and attributes for structuring SMIL content. This module defines a *head* element that contains non temporal behaviour information defined by *Metainformation*, *Layout* and *Content Control* modules. This module also defines defines the *body* element, where all temporal related module information is contained.
- **Timing** module is the most important module on SMIL specification, because of its complexity, it is divided into seventeen submodules for coordination and synchroniztion of media over time. The three main elements are *seq*, *excl* and *par*, respectively, they play child elements in sequence, one at a time and all at the same time.
- **Time Manipulations** module adds time behaviour attributes to SMIL elements, such as speed, rate or time.
- **State** module defines attributes that defines the state of SMIL elements, such as element visibility, current element time, amount of repeated loops, playing state and many others.
- **Transitions** module defines attributes and elements that defines transitions across multiple SMIL elements according to *Timing* module.

The Document Object Model (DOM) is a standard API that allows easy management of documents that are organized in a tree structure, by providing Create, Read, Update and Delete (CRUD) operations over its elements and their attributes. DOM makes it easy to interoperate between imperative and declarative programming languages.

Like DOM, SMIL DOM is an API for SMIL documents. Allowing CRUD operations over SMIL documents is an important feature for extending SMIL capabilities, for example for creating non-linear animations and triggering external events like *javascript* functions.

SMIL's modules are used to synchronize and animate eXtensible Hypertext Markup Language (XHTML) and Scalable Vector Graphics (SVG) elements.

In order to create a multimedia rich hypercall, SMIL fits our goals, but it lacks on browser compatibility. Ambulant [2] was one of the SMIL players that were developed for browsers, although this player implements most of SMIL 3.0 [10] specifications, it needs to be installed on browsers as a plugin.

HyperText Markup Language (HTML) is a markup language based on XML that is used for creating web pages. HTML alone is a very poor language when we are focused on visual appealing and interactive web pages. Languages like CSS and Javascript are tipically combined to HTML for improving the interaction and appearance of a web page.

CSS idea is to separate the structure of an XML document from its appearance. CSS defines styles for XML tags based on their name, class, identifier or position, besides static styling it also supports animations and transitions leading to a more dynamic content.

Javascript is an imperative object oriented language based on ECMAScript. It is used mainly on client-side and executed by a web browser. Javascript has it own implementation of DOM and one of its advantages is the ability to download and execute code on the fly without the need of pre-installed plugins.

Javascript has compatibility issues amongst different web browsers, leading to different behaviours. To solve that problem, there are libraries written in javascript, namely *jQuery*, that implements the same funcionality for multiple browsers, solving most of incompatibility issues.

SmillingWeb [3] attempts to implement a cross platform multimedia player designed for SMIL 3.0 presentations with *JavaScript* and *jQuery* which, unlike [2], doesn't need to be installed and shouldn't have incompatibility issues.

SmillingWeb already takes advantage of HTML5 and CSS3, they take into account unsuportted web browsers though Modernizr[22], a simple JavaScript library that may require plugins if new features aren't supported.

But SmillingWeb just implements a subset of SMIL 3.0 and their scheduler engine loads the SMIL file only once, which could raise problems when leading with SMIL changes due to real time communications.

Another problem with SmillingWeb is pre-loading and playing elements at the correct interval of time, which is not always possible due to low latency netwoks leading to experience pauses during playback.

With the emergence of HTML5, tags like *video*, *audio* and *track* allow us to play video with multiple codecs, audio and subtitles in Web Video Text Tracks (WebVTT) format. Another important tag is *canvas* that allows to draw graphics with javascript on a rectangle within a web page.

SVG is an XML based format that incorporates the animation module of SMIL. Currently SVG allows to add movement and animate attributes of elements. When embedded on HTML5, it allows dynamic changes to inner content in real time through DOM API, besides that, it also allows to call javascript functions on events such as animation end, mouse over and mouse click.

Video and audio functionalities are already possible with HTML5, like SVG it is also possible to bind javascript functions for different kinds of events over video and audio elements.

Back in 1995, *flash*[23] was developed for web-based animations, introducing video support in 2002, flash started to grow after that. Concurrency players, at that time, were focused on playing video and audio, flash had vector graphics and they were focused on streaming *on-demand* video across multiple platforms. VP6 was their choice on video codecs, optimizing for half of video size for the same quality and providing adjusted video quality based on internet connection latency.

[22] modernizr url
[23] `flashsite`

Adobe Flash was the most widely used applications for reproducing live broadcast and recorded video [?], it supports progressive video download using HTTP and streaming using Real Time Messaging Protocol (RTMP).

RTMP is a TCP based protocol used to streaming audio, video and data between Flash Media Server (FMS) and flash player. A bidirectional connection is established between the two to in order allow real time coomunications. A flash player can stream a webcam video to FMS according to RTMP and another flash player can request a video stream to FMS that can be pre-recorded stream, live stream or data. Multiple FMS servers can be chained together in order to increase capacity and handle more streams simultaneosly.

FMS can stream video and audio to one or more subscribers by sending a separate copy for each subscriber. With Real-Time Media Flow Protocol (RTMFP) is possbile to stream video between flash players, allowing a publisher breaking a stream into pieces that can be cooperatively distributed in a P2P mesh, RTMFP uses UDP to speed packet delivery, although is not reliable, it is well suited for video streaming, like WebRTC flash players also need to apply techninques like STUN and TURN for NAT traversal.

Although HTML5, javascript, CSS and WebRTC are implementing some functionalities of flash, it doesn't mean that flash will be replaced, instead of that both technologies can be used to develop rich internet applications. It is also important to note that HTML5 is more compatible with mobile devices than adobe flash.

Like Flash, Microsoft Silverlight is a cross browser plugin and platform that is used to develop rich internet applications. It supports vector graphics, animation and video. Compared to flash, which uses ActionScript, Silverlight applications can use languages like C#, VisualBasic and eXtensible Application Markup Language (XAML). Silverlight uses a technique called Smooth Streaming from IIS Media Service that consists on delivering video in real time with adjusted quality in function of bandwidth changing and Central Processing Unit (CPU) usage.

By using techonologies that relies only on web standards, like CSS, HTML5, Javascript and SVG, it's possible to raise communications to a new level. For example, with APIs like WebGL[24], it is now possible to manipulate a three dimensional environment in the context of a hypercall. Another example would be a collaborative spreadsheet using WebRTC. With this, hypercalls are not limited to only audio, image, text and video, but also interaction with complex graphical user interfaces that changes over time.

In this project our goal is to enrich hypercalls with no limits, every user should be free to choose how it wants to be contacted and it wants to share its contents.

In order to give users a personalized communication channel, each user must have a personal web page where its available plugins could be downloaded from other peers, after that they can talk in the same language whatever it is.

[24] `http://khronos.org/webgl/`

4 Applications

Real time collaboration applications have become a huge help on team tasks, providing a great boost on business, research and investigation velocity. Technologies like this are appearing along this days, but they couln't be possible years ago because technology was limited or unavailable. Although todays technology is limited on some aspects, we are doing progress in order to improve the web ecosystem, by creating standards and migrating to newer technologies.

Our first concern on real time collaboration applications, besides the communication itself, is the data storage and representation. Because most browsers are recommended to limit local storage to at least five megabytes per origin, storing multimedia content is not a viable solution.

If, for instance, one whants to rewind a real time video, recordings will be needed from who is streaming the video.

Real-time Transport Protocol (RTP)[25] is used for streaming audio and video over IP, the multimedia content is transported on the payload of RTP messages, RTP contains headers for payload indentification. RTP is independent from its payload type, allowing to transport any kind of encoded multimedia. A sequence number is used for sorting received packets.

RTP allows to change its requirements and add extensions to it with profiles, one of the most used is the RTP profile for audio and video [26] which lists the payload encodings and compression algorithms. This profile also assigns a name to each encoding which may be used other protocols like SDP.

[DTLS and SRTP]

RTP recorders are independent of payload encoding, they don't decode RTP packets, they record packets instead, allowing to record all video and audio formats.

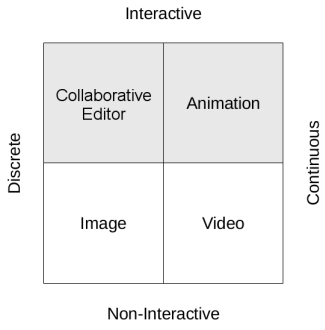

Fig. 1: Media Types

[25] rfc3550
[26] rfc3551

Media Types can be distinguished by two criteria, the first one describes a media as discrete or continuos, the second one describes it as interactive or non-interactive. A discrete media is caracterized by its type not depending from time, as continuous media depending from it. Interactive media is caracterized by its state being changed by external events such as user interactions.

For example, an image is non-interctive and discrete, for instance, a video is continuos and non-interactive. A simple colaborative editor with just text is interactive and discrete. An animation that changes in function of user behaviour is interactive and continuous.

Streaming protocols like RTP were designed for continuous and non-interactive media types, such as audio and video. Discrete and non-interactive media don't need to streamed through RTP because they don't change with. For example if an image appears in a specific time interval, just the HTML or javascript that will reference the image must be streamed, the image itself is then transfered through HTTP.

In order to play any kind of stream, a player for interactive stream is needed for downloading an environment, decoding the RTP payload to determine the state and display it to the user. Streaming interactive media like the combination of HTML, CSS and JavaScript requires more than interpreting the code, a streamed user interface may contain an internal state that is not shown on code.

Everytime an event is proccessed on one of the endpoints, both sender and receivers state must stay synchronized, otherwise events may behave differently.

To achieve synchronization of interactive data most packets have three types: *State*, *Delta-State* and *Event*. State packet defines environment's complete state. Delta-State packets transports just the piece of state that changed. Event packets informs that an event ocurred over the interactive media.

An RTP recorder can have two operation modes, recording or playback. Traditional RTP players can do random access, in contrast, interactive RTP players must restore the environment and context at a given time. The environment is the initial state, so we can call it a non-interactive descrete media and handle it over HTTP. After the receiver has received the environment, it should calculate the state at the given time.

If the RTP recorder controls the correct data to send to the receivers, it cannot be a simple RTP recorder as it must compute the state or delta-state to send. Therefore, if the receiver receives all recorded packets, it can calculate the current state from a nearest complete state. Streaming too much complete states, results on more precise random accesses but the trade-off is the higher bandwidth usage and used storage space on the recording server. On the other hand, if there are fewer complete states recorded followed by delta-states, the recorded stream will occupy less storage space, but random accesses will be less granular.

By recording and streaming the interactive media's complete state periodically, it is possible to restore the media state even if messages are lost.

In order to synchronize an interactive application state amongst participants, the needed objects to synchronize must be serializable and sent to other participants.

[Model View Controller (Suited for interactive RTP)]

With such an interactive RTP recorder it is then possible to record, play, fast forward, fast rewind, stop and jump to random positions.

[7] proposed an RTP profile for real-time transmission of interactive media. This new profile reuses much of video and audio profile implementation, integrating the interactive component. [**?**] explained how to record interactive video with this new profile.

Multiparty video calls can be achieved on WebRTC by streaming video from each participant to all the other participants. Although this works, the bigger a conference room is, the bigger is the bandwidth used to stream video to all participants within the conference room.

Jitsi Video Bridge [27] receives one stream from every participant on a conference, either from a jitsi client or a WebRTC application, and redirects it to all the other conference participants, reducing the amount of data that each peer sends. Although all the participants need to download all the streams from Jitsi Video Bridge Server, typically download rates are much bigger than upload rates, making this solution more feasible.

Jitsi Video Bridge uses XMPP as a signaling protocol and its colibri extension (XEP-0340) to reserve channels for video transmission. Although this choice for signalling protocol, Jitsi Video Bridge also supports SIP and **Jingle!** (**Jingle!**).

[Identity]

> There is strong growth in the deployment of devices that integrate regular Web technologies such as HTML, CSS, and SVG, coupled with various device APIs.

4.1 Context

4.2 Problem Statement / Solution Statement

4.3 Thesis Contributions

4.4 Article Structure

4.5 Methodology

4.6 Planned Schedule

5 Conclusions

5.1 Summary

6 Conclusions

References

1. Bulterman, D.C.: Smil 2.0 - part 1: Overview, concepts, and structure (2001)

[27] jitsi VB

2. Bulterman, D.C., Jansen, A., Cesar, P.: Video on the web: Experiences from smil and from the ambulant annotator (2007)
3. Gaggi, O., Danese, L.: A smil player for any web browser (2011)
4. Gradvohl, A.L.S., Iano, Y.: Combinando tv interativa e hipervideo (2007)
5. IEEE-USA: Next generation internet: Ipv4 address exhaustion, mitigation strategies and implications for the u.s. (2009)
6. Lin, Y.D., Tseng, C.C., Ho, C.Y., Wu, Y.H.: How nat-compatible are voip applications? (2010)
7. Mauve, M., Hilt, V., KuhmÃ¼nch, C., Effelsberg, W.: A general framework and communication protocol for the real-time transmission of interactive media (1997)
8. Sawhney, N., Balcom, D., Smith, I.: Hypercafe: Narrative and aesthetic properties of hypervideo (1997)
9. Shipman, F., Girgensohn, A., Wilcox, L.: Creating navigable multi-level video summaries (2003)
10. W3C: Smil 3.0 (2008), http://www.w3.org/TR/SMIL3/
11. Zhou, T.T., Jin, J.S.: A structured document model for authoring video-based hypermedia (2005)