Computing
DOI 10.1007/s00607-014-0429-2

WebRTC-based video conferencing service for telehealth

Julian Jang-Jaccard · Surya Nepal · Branko Celler · Bo Yan

Received: 28 January 2014 / Accepted: 9 September 2014

Abstract Existing video conferencing systems that are often used in telehealth services have been criticized for a number of reasons: (a) they are often too expensive to purchase and maintain, (b) they use proprietary technologies that are incompatible to each other, and (c) they require fairly skilled IT personnel to maintain the system. There is a need for less expensive, compatible, and easy-to-use video conferencing system. The web real-time communication (WebRTC) promises to deliver such a solution by enabling web browsers with real-time communications capabilities via simple JavaScript APIs. Utilizing WebRTC, users can conduct video/audio calls and data sharing through web browsers without having to purchase or download extra software. Though the promise and prospective of WebRTC have been agreed on, there have not been many cases of real life applications (in particular in telehealth) that utilizes the WebRTC. In this paper, we present our practical experience in the design and implementation of a video conferencing system for telehealth based on WebRTC. Our video conferencing system is a part of a large tele-home monitoring project that is being carried out at six locations in five different states in Australia. One of the aims of the project is to evaluate whether high-bandwidth enabled telehealth services, delivered through tele-home monitoring, can be cost effective, and improve healthcare outcomes and access to care. This paper however focuses on WebRTC-based video conferencing system which allows online meetings between remotely located care coordinators and patients at their home. We discuss the underlying issues, detailed design and implementation, and current limitations of using WebRTC in a real life application.

Keywords Telehealth · Video conferencing · WebRTC · P2P · Real time communications · Multimedia

J. Jang-Jaccard (✉) · S. Nepal · B. Celler · B. Yan
CSIRO Computational Informatics (CCI), Marsfield, Australia
e-mail: julian.jang-jaccard@csiro.au

Published online: 25 September 2014

Mathematics Subject Classification 68U35 Information systems (hypertext navigation, interfaces, decision support, etc.)

1 Introduction

Telehealth promises to revolutionize the way health services are delivered to patients at home [6]. Telehealth services offer many benefits by increasing the availability of health services and improving the health outcomes for the patients, who are disadvantaged from getting routine medical services due to the geographical locations they live in or having the limited mobility. For example, the patients can be conveniently linked to their GPs and specialists without the need for travel. Telehealth can provide timely access to medical services and specialists which could result in improving the ability to identity developing conditions at an early stage. Telehealth can also provide means for patients to monitor and manage their health on everyday basis without seeing their GPs and specialists.

A number of projects have been conducted to study the feasibility of implementing tealehealth services. Example projects include chronic disease management [1], mental health care [2], ophthalmology [3], diabetes [4,7] and allied health [5]. Ageing populations and the growing cost of healthcare are some of the main reasons governments around the world are attracted to telehealth services as a way to reduce overall health cost [5,8]. Furthermore, advancements in information and communication technologies, such as ubiquitous broadband networks, mobile technologies, cloud computing, and social networks, have also played a key role in developing further interests from governments in telehealth. As a result, a number of government-funded telehealth services research projects have been established. CSIRO Health Services research program was recently awarded a project under the Commonwealth Telehealth Pilots Program to demonstrate how telehealth services for chronic disease management in the community can be deployed nationally to reduce the rate of hospitalization and improve healthcare outcomes of chronically ill patients living at home [9].

Traditionally, many telehealth services have been delivered using the store-and-forward mode [37]. In this mode, digital images, video, audio, and clinical data are captured and stored on the client computer or mobile device offline. They are then transmitted securely to a clinic at another location at a later time so that the data is studied by relevant specialists. The results or opinions of the specialists are then transmitted back sometimes later. Though this method worked well for many decades, there were requirements for real-time telehealth services which can allow instantaneous interactions between stakeholders. Video conferencing is one of the most common forms of real-time technology adopted in telehealth services. In many cases, peripheral devices were attached to computers or the video-conferencing equipment in order to aid in an interactive examination [11]. With the availability of better and cheaper telecommunication, direct two-way audio and video streaming between medical centers through computers is now possible at lower costs. It is predicted that there will be explosive growth in the use of video conferencing in telehealth services as a fundamental tool to deliver more effective care [10]. It is anticipated that nearly half of information workers will have some type of personal video solution in 2016 [10].

Though the usefulness of the video conferencing system specially made for telehealth services gained positive responses, a number of shortcomings of using these solutions have been raised over the years. There has been a concern about purchasing price, which are often too expensive for many budget strapped small scale hospitals/clinics. Most of the solutions are developed using proprietary technologies in ad-hoc manner that were not compatible to use with other similar solutions. For example, there are hundreds of different ways to process audio and video with no interoperability among them. Often the solutions required a comparatively high level of IT skills to download the software, to understand the set up and configuration instructions, and to provide ongoing maintenance (e.g., upgrades). These concerns are also equally valid with users in our project who are all senior citizens of Australia over 55 years old who suffer from one or multiple chronic illnesses and do not necessarily have the required IT skill.

To satisfy the need of simpler, inexpensive and interoperable approaches towards developing a video conferencing system for the project, we adapted the efforts from emerging WebRTC (web real-time communication) initiative [13]. The WebRTC refers to the work that is being carried out by standardization bodies such as W3C [30] and IETF [31] to support real time communications. The work aims to offer web developers native browser tools to insert real-time media stream exchange services easily into the web pages [13]. This allows developers to build applications more easily for audio and video calls, live video sharing and screen sharing, instant messaging as well as peer-to-peer data sharing. The in-browser support of media content means that end users no longer require to download, install and manually configure an application or to use some proprietary plug-ins in the browser. All multimedia functions are natively integrated by the browsers.

Though the work of WebRTC is promising, it is still difficult to find good examples of real life applications in telehealth services. Rhinow et al. [14] analyze the feasibility of implementing live video streaming protocols into web applications using WebRTC. We found that details of design and implementation are too abstract and it is difficult to see exactly which WebRTC components were employed in their approach. The work in [15–18] only describe the potential of WebRTC and the possibility of using it for a telehealth context without the mention of how it can be actually done. Rodriguez et al. [19] propose an advanced video conferencing system using WebRTC. Their proposal only focuses on the design issue to integrate the WebRTC for multi-party scenarios, but do not mention any implementation details. Fernandez et al. [20] propose Kurneto, a media server for mobile real time multimedia communications using WebRTC. Their focus is on the media server which can be only a part of component in a video conferencing system. A model focused proposal towards multimedia conference based on WebRTC is described by Elleuch in [21].

Set apart from the previous works, we have designed and developed a standard-based video conferencing system using WebRTC for our telehealth research project. This paper describes our practical experience of the development and provides unique insights into how easy or difficult it is to use currently available WebRTC technology into a real-life application in telehealth services. The design part describes the details of WebRTC concepts that were used to meet the need of our video conferencing system. The implementation part describes how design issues were realized with coding

examples. Then, we discuss limitations of our current system as well as WebRTC and provide future directions on how to overcome such limitations.

The rest of the article is organized as follows. We explain the motivation and background information on our tele-home monitoring project in Sect. 2. Related work on video conferencing technologies and basic introduction to WebRTC are presented in Sect. 3. We then describe the architecture and design of our video conferencing system in Sect. 4. We have implemented a prototype system to realize the underlying architecture and model, which we describe in Sect. 5. Finally, we present a brief conclusion and future work in Sect. 6.

2 Motivation

CSIRO through its health services theme has been carrying out a comprehensive analysis of the international literature of the factors impacting on the health care system associated with the ageing of the population and increasing burden of the chronic diseases [22–25]. It is found that a strong primary health care system is critical to the future success and sustainability of the healthcare system. The importance of primary care has emerged as a recurrent theme in government reports and strategy documents [22–24].

Chronic and aged care accounted for over 70–78 % of Australia's $140.2bn expenditure on healthcare during 2011–12, and is projected to dramatically increase into the future [25]. Australia's hospital-centric public health system is unnecessarily burdened by the management of the chronic disease, which should preferably occur in home and community settings. Telehealth services delivered through home monitoring have been demonstrated to deliver cost-effective, timely and improved access to quality care [26]. They also reduce social dislocation and enhance the quality of life and sustainability of these communities by allowing chronically ill and aged members to stay in their homes and communities longer. These statements are also supported by recently released findings of the whole system demonstrators (WSD) by the UK department of health [27]. Similar outcomes were observed in the telehealth programs administered by Veterans Health Administration (VHA) in the US [28].

However, experience in Australia with the deployment of telehealth services is extremely limited, with most deployment of small scale and lack of detailed analysis of the following key success factors: health care outcomes, health economic benefits, impact of clinical workforce availability and deployment, human factors, workplace culture and organizational change management and business processes. The development of robust business case and business models for large scale commercial deployment of telehealth services based on solid socio-economic evidence is therefore essential if these services are to be scaled up nationally to have an impact on escalating costs of health services delivery and increasing deficit in clinical workforce.

With the aim of addressing the above raised issues, a project "Tele-home monitoring of chronic disease for aged care" is being carried out by CSIRO since January 2013 for the duration of 18 months under the Commonwealth Teleheath Pilot program administered by the Department of Health and Ageing (DoHA). The project is the first multi-state, multi-site telehealth trial performed in Australia. The telehealth trial carried out by the project is conducted at 6 sites across five states of Australia.

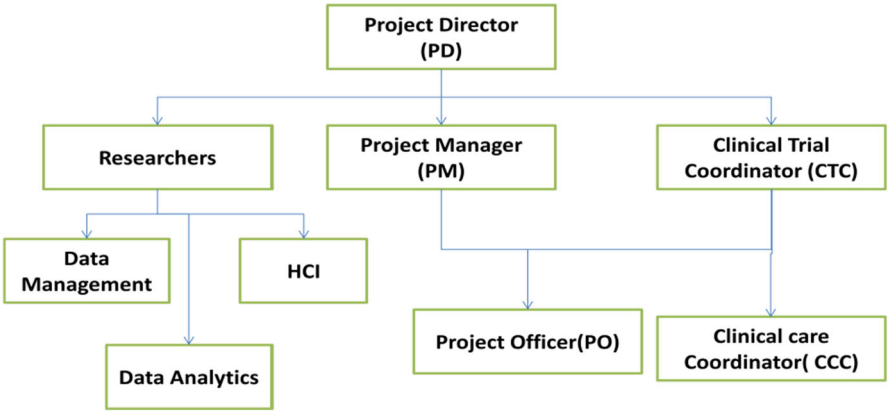

Fig. 1 Project role allocation

The participating 6 sites include: Nepean Blue Mountains Medicare Local (New South Wales), Anglican Retirement Villages (New South Wales), ACT Health (Australian Capital Territory), Northern District Health Service (Tasmania), Townville Mackay Medicare Local and Health District (Queensland), and Ballarat Health Services and Grampians Health Alliance (Victoria). Each site manages 25 test patients and 50 control patients (i.e., in total of 150 test patient and 300 control patients) in both public and private health settings.

There are people involved with different roles and the responsibilities during the course of the project which is shown in Fig. 1.

The project director (PD) oversights the overall project team consisting of a number of researchers, a project manager (PM) and a clinical trial coordinator (CTC). Each trial site has a project officer (PO) and one or more clinical care coordinators (CCC).The PO and CTC works with PM and CCCs to run the project on each site. Researchers work with PD to conduct different research activities. Three groups of researchers are involved in the project. The data management group deals with collection, integration, and generation of data. The Human–Computer Interaction (HCI) group deals with workforce availability and deployment, human factors, workplace culture and organizational change management and business processes. The data analytics group, mostly consist of mathematicians, deals with developing advanced modeling and data analytics to risk stratify patients on a daily basis to automatically identify exacerbations of their chronic conditions.

A variety of patient data from different sources is collected at the different phases of the trial. These include:

Hospital records The project collects hospital records of all trial patients (both test and control) twice. In the beginning of the trial, we collect the last 3 years of the hospital records of all patients. Then again, we collect the hospital data for the duration of the trial at the end of the trial. These two sets of data is finally analyzed and compared at the last stage of the trial to gather the overall benefits of telehealth services in managing patients in their own homes and communities.

Fig. 2 A TeleMedCare (TMC) device

PBS/MBS We use Medicare Benefits Schedule (MBS) and Pharmaceutical Benefits Scheme (PBS) records from the Department of Human Services (DHS) (formerly known as Medicare) to capture information about patients' visits to their local GPs or the use of medications.

TeleMedCare (TMC)[1] TMC, a home monitoring device, is installed at the home of 150 test patients. Patients use TMC to monitor the vital signs as per schedule to gather clinical data of their health progress. The vital signs that are captured by TMC include weight, blood pressure (auscultatory and oscillometric), temperature, single lead ECG, spirometry, glucometry, etc. The measured vital signs are stored in the local computer and are sent to the TMC server at regular intervals. Figure 2 shows the TMC device the patient use for the trial.

OpenClinica (OC) We use OpenClinica,[2] which is open source clinical trial software, to model and capture various types of questionnaires at different stages of the project. The researchers at CSIRO administer a number of specific clinical and wellness questionnaires during the course of the trial. The questionnaires are designed to develop an understanding of the model of care provided by the trial. Questionnaires include demographic information, use of health services, behavior, physical activities, anxiety and depression, living with and managing medical conditions, quality of life, social isolation, and medical adherence. These questionnaires and their answers from the patients are recorded in the OpenClinica on a regular basis.

CSIRO Portal One of the important tasks in our project is linking different data coming from various data sources. Our CSIRO portal, developed in-house (Fig. 3), consolidates the data from different data sources and offers data services to other people (e.g., our Statistician and HCI colleagues) for further modeling and analysis. Further to collect the data from the existing sources, we also collect project specific data to conduct impact of clinical workforce availability and deployment, human factors, workplace

[1] www.telemedicare.com.au.

[2] www.openclinica.com.

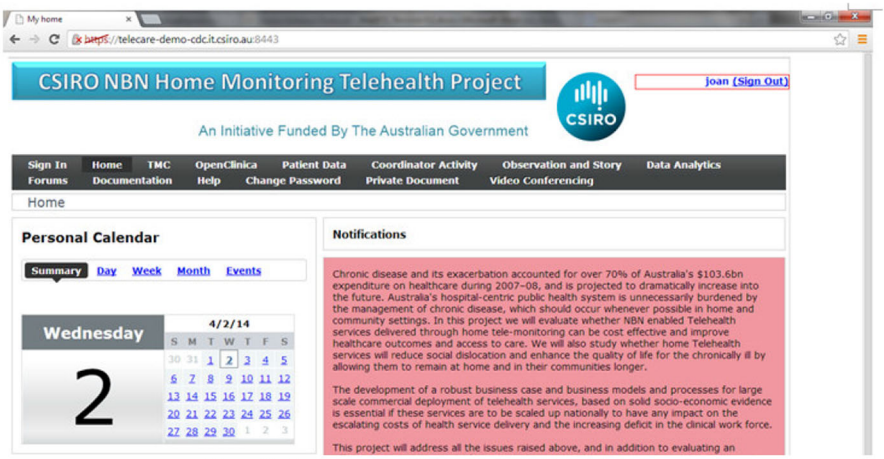

Figure 3. The snapshot of the (part of) CSIRO portal

Fig. 3 The snapshot of the (part of) CSIRO portal

culture and organizational change management and business processes. The further details of our portal site and services it offer can be found in [29].

iiNet,[3] the second largest internet service provider in Australia, is our technology partner. iiNet provides the connection of high bandwidth broadband (i.e., up to 100 Mbps wired broadband and up to 12 Mbps wireless broadband) into the residence of our 150 test patients. Utilizing such high broadband capacity, the project plans to offer a high definition (HD) video conferencing facility. The video conferencing we plan to offer is to connect our trial test patients with their clinical care coordinators (CCC) to discuss any concerns and issues that they may be facing in the management of their condition and their daily life. This paper focuses on the design and implementation of such a video conferencing system.

3 Related work

3.1 Video and audio compression

The core technology used in a videoconferencing system is the digital compression of audio and video streams in real time. Until 1990s, hardware and software were not capable to deal with audio and video streams which were usually huge compared to their text counterpart. For example, a PC usually had an XGA monitor with a resolution of 640 × 480 pixels at 16 bits per pixel. On the other hand, video had a resolution of 320 × 240 pixels. At a video refresh rate of 24 frames per second, the data bus on the PC had to process 320 × 240 × 2 (bytes per pixel) × 24 bytes per second, which works out at about 3.5 MB per second. At this rate, a one minute video would take up 200 MB on the hard drive. There were many problems in processing

[3] www.iinet.net.au.

this amount of file at that time. First of all, this amount of space was not readily available on most PCs of the time. Second of all, the data bus of the PC was not able to handle transferring that amount of data to the video sub-system. Third of all, the CPU was not fast enough to refresh the monitor at the correct frame rate received from the video sub-system. Lastly, the capacity of the network was not enough to transmit such large file from one side to the other side of the network. With the slow computer and dial-up Internet connection, the delivery of media streams took too long for most end-users to bear. Even if the media streams were delivered, they were choppy and pixilated.

The success in the compression and streaming technologies by the late 90s, along with the improved capacity in hardware and software, has made feasible the more widespread use of videoconferencing systems. Compression technology enables reduction on the footprint of the media files on disk as well as transfer of files on the network more efficiently without the need for a high capacity network. Further, streaming technology allowed the media content being played as it is received without having to wait the media file to be downloaded in its entirety. Hardware or software that performs compression and other important tasks is called a codec. A codec digitizes analogue video and audio into digital packets, compresses and encodes the digital packets into data stream, signals for transmission, or decodes for playback. Often encryption technology is incorporated into the codec to encrypt the compressed files in order to improve the security. However, there have been a number of issues associates with codecs and media delivery.

There were concerns regarding the codec compatibility [15,17]. There are a great number of audio and video codecs available in the market because codecs were often designed to emphasize certain aspects of the media to be encoded [35]. For example, a digital video of a sports event needs to encode motion well but not necessarily exact colors, while a video of an art exhibit needs to encode color and surface texture well. Similarly, audio codecs for cell phones need to have very low latency as most cell phone have a small CPU power, disk space, and battery life. In contrast, audio codecs for recording or broadcast need high latency to achieve higher quality audio replay. Different codecs were not compatible and users were inconvenienced by having to install many different codecs for different audio and video streams.

To exchange media streams, both sender and receiver need to understand the specifics of the underlying network configurations (e.g., IP addresses and port numbers) and the details of the media constraints (e.g., the details of the codec, media type, etc.). Though there were a number of popular media exchange communication protocol existed, such as SIP and Jingles, a vast amount of codecs were exchanged using ad-hoc communication protocols [35]. These protocols were often proprietary and were not always compatible from each other [15,17,33].

There was also the inconvenience of having to download plug-ins and install extra software to display media data [13,16,17], often called media players. Media players were usually downloaded either as stand-alone software or as a browser plug-in by the end users to play audio and video content. As codecs were incompatible to each other, the end users required to install many different media players. With the versions of the codecs and media players changing over the years, maintaining them becomes too cumbersome for the majority of naïve users [35].

Complex configuration setting for different media devices was also identified as a cause of stress [15,19]. As the popularity of video and audio usage increases along with the advancement of the networking technologies and capacity, there were many companies producing hardware audio and video devices (e.g, webcam, microphones, monitors, speakers, etc.). The media capacity and constrains (e.g., video resolutions, capacity to deal with audio bitrates, etc.) of a device from a company was different from others. This was because of the competition to produce better quality audio and video devices among the companies to gain better market advantage. The media capacity and constraints were often required to be configured manually in many plug-ins to maximize the outcomes. However, not all end users were equipped with the knowledge of distinguishing the different capacity and constraints of the diverse multimedia devices.

3.2 Video conferencing system

With the advancement in the media technology, many offering in the video conferencing systems have been appeared both from commercial and research industries. In the realm of commercial offerings, Skype [38] is arguably the most well known technology in this space. Skype runs on a wide range of OSs and most mobile and desktops. Skype allows these registered users to communicate through both instant messaging and voice chat which uses a proprietary audio codec. Skype also supports group video calls with three or more people. However, only Windows 7/XP/Vista and Mac users can start a Skype group video call although mobile device users can join it. Facetime [40] is an application created by Apple and is exclusively for iOS-based devices. Like many Apple products, it provides a simple user interface that comprises of only three buttons to communicate with other people and a contact list to choose whom to contact. The user can only choose a single person for a video chat. Google Hangout+ [39] offers web-based video calls on Android, iOS, and the Web using Chrome extension. It also offers a group video chat up to 10 participants. With Hangouts, the user can easily switch views by tapping the contact or the application will automatically switch when someone is talking. Though these are not particularly designed for medical use, many telehealth services utilize them due to their wide acceptance of use and free of charge condition.

In research oriented open source offerings, ooVoo [41] allows the user to have multi party video conferences with other people. However, when talking in text, the user cannot see the video conferencing facility. Camfrog [42] allows a user to chat in public chat rooms with other users over text, voice and video. Video chat is only restricted to do by a single person at a time. The Fuze Meeting [43] is an online meeting application that allows the user to have multiparty video conferences. It also allows document sharing and annotation as well as shared video watching. The user interface is not as simple as the other video conferencing applications with many features hidden in sub menus and the manual that details the insight of the platform is not available.

Video conferencing systems that allow multiple participants by the use of centralized distribution and call management system are offered by large telecommunication companies, such as Cisco TelePresence [47] and PolyCom RealPresence [48]. They provide high-end comprehensive video conferencing systems for cooperates and large organizations, most often with high purchasing price.

Video conferencing systems specially designed for telehealth services have started to appear in the commercial market and academic research to meet the popular demand. For example, Frontline HD from Audisoft technology [44] provides a HD camera that can be carried around by doctors in their visits to patient outside medical centers and then automatically send the captured image to the server. Visimeet from IOCOM [45] offers a video conferencing and collaboration tool that scales from desktop users with a single camera to high definition tele-conferencing rooms with multiple cameras using the same software and interface. Mirial [46] provides a comprehensive solution for remote audio and video contact services between doctors and patients while Polycom provides a number of solutions for immersive telemedicine education system [36]. Virtual critical care unit (ViCCU) [11] and Echocardiographic Healthcare Online Networking (ECHONET) [12] were two experimental research-based technologies developed by CSIRO that combine video conferencing capability to deliver cost-effective access to specialist services for those Australians who live far from major hospitals.

3.3 WebRTC

WebRTC promises to deal with the many difficulties associated with communicating media. WebRTC is a collection of standards, protocols, and JavaScript APIs. The collection enables audio, video, and data sharing between browsers in a peer to peer (P2P) fashion. Instead of relying on third-party plug-ins or proprietary software, WebRTC turns real-time communication into a standard feature that any web application can leverage via a simple JavaScript API [13]. Two standardization bodies work in parallel in the development of WebRTC.

- Web real-time communications (WEBRTC) Working Group [30] at W3C is responsible for defining the browser APIs through the use of HTML5. HTML5 represents an application and its data in a structured way and enables developers to style the web application with CSS and control it via JavaScript.
- Real-time communication in Web-browsers (RTCWEB) [31] is developed by IETF Working Group. It defines the protocols, data formats, security, and all other necessary aspects of communication to enable peer-to-peer communication in the browser.

Delivering rich, high-quality, real time communication applications such as audio and video teleconferencing and peer-to-peer data exchange requires a lot of new functionality in the browser [35]. Figure 4 illustrates the overall architecture of WebRTC proposal that shows the minimum features that need to be supported by browsers for the real time communication [13].

Voice Engine The voice engine (also often referred to as audio engine) is a framework that deals with audio streams. The voice engine supports audio codecs. The agreed audio codecs that need to be supported by WebRTC compatible browsers include: iSAC (i.e., a wideband audio codec for VoIP and streaming), iLBC (i.e., a narrowband speech codec for VoiP and streaming audio), and Opus (i.e., a codec that supports bitrate encoding). A dynamic jitter buffer is supported for error concealment algorithm to conceal the negative effects of network delay jitter and packet loss. Echo concealment,

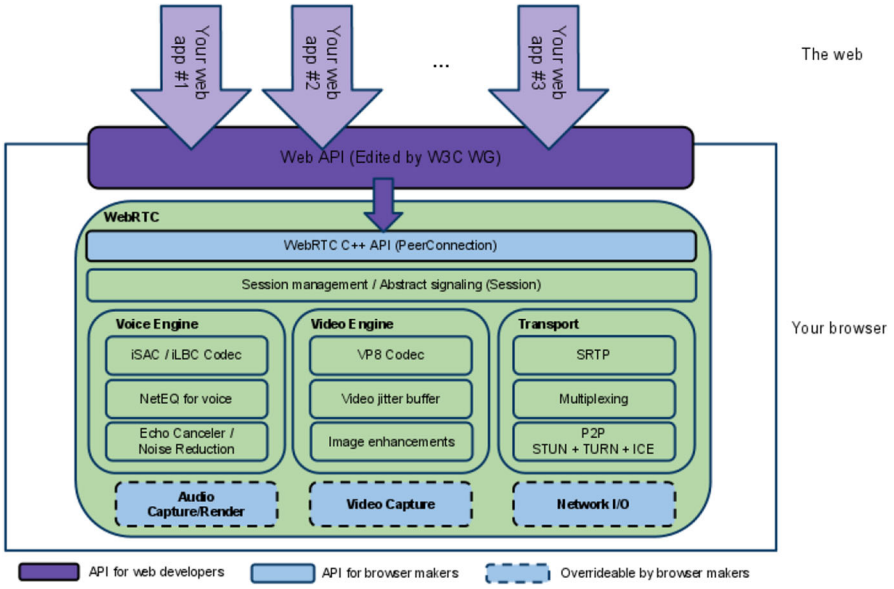

Fig. 4 WebRTC structure (from WebRTC website [13])

a process refers to remove the acoustic echo resulting from the voice being played out coming into the active microphone, is supported in real time too. Furthermore, the noise reduction system is provided to remove certain types of background noise usually associated with voice over the Internet Protocol.

Video Engine The video engine is a framework that deals with video streams which come from the camera to the network and from the network to the screen. Similar to the voice engine, the video engine supports video codecs. The agreed video codec at the time of writing is VP8. It supports dynamic jitter buffer for video which helps to conceal the effects of delay jitter and packet loss to improve the overall video quality. It also supports image enhancement, for example, removing video noise from the image capture by the webcam.

Transport Support This defines activities required for peers to communicate and pass audio/video streams and other data over the transport layer. Peer to peer (P2P) allows the peers to communicate directly by establishing a secure channel no matter what types of network topology peers reside on. A support for secure real time protocol is provided as WebRTC recommends all data to be encrypted at all time.

Session Management WebRTC only defines abstract signalling process and the protocol to describe the session information, and leaves the details of the signalling implementation on the hand of application developers. The rationale is that different applications may prefer to use different protocols, such as the existing SIP or Jingle, or something custom-made for a particular application. In this approach, the key information that needs to be exchanged is the multimedia session description, which specifies the necessary transport and media configuration information necessary to establish the media plane. WebRTC only provides the session description protocol.

WebRTC Native C++ API This API layer enables browser makers to easily implement the Web API for their browsers.

Though the description of overall structure of WebRTC above appears complex, in practice, the complexities are typically hidden from developers. The developers only need to access a Web API layer defined by W3C WebRTC working group. According to the current Web API [30], the browser abstracts most of the mentioned complexity behind three primary APIs. With the support of the following three APIs, developers only require to write a dozen lines of JavaScript code to facilitate a rich teleconferencing experience.

MediaStream A MediaStream is an abstract representation of actual data stream of audio or video acquired from the underlying platform.

PeerConnection A PeerConnection lets two users communicate using browsers. Communications are coordinated via a signalling channel. Once the calling browser establishes a peer connection, it can send MediaStream objects directly to the remote browser.

DataChannel The DataChannel provides a generic transport service that enables browsers to exchange generic data (e.g., chat messages) between peers.

We explain these APIs again in our implementation section later.

4 Architecture and design

4.1 Architecture

WebRTC is the first standard that enables peer-to-peer distribution of multimedia content natively in web browsers. We discuss the architecture and design choices we adapted from the WebRTC to use in the various parts of our system. The overall architecture is shown in Fig. 5.

Patient at Home In addition to TMC telehealth monitoring device installed in their homes, our design includes providing test patients with a Samsung Galaxy Note (sponsored by Samsung Inc.). We ensure each of the Galaxy Note is equipped with a Chrome browser (for Android) that supports WebRTC. The patient uses a native application to connect to a client application which is a simple web page. Using the web-based client application, the patient makes video conferencing requests to one of the care coordinators (CCCs).

Clinical Care Coordinator (CCC) The care coordinator in our trial conducts a number of activities as part of providing daily care to patients. These include home visits, phone calls to family members, and calls to patients or their carers. The CCCs in our trial typically resides remotely from patients in one of the six sites maintaining a regular contact with the patients. Each site typically runs its own intranet and operates within a firewall. The CCCs connect to our portal to record their activities in dealing with patients during the trial. They also use the portal to see any online meeting requests and conduct online meetings with the patients.

CSIRO Portal our portal provides APIs and functionalities required for real-time video conferencing.

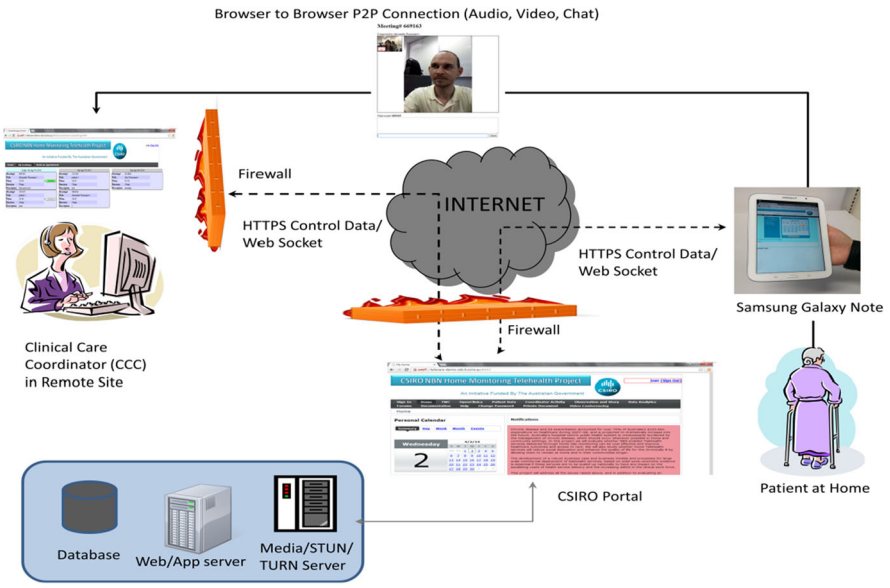

Fig. 5 Video conferencing architecture

– *Web/App Server* The CSIRO portal is supported by a web server that also hosts web applications for the video conferencing. This includes a web-based video conferencing request application for our enrolled test patients as well as a web-based video conferencing scheduling and management application for CCCs. These web applications reside within CSIRO and are protected by a firewall. All video conferencing requests are authenticated by checking the identity of the senders ensuring that the requests are from the devices owned by one of our test patients. Once the identities are authenticated, the requests are authorized.
– *Database Server* We use the database to support video conferencing applications such as recording authorized video conferencing requests. Further information related to video conferencing is also saved, for example, when a video conferencing session was conducted, how long it was conducted, details of the participants, to name a few. We plan to study the information to understand the impact of adding video conferencing feature in our project.
– *Media/STUN/TURN Server* We also host a media server that assists for real-time video/audio streaming communication such as signaling. In P2P communication, many peers often reside behind the layers of firewalls and network address translation (NAT). In this situation, their public IP addresses and port numbers for communication are hidden from their peers. To address this problem, session traversal utilities for NAT (STUN) is used to assist in getting public IP addresses and port numbers of users. STUN makes possible to find peers behind firewalls and NATs. However, in practice, STUN is not sufficient to deal with all NAT topologies and network configurations. Unfortunately, in some cases, media data may be blocked altogether by a firewall or some other network appliance. To address this issue, traversal using relays around NAT (TURN) is used as a fallback to direct the media to specific intermediaries whenever STUN fails. According to one report [35], about

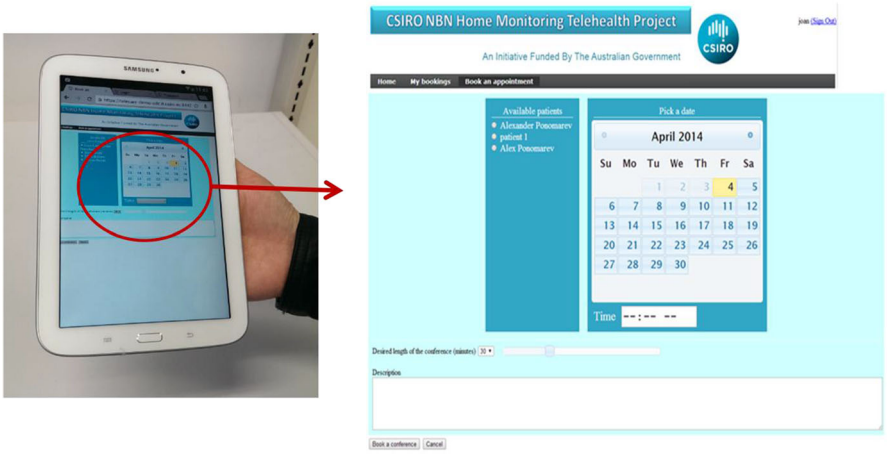

Fig. 6 Video conferencing session request screen

92 % of peers are found using STUN server and the remaining 8 % of connections need to rely on TURN server. We use TURN server as a proxy to relay the traffic directly between peers.

4.2 Video conferencing requests

The video conferencing request system is a simple web-based application that is accessible through the web browser. When a patient needs a face-to-face conversation with his/her care coordinators, the patient can use his/her Samsung Galaxy Note to access the video conferencing request system (see Fig. 6).

The system provides a simple user interface where the patients can define: which case coordinator he/she wants to talk to, a date/time that is convenient to have an online meeting, how long the meeting is going to last and a simple text box to write down a short description of the nature of the request.

Though the request system would look like a typical web-based application, a number of operations are carried out behind it. For example, the system collects network configuration (e.g., public IP address and port number) and media capabilities (e.g., types of media to be changed, what codecs and resolutions can be handled by the browser) and send them to its peers. This process in the P2P communication is typically called as a term `offer`. The request details and the `offer` are sent to the CSIRO portal over a secure channel.

4.3 Video conferencing scheduling and management

We provide clinical care coordinators (CCCs) with a web-based application to check any online meeting requests and a service to connect them to the patient. The video conferencing scheduling and management application lists the video conferencing requests for a particular CCC after authenticating him/her (see Fig. 7).

Fig. 7 Video conferencing scheduling and management screen

The video conferencing requests are grouped together by date. For each meeting request, the automated meeting number appears to distinguish one meeting from others. This is followed by the display of the patient who sent the request. The meeting details such as date/time, duration, and the description of the meeting are also displayed. The presence of the patient, whether he/she is connected online at the time the meeting requests are being displayed, as indicated by the green connect button in the figure. If the patient is not online, the connect button is grayed out and cannot be activated.

One of the important functionalities of the scheduling and management system is its ability to carry out a peer discovery. The peer discovery helps to know whether the patient with online meeting request is online at that moment. The heart of the peer discovery is a signaling mechanism that finds any present peers and then coordinates communication between the peers. It should be noted that signaling methods and any protocols necessary for signaling are not specified in the WebRTC. Instead, it is all left to the hands of WebRTC application developers to choose whatever messaging protocol they prefer to implement depending on the nature of the application. Given the limited number of patients and CCCs interacting with them in our trial, we decided that we do not need a high volume and high performance signaling mechanism that are popularly available in commercial video conferencing products. Instead, we use a simple signaling server to obtain the presence of a peer by sending a web socket connection request and validating the response through Java APIs.

4.4 P2P multimedia delivery

The connect button enables the connection between a care coordinator and a patient. To allow a CCC and a patient to see and hear each other, the application needs to capture audio and video streams from the underlying platform before anything happens.

Enabling a rich video conferencing experience in the browser requires that the browser is able to access the system hardware (e.g., webcam, microphone, monitor, etc.) to capture both audio and video. Simply capturing raw audio and video streams is not sufficient as each stream must be digitized, compressed, encoded, and synchronized

to be able to travel through the continuously fluctuating bandwidth and latency between two peers [35]. On the receiving end, the process is reversed, and the client application must decode the streams in real-time and be able to adjust to network jitters and delays. In short, capturing and processing audio and video is a complex problem. Fortunately, the WebRTC engine, implemented in WebRTC enabled browser, hides all the complexity of processing audio and video. Our application simply uses a function from getUseMedia() API to acquire audio and video streams. These media streams are then automatically optimized, encoded, and decoded by the WebRTC audio and video engines. Once the audio and video streams are captured, it is time to deliver the streams using the signaling service.

As a part of signaling service, the `offer` is described inside a session description protocol (SDP) and exchanged with the patient's browser using JavaScript session establishment protocol (JSEP). The use of JSEP abstracts all the inner workings of SDP behind a few simple method calls on WebRTC APIs. If the patient is ready for a teleconferencing upon receiving the signal, the patient clicks the take a call button. The WebRTC video and audio engine at the patient's browser then captures audio and video streams and is ready to exchange them with the care coordinator.

However, most care coordinators reside within their own private networks behind one or more layers of firewalls and NATs. As a result, the patient's browser does not know the possible public IP address and port number to connect to the CCC's browser directly. To overcome the issue, our application makes connection to the external session traversal utilities for NAT (STUN) to obtain the allocated public IP address and port number tuple of the CCC's browser. In the case where both patients and care coordinators are residing behind firewalls, we utilize a TURN server to act as a proxy to connect them directly given that both of their public IP address and port number tuple are discoverable by the STUN. Once again, WebRTC provides APIs to do this process easily by the use of interactive connectivity establishment (ICE). The built-in ICE protocol within the WebRTC enabled browsers automatically begins the process of discovering all the possible candidate IP address and port number tuples and connecting them once STUN/TURN servers are specified and enabled.

By now, a peer to peer connection is established along with the video and audio streams ready for transport. At this point, we still need a couple of mechanisms in place to support security and reliability. WebRTC requires that all communication must be encrypted while in transit. To satisfy this requirement, we utilize datagram transport layer security (DTLS) handshake to obtain the secret keys for encrypting media data. The encrypted media data is then transported. After the encrypted media data arrives at the patient's browser, the browser's video and audio engines decrypt, decode and display the media content to the patient as illustrated in Fig. 8. In addition to the video conferencing sessions, we also provide chat room to exchange text messages.

5 Prototype implementation

In this section, we describe a prototype implementation to illustrate how some of the design ideas described earlier have been realized in the video conferencing system using WebRTC.

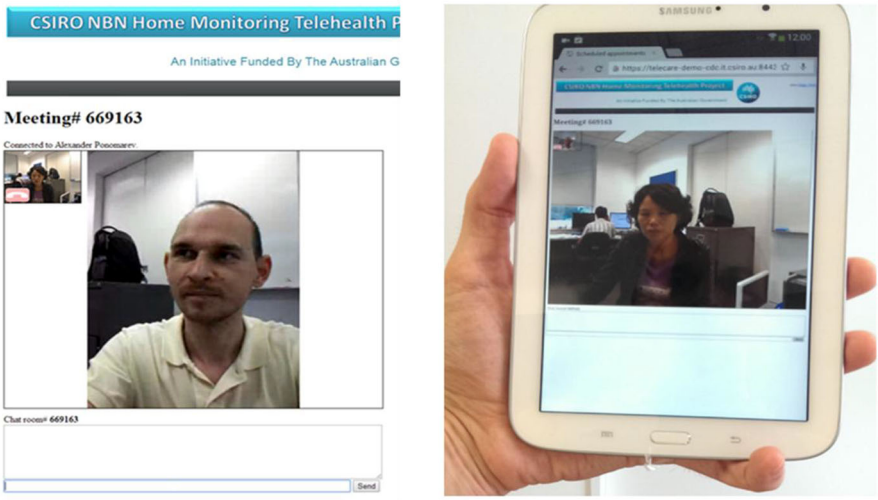

Fig. 8 A video conferencing screens at both sides

5.1 Environment set up

Though the WebRTC is still at a draft stage, there are number of open source projects and commercial platforms available with the promise to assist in the fast and effective development of WebRTC based video conferencing applications with the minimum efforts from the developers. Among those, we decided to use EasyRTC[4] which is a full-stack open source WebRTC toolkit that supports the building of secure WebRTC applications. EasyRTC is a bundle of web applications, code snippets, client libraries and server components written and documented to work out of the box. We use EasyRTC APIs and JavaScripts to access the functionalities of WebRTC engines already implemented in many browsers. We use Chrome browser installed in the Samsung Galaxy Notes. We also use Node.js,[5] which is a JavaScript based runtime platform, as a web server to develop our web applications.

5.2 Acquiring audio and video streams

In practice, the complexity of representing the video and audio streams and specifying the constraints of the media streams are all hidden from the developers. The following code snippets illustrate how easy it is to define a local stream (i.e., getting a video and audio stream from the local machine).

[4] www.easyrtc.com.

[5] http://nodejs.org/.

Fig. 9 Browser asks for a permission to access the local camera and microphone

```
easyrtc.initMediaSource(
    function(){
        var selfVideo = document.getElementById("me");
        easyrtc.setVideoObjectSrc(selfVideo,easyrtc.getLocalStream());
    },  easyrtc.connect("VC test",connectSuccess,connectFailure);
    connectFailure
);
```

It is important to note the call to easyrtc.getLocalStream and easyrtc.setVideo ObjectSrc. The former gets a video and an audio stream from the local camera and microphone, once the call to easyrtc.initMediaSource succeeds. The latter ties a video tag to a media stream object. Invoking this method will cause the user's browser to ask for permission to access the requested local camera and microphone as seen in Fig. 9. Once the permission button is clicked, the users see their own images on the screen.

Obtaining a remote peer's media stream is also straightforward by using a callback method. The callback method ties the video tag to the incoming stream. When the remote peer hangs up, the callback clears the video tag.

```
easyrtc.setStreamAcceptor(
    function(callerEasyrtcid, stream) {
        var video = document.getElementById('caller');
        easyrtc.setVideoObjectSrc(video, stream);
    });

easyrtc.setOnStreamClosed(
    function (callerEasyrtcid) {
        easyrtc.setVideoObjectSrc(document.getElementById('caller'), "");
    });
```

In addition, EasyRTC provides a number of functions for developers to set up media constrains. For example, calling `easyrtc.setVideoBandwidth()` allows to set the bandwidth used to send and receive each media stream's video track.

5.3 Signalling and peer connection

A signaling process assists the finding of peers and establishing communication among peers by exchanging data through dedicated channels. The dedicated channel allows the privacy and security of the data from the interference of concurrently running processes. The implementation of signaling process can vary depending on the requirements of each application and the environment the application runs on. For example,

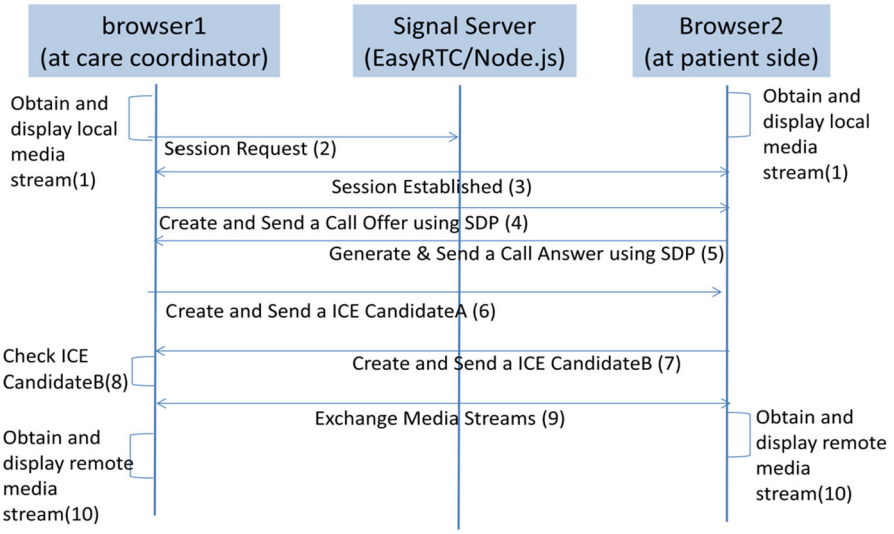

Fig. 10 Signaling and interactions

if an application only requires communication among peers within the same network, it is relatively straightforward to obtain public IP addresses of the peers and make the connections. However, a signaling process becomes complex if peers' public IP addresses and port information are hidden away from peers as they are located behind their own private network. As a result, neither peer is directly reachable by each other. To initiate a session, one must first gather the possible IP addresses and port candidates for each peer, traverse the NATs, and then run the connectivity checks to find the ones that work.

The PeerConnection API in the WebRTC is responsible for managing the full life cycle of each peer-to-peer connection by encapsulating all the connection setup, management, and state within a single interface. However, before the application developer dives into the details of each configuration option of the API, one needs to understand the interactions among peers before choosing a right signaling process. Figure 10 illustrates our signaling requirements and interactions required between peers.

(1) When a care coordinator clicks a "connect" button, the care coordinator's browser obtains and displays the local media stream.
(2) We then need assistance from a signal server to create a secure channel between the care coordinator's browser (browser 1) and the patient's browser (browser 2) which is requesting a particular video conferencing session.
(3) A session is established.
(4) The browser 1 uses session description protocol (SDP) to describe the session profile which contains information such as types of media to be exchanged, codecs and their settings, and bandwidth information. The SDP is used to make an offer to the browser 2. At this stage, actual media itself is not attached to the offer.

(5) Upon receiving the offer, the browser 2 creates an answer that it is willing to connect to the browser 1 and also sends its corresponding session profile using SDP. Now the browser 1 knows that the browser 2 is ready to run a peer to peer communication.
(6) After getting an answer from the browser 2, browser 1 creates an interactive connectivity establishment (ICE) agent (ICE A). The ICE agent gathers local IP address and port tuple and queries an external STUN server to retrieve the public IP and port tuple of the peer.
(7) Upon receiving the ICE A, the browser 2 performs the same operation as browser 1 to create and send ICE agent (ICE B).
(8) Browser 2 checks that the public address received in ICE B matches with the information received earlier (i.e., the browser 2 sends this information when the patient clicks video conference reservation request). At this point, if the browser 1 and browser 2 cannot establish a connection directly as P2P fashion, the TURN server is used as a proxy to relay traffic.
(9) Browser 1 sends the media stream to browser 2. Likewise, browser 2 sends its local media streams to browser 1 after obtaining the public IP addresses and port number tuples from the ICE A received earlier.
(10) At this stage, both browsers start displaying media contents and the video conferencing is working.

The above mentioned interactions clearly demonstrate a need for STUN/TURN servers and a signal server that can handle a small number of participants. EasyRTC supports a signaling server that fits our requirements. Though the interactions in our case look lengthy and intricate, the complexity is actually all wrapped together by the signaling process. All we had to do in our case was to define a STUN/TURN server and adding a few lines of JavaScript code.

The enabled STUN/TURN server enforces the EasyRTC to go through the external STUN/TURN server to get public IP addresses and ports of the peers that interact in our application. If this set up is omitted, the EasyRTC will only attempt the direct peer to peer connection within the same network using the local IP addresses. The following code excerpt illustrates how to specify STUN/TURN server in the code and direct the EasyRTC to use the configuration.

```
var onGetIceConfig = function(connectionObj, callback) {
      var myIceServers=[
              {url:'stun:stun.l.google.com:19302'},
              {url:'stun:stun1.l.google.com:19302'},
              {url:'stun:stun2.l.google.com:19302'},
              {url:'stun:stun3.l.google.com:19302'},
              {url:'turn:tele@telecare-demo-cdc.it.csiro.au:3478?transport=tcp',
                      credential: 'test',
                      username: 'test'}
      ];
}

easyrtc.on("getIceConfig", onGetIceConfig);
```

The following JavaScript is added to make peer to peer connections. With the STUN/TURN server enabled, EasyRTC makes a number of decisions on our behalf. In the background, it initializes the `PeerConnection` with a public STUN/TURN server for NAT traversal by creating ICE agents, requests audio and video streams

with getUserMedia, and initiates a WebSocket connection to establish a session with its own EasyRTC signaling server and passes the media streams between the peers.

```
function performCall(easyrtcid) {
    easyrtc.call( easyrtcid,
        function(easyrtcid) {
            console.log("completed call to " + me);},
        function(errorMessage) {
            console.log("err:" + errorMessage);},
        function(accepted, peers) {
            console.log(
                (accepted?"accepted":"rejected")+ " by " + peers);});
}
```

5.4 Chat rooms

Rooms are a compartmentalizing feature of EasyRTC that we use to create chat services. The chat service allows a care coordinator and a patient to exchange text messages in addition to the online meeting they are conducting. To create a chat service, both client and server codes need to be implemented. On the client side, first the client connects to a socket.io server to get a chat channel. Once connection is established, the client sends a chat message.

```
//connection to socket.io server
var chat = io.connect(window.location.protocol + '//' + window.location.host +
'/appointments);

//sending a chat message
function sendChatMessage(val) {
//sending message to the server
chat.emit("chat message", {text: val });
}
```

Once the server receives the 'chat message' from the client, it will fire the 'chat message' in the current room to every joined participant.

```
var chat = socketServer.of('/appointments').on('connection', function(socket) {
      socket.on('subscribe', function(data) {
               socket.join(data.room);
               roomName = data.room;
      });
      socket.on('chat message', function(data) {
               chat.in(roomName).emit('chat message',
                        {'text' : txt, 'from' : userName , 'userId' : userId});
    });
});
```

Upon receiving the 'chat message' by clients from the server, the client parses the incoming data and display the text in the chat log.

```
function initChatRoom(appointmentID)
{
      //join the chat room
      chat.emit("subscribe", {'room': appointmentID });

      //receive the chat message sent by the peer (sent via the server)
      chat.on("chat message", function(data) {
            chatlog.append(data.from + ": " + data.text + "\n");
      } );
}
```

5.5 Limitations

We have designed and implemented a simple video conferencing application using WebRTC for our purpose. The current design and implementation has a number of limitations. We next describe these limitations and how we address them in future.

Currently, our video conferencing system only offers one to one call. However, it is easy to imagine that a patient may need to have online meeting with multiple participants in a single request, such as nurses, GPs and specialists. To allow multi party video conferencing, we could use multiple peer connections so that every endpoint connects to every other endpoint like a mesh configuration. This should still work well for a small number of peers although the processing and bandwidth consumption becomes heavy, especially when more mobile clients are added (i.e., because mobile devices runs on a small memory space and depends on a battery life). Alternatively, we could expand our application to set up WebRTC endpoint on a server and use it to distribute streams to all others. This model will give more flexibility as it enables our application to handle media routing by choosing which other peers to connect to. A better option for a large number of peers is to use a Multipoint Control Unit (MCU). MCUs have been popularly used in the non-browser peer to peer real time communication. In this model, a MCU works as a bridge to distribute media between a large numbers of participants. MCUs can cope with different resolutions, codecs, and frame rates within a video conference, and also do selective stream forwarding as well as mixing or recording audio and video [32,33]. We plan to implement MCU in our current prototype in future.

For signaling process, our web application uses a signaling server that is built with Socket.io and Node.js supported by EasyRTC. This design choice makes it simple to build a service to exchange messages. Socket.io is particularly suited to WebRTC signaling because its built-in concept of rooms enables peer to peer pairing easily and a chat room set up between the peers simple [33]. This model only works well for a relatively small number of users like our current setting. However, the model would not scale up well as a production-grade signaling service for a large number of participants. The model needs to be upgraded to a signal sever that can handle high volume participants, something similar as SIP and Jingle. Priologic Software, the company that distributes the free EasyRTC, has been working on SIP integration methods commercially [34].

The current chat room service only allows each participant to register to a specific chat room to participate in text exchange. This model also does not scale up well if participants are frequently changed. The better model would be that a number of participants can be grouped together as a role. Instead of registering each participant to a room, a role can be assigned to a room connection to allow for greater flexibility.

6 Conclusion and future work

This paper proposes a standard-based interoperable, simple and inexpensive approach to develop a practical video conferencing system using emerging WebRTC technology. The WebRTC approach promises to assist developers in building a browser-based peer

to peer real time communication which is often required for multimedia systems such as video conferencing. We have described the system architecture and the models of components that we adapted from WebRTC for use in our trial. We also describe various insights of the prototype implementation and provide code snippets. Furthermore, we have described some of the limitations of our current system that is implemented under the assumption of a small number of participants operating within strict constraints. In the future, we plan to address those limitations in order to enable our video conferencing system to be able to handle high volume participants.

Performance evaluation is an important part of an application development to ensure that the operations implemented within the application work within a reasonable expectation. There are many different places within our implementation where the performance of a certain component can be evaluated. For example, the response time for acquiring video and audio streams can be measured on different browsers or the same browser but different versions or under different operating system that interacts with the browsers. Response time for the signaling process can be also measured on different signaling mechanisms and protocols to evaluate the effectiveness of the session establishment and negotiation. We plan to perform a detailed performance evaluation of the system in the near future. In addition, we plan to measure the quality of the audio and video streams under different situations.

Acknowledgments We would like to thank our industrial trainee Alex Ponomarev for the contribution towards the development of the prototype implementation. The research is funded by Australian Government Department of Health and Ageing under Commonwealth Telehealth Pilots program. We also would like to acknowledge and thank our partners: Department of Human Services, ACT Health, ACT, Northern District Health Service, TAS, Townville Mackay Medicare Local and Health District, QLD, Ballarat Health Services, Grampians Health Alliance, VIC, Nepean Blue Mountains Medicare Local and LHD, NSW, Anglican Retirement Villages, NSW, TelMedCare, iiNet, and National Brodband Network (NBN) Co.

References

1. Polisena J, Coyle D, Coyle K, McGill S (2009) Home telehealth for chronic disease management: a systematic review and an analysis of economic evaluations. Int J Technol Assess Health Care 25(03):339–349
2. Stamm BH (1998) Clinical applications of telehealth in mental health care. Prof Psychol Res Pract 29(6):536
3. Bahaadinbeigy K, Yogesan K (2012) A literature review of teleophthalmology projects from around the globe. In: Digital Teleretinal Screening. Springer, Berlin, pp 3–10
4. Polisena J, Tran K, Cimon K, Hutton B, McGill S, Palmer K (2009) Home telehealth for diabetes management: a systematic review and meta analysis. Diabetes Obes Metab 11(10):913–930
5. Finkelstein SM, Speedie SM, Potthoff S (2006) Home telehealth improves clinical outcomes at lower cost for home healthcare. Telemed J e-Health 12(2):128–136
6. Mohan J, Razali R, Yaacob R (2004) The Malaysian Telehealth Flagship Application: a national approach to health data protection and utilisation and consumer rights. Int J Med Inf 73(3):217–227
7. Hebert MA, Korabek B, Scott RE (2006) Moving research into practice: a decision framework for integrating home telehealth into chronic illness care. Int J Med Inf 75(12):786–794
8. Bird SR, Kurowski W, Dickman GK, Kronborg I (2007) Integrated care facilitation for older patients with complex health care needs reduces hospital demand. Aust Health Rev 31(3):451–461
9. Australian Department of Health (2003) NBN Enabled Telehealth Piolets Program. http://health.gov.au/ehealth-nbntelehealth. Accessed 28 Jan 2014

10. PolyCom (2012) An Introduction to the Basics of Video Conferencing. http://www.polycom.co.uk/content/dam/polycom/common/documents/brochures/video-basics-br-engb.pdf. Accessed 28 Jan 2014
11. Cregan P, Stapleton S, Wilson L, Qiao RY, Li J, Percival T (2005) The ViCCU project-achieving virtual presence using ultrabroadband internet in a critical clinical application-initial results. Stud Health Technol Inf 111:94–98
12. Wilson LS, Stevenson DR, Cregan P (2010) Telehealth on advanced networks. Telemed e-Health 16(1):69–79
13. WebRTC Official Website. http://www.webrtc.org/. Accessed 28 Jan 2014
14. Rhinow F, Veloso PP, Puyelo C, Barrett S, Nuallain EO (2014) P2P Live Video Streaming in WebRTC. http://portovep.com/static/docs/2013-webrtc-p2p-live-streaming.pdf. Accessed 28 Jan 2014
15. Loreto S, Romano SP (2012) Real-time communications in the web: Issues, achievements, and ongoing standardization efforts. IEEE Internet Comput 16(5):68–73
16. Eriksson GP, Hakansson S (2012) WebRtC: enhancing the web with real-time communication capabilities. Ericson Rev 5–9
17. Romain C (2013) Some WebRTC opportunities for RCS: And some inner challenges to overcome. In: 17th international conference on intelligence in next generation networks (ICIN). IEEE Press, New York, pp 31–38
18. Bertin E, Cubaud S, Tuffin S, Crespi N, Beltran V (2013) WebRTC, the day after: what's next for conversational services? In: 17th international conference on intelligence in next generation networks (ICIN), IEEE Press, New York, pp 46–52
19. Rodríguez P, Cerviño J, Trajkovska I, Salvachúa J (2013) Advanced Videoconferencing Services Based on WebRTC. In: Proceeding of IADIS multi conference on computer science and information systems
20. Fernandez LL, Diaz MP, Mejias RB, Lopez FJ, Santos JA. (2013) Kurento: a media server technology for convergent WWW/mobile real-time multimedia communications supporting WebRTC. In: 14th international symposium and workshops on a world of wireless, mobile and multimedia networks (WoWMoM). IEEE Press, New York, pp 1–6
21. Elleuch W (2013) Models for multimedia conference between browsers based on WebRTC. In: IEEE 9th international conference on wireless and mobile computing, networking and communications (WiMob). IEEE Press, New York, pp 279–284
22. Australian Government, Department of Health and Ageing. Building a 21st Century Primary Health Care System. Australia's First NationalPrimary Health Care Strategy. http://www.yourhealth.gov.au/internet/yourhealth/publishing.nsf/Content/nphc-draft-report-toc/$FILE/NPHC-Draft.pdf. Accessed 28 Jan 2014
23. NHHRC (2014) A Healthier Future For All Australians-Final Report of the National Health and Hospitals Reform Commission-June 2009. http://www.health.gov.au/internet/nhhrc/publishing.nsf/content/nhhrc-report. Accessed 28 Jan 2014
24. Preventive Health Taskforce (2014) Taking Preventative Action-the Government's response to the report of the National Preventative Health Taskforce. http://www.preventativehealth.org.au/internet/preventativehealth/publishing.nsf/Content/taking-preventative-action. Accessed 28 Jan 2014
25. Australian Institute of Health and Welfare (2014) Health Expenditure Australia 2011–2012. http://www.aihw.gov.au/publication-detail/?id=60129544658. Accessed 30 April 2014
26. Litan RE (2008) Vital signs via broadband: Remote health monitoring transmits savings, enhances lives. Better Health Care Together
27. Whole System Demonstrator Programme Headline Findings (2011) https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/215264/dh_131689.pdf. Accessed 28 Jan 2014
28. Darkins A, Ryan P, Kobb R, Foster L, Edmonson E, Wakefield B, Lancaster AE (2008) Care coordination/home telehealth: the systematic implementation of health informatics, home telehealth, and disease management to support the care of veteran patients with chronic conditions. Telemed e-Health 14(10):1118–1126
29. Nepal S, Jang-Jaccard J, Celler B, Yan B, Alem L (2013) Data architecture for telehealth services research: a case study of home tele-monitoring. In: 9th international conference conference on collaborative computing: networking, applications and worksharing (Collaboratecom). IEEE Press, New York, pp 458–467
30. W3C (2014) WebRTC 1.0: Real-time Communication Between Browsers. http://dev.w3.org/2011/webrtc/editor/webrtc.html. Accessed 28 Jan 2014

31. IETF (2014) Real-Time Communication in Web-Browsers (RTCWEB). http://datatracker.ietf.org/wg/rtcweb/. Accessed 28 Jan 2014
32. Dutton S (2012) Getting Started With WebRTC. http://www.html5rocks.com/en/tutorials/webrtc/basics/. Accessed 28 Jan 2014
33. Dutton S (2013) WebRTC in the real world: STUN, TURN, and signalling. http://www.html5rocks.com/en/tutorials/webrtc/infrastructure/. Accessed 28 Jan 2014
34. EasyRTC (2013) EasyRTC Documentation. http://www.easyrtc.com/docs/guides/easyrtc_client_tutorial.html. Accessed 28 Jan 2014
35. Grigorik I (2013) High performance browser networking. O'Reilly Media, Sebastopol
36. Polycom (2014) Polycom Solutions for Telemedicine/Patient Care. http://www.polycom.com.au/solutions/solutions-by-industry/healthcare/telemedicine-telehealth.html. Accessed 4 April 2014
37. Bahaadinbeigy K, Yogesan K, Wootton R (2010) A survey of the state of telemedicine in Western Australia. J Telemed Telecare 16:176–180
38. Microsoft Skype. http://www.skype.com/en/features/group-video-chat/. Accessed 30 April 2014
39. Google unveils hangouts: unified messaging system for Android, iOS, and Chrome. http://www.theverge.com/2013/5/15/4332556/google-hangouts-unified-messaging-google-io-2013. Accessed 30 April 2014
40. Apple FaceTime. http://www.apple.com/au/ios/facetime. Accessed 30 April 2014
41. http://www.oovoo.com/. Accessed 30 April 2014
42. http://www.camfrog.com/y. Accessed 30 April 2014
43. https://www.fuzebox.com/products/fuzemeetingy. Accessed 30 April 2014
44. http://www.audisoft.net. Accessed 30 April 2014
45. http://janet.iocom.co.uk. Accessed 30 April 2014
46. http://www.mirial.com. Accessed 30 April 2014
47. http://www.cisco.com/c/en/us/products/collaboration-endpoints/index.html. Accessed 30 April 2014
48. http://www.polycom.com/products-services/realpresence-platform.html. Accessed 30 April 2014