General Title:

Real-Time Streaming of mHealth Data from Multi-Sensor Body Area Networks

by

James Delaney, BA

Dissertation

Presented to the
University of Dublin, Trinity College
in fulfillment
of the requirements
for the Degree of
Master of Science in Computer Science

University of Dublin, Trinity College

August 2013

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

_____

James Delaney

Permission

I, the undersigned, agree that Trinity College Library may lend or copy this the upon request.

_____

James Delaney

Acknowledgements

Firstly I would like to thank my friend Carmel who was there at the beginning to start me on the right road and and there again at the end to get me off it.

I would like to express my gratitude to my supervisor Jonathan Dukes for his valuable guidance over the course of the project. His professionalism and ability to see the "big picture" brought me back on track on more than one occasion. I would like to also thank Meriel Huggard for her support throughout the year and over the course of my dissertation.

Finally, I would like to thank my wife Sharon, my children, and my family for their unflagging support and belief in me over the past year. I most certainly would not have been able to complete the course without them as behind me as they all were. It has been a great year for me and I am grateful to them for giving me the opportunity.

James Delaney

University of Dublin, Trinity College

September 2013

Abstract

Balancing Efficiency, Accuracy and Timeliness in Wireless, Multi-Sensor Body Area Networks for Healthcare Monitoring

James Delaney, BA

University of Dublin, Trinity College, 2013

Supervisor: Jonathan Dukes

Abstract Summary

This dissertation investigates the use of real-time streaming protocols for transport and synchronisation of mHealth data from a wireless body area network (WBAN) to a remote monitoring application. The main streaming protocols are compared and the RTSP/RTP protocol suite is selected as the most suitable. A prototype real-time WBAN monitoring system is designed, implemented and evaluated.

Contents

List of Tables

List of Figures

Chapter 1 Introduction

Longer life expectancies and an increase in "lifestyle" illness such as cardiovascular disease and diabetes in the developed world have created a market for pro-active health monitoring and data-driven diagnostics (Baum and Abadie 2013). Collection of personal health data using sensors and smart phones has become affordable and achievable. People are taking a personal interest in their own health data and from a commercial or medical perspective there are many opportunities to improve healthcare to patients using this data in a live or recorded format.

1.1 Monitoring and the need for Contextual Information

The Holter device is considered as an example of the need for the contextualisation of health data.

When a person suffers cardiac arrest it is often necessary for them to wear a Holter monitor to record their cardiac output. The device is worn for a 24-48 hour period and usually comprises some electrodes for detecting the cardiac rhythms and storage, such as an SD card, to record them.

The wearer must keep an accurate diary of their activities over the course of the testing period (Danbury Hospital 2010). This diary would normally be hand-written and would rely on the patient to record accurate information at the right time. There is a variant of the unit in which the patient presses a button when they have a palpitation.

This contextual information is used by the medical professionals to assess events in the electrocardiography (ECG) with respect to what the patient was doing at the time.

If the patient does not record the event in their diary, or they do it retrospectively, it is likely that events on the data log will not directly match events in the diary.

Combining the heart rate monitor with other sensors present in smart phones like accelerometers or with sensors in other embedded devices would provide more accurate contextual information for the medical practitioner.

Once synchronised, this data could be stored on the embedded device or on the Smartphone, uploaded to a server or displayed in real-time.

e.g. Some event is registered on the heart-rate monitor at 2pm. The Smartphone alerts the patient and asks them to identify what they were doing from a list of activities, e.g. running, walking, sitting, other. This is recorded in the diary with the activity level as recorded by the accelerometer. The doctor can review the diary on their computer and can look for such significant events. For each event, the diary entry can be cross checked with the recorded activity level for a fuller picture of what activity the patient was undertaking at the exact time of the event.

1.2 Requirement for Real-Time Monitoring and Synchronisation

Holter monitors typically record data to their internal storage for upload and analysis at a later time. However, in the case of an emergency event or even for routine monitoring it would be desirable for a medical professional to be able to view a remote patient's biometric output in real-time. A similar use case would be a carer monitoring an elderly person via a smart phone.

A physician may also be interested in other patient details such as temperature, pulse rate, blood pressure or activity level. This is heterogeneous data that is collected at different sample rates and at different precision levels. A real-time monitoring system must be able to synchronise this data in order to provide a meaningful picture of the patient's condition and to allow a physician to make an accurate diagnosis.

1.3 This Dissertation

It was noticed that the real-time and multi-stream synchronisation requirements of a mobile health monitoring application are similar to the requirements of multimedia presentation and distribution technologies. Table 1 outlines some general characteristics of a health monitoring application and

some corresponding features of a multimedia presentation and distribution system such as a video conferencing system.

Mobile Health Monitoring Application	**Multimedia Streaming Protocol**
Continuous sensor data	Continuous video/audio
Multiple sources of sensor data requiring synchronisation	Audio/video must be lip synched
Requirement for end-to-end control to turn on/off monitoring equipment	Requirement for end-to-end control for call setup/scheduling etc
Privacy/authentication of health data	Privacy/authentication of conversations

Table 1 Similar features of a health monitoring application and a multimedia streaming application

Video conferencing protocols such as H.323 provide control and transport mechanisms for continuous streams of audio and video data over the internet. It makes sense therefore to look at the architecture and protocols used to facilitate these multimedia systems.

In this dissertation the state of the art in mHealth monitoring is reviewed. Then existing media transport and control protocols will be examined as a means of packaging sensory data from biometric sensors such as ECG, heart rate and accelerometers. The emphasis in this work is placed on synchronisation and real-time monitoring of the data.

A protocol was selected based on pre-determined criteria. A prototype was designed and developed as a proof of concept using an ECG sensor for biometric data and an accelerometer to generate contextual information in real-time.

The choice of protocol and design was then evaluated using information gathered in the implementation.

Chapter 2 Related Work and Background Information

As the focus of this work is on mHealth monitoring and Wireless Body Area Networks, this chapter starts with an overview of the recent trends in mHealth. It establishes the need for contextual information in an mHealth application and addresses the state of the art in WBAN research.

2.1 Health Monitoring

Non-communicable diseases (NCDs) are the number one cause of death globally: 63% of all deaths were caused by NCDs in 2008 (Alwan 2011). Amongst the deaths cardiovascular diseases (CVDs) are the principle cause, accounting for 48% of all NCD deaths worldwide. Other NCDs include stroke, diabetes and certain types of cancer. The aging population in the developed world means that there is a growing demand for preventative measures and treatments for these types of conditions.

A survivor of a heart-attack, or of a stroke, has a high risk of recurrence and has a high risk of dying from such a recurrence. While prevention is the best cure, once a patient is suffering from a CVD their risk of death can be substantially lowered using a combination of drugs - statins, blood pressure drugs etc. (WHO 2013). Close monitoring of these types of at-risk patients by medical professionals using telemonitoring has been shown to produce real benefits, including increased compliance with drug regimes (Lusignan et al. 2001). Real-time monitoring of a patient's biometric data would clearly be of benefit in emergency situations. Furthermore, with the wider availability of sensor equipment now prevalent in consumer electronics it is conceivable that an application (e.g. on a smart phone) could be programmed to act proactively in the event of the detection of some abnormal reading, telling a user that they should show their collected health data to a cardiologist.

2.2 Metrics

There are many affordable devices coming onto the market that can measure the biomedical data of a patient. Some of the various metrics that can be measured and that would be of interest to a medical professional are summarised below.

2.2.1 Electrocardiography (ECG)/Holter

The Holter monitor or ambulatory electrocardiography device is a machine that continuously records the heart's rhythms. The monitor is usually worn for 24-48 hours during normal activity. The monitor may also be used after a heart attack to diagnose heart rhythm problems or when starting a new heart medicine.

The ECG device comes in a variety of forms, but all include electrodes that are attached to the body, usually on the chest but sometimes on the arms and legs. The device measures the voltages across the electrodes and the activity of the heart is then collated from these measurements. There are now various inexpensive ECG devices in the marketplace designed to be used with low-cost embedded systems. This has reduced the cost of developing and designing portable biomedical monitoring systems.

2.2.2 Blood sugar monitor

More frequent self-monitoring of blood glucose levels has been shown to be associated with clinically and statistically better glycaemic control (Karter et al. 2001). An implantable wireless sensor for continuous glucose monitoring is described in (Ahmadi and Jullien 2009).

2.2.3 Contextual information/Accelerometer

Contextual information can be of use to a medical professional, for example monitoring a patient's exercise regime/diet or activity levels. This information can be obtained from the patient in a number of ways, either actively or passively. A context aware Body Area Network (BAN) that uses the accelerometer in a Smartphone to categorise a patient's physical activities passively using a

movement recognition approach was introduced by (Lau, Konig, and David 2010). An active approach would be to just ask the patient what they are doing when a new activity is detected. For example, a Smartphone app could detect an arrhythmia from its sensors using a policy engine, as in (Jurik and Weaver 2012), and then classify the activity of the person using a simple Q&A either on-screen or using speech recognition.

2.2.4 Other metrics

Other metrics that would allow medical professionals and patients to build a more accurate picture of the health of the patient include blood pressure and body temperature. Both of these can be easily measured using current inexpensive, embedded system devices. Wireless implants are appearing on the market to measure blood sugar levels.

All of these devices collect data points of different data-size per recorded incident and at different frequencies and with various levels of accuracy.

2.3 Wireless Body Area Networks (WBANs)

Whereas eHealth refers to the healthcare practice supported by electronic communication and systems, mHealth refers to the use of mobile technologies to further improve the healthcare service. The "Wireless Body Area Network" (WBAN) is a network of sensor and/or actuator nodes on, near or indeed inside the body that can communicate wirelessly with each other and/or with a Body Control Unit (BCU). The BCU is often referred to as a sink or gateway (as in this text).

2.3.1 Overview

The sensor nodes in a WBAN are used to read the physical characteristics of the person or the environment. For example a pulse oximeter shines light at different wavelengths through a patient's fingertip. The changes in absorbance due to the pulsing of the blood is detected by a photodetector.

An actuator in a WBAN is used to control some device such as an automatic insulin pump.

Sensors and actuators may comprise several components. A sensor typically comprises: sensor hardware (e.g. electrodes), a power source (e.g. a battery), a processor, memory and a transceiver for communication.

An actuator typically has actuator hardware (e.g. a relay or 5V contacts), a power source (e.g. battery), a processor, memory and a transceiver for communication.

A BCU will typically have a larger processor as it is likely to carry out more analysis and may have to act as a gateway between the WBAN and other systems.

A WBAN is similar to a Wireless Sensor Network (WSN) however the WBAN is further characterised as follows:

- Low transmission power to keep interference at a low level so as to reduce the impact on other devices
- High propagation loss due to proximity to the human body
- Very limited battery life of devices
- Mobility of the user: the body is mobile so nodes must be able to cope with movement
- Privacy of data: the nature of medical data is highly sensitive from both a privacy and a clinical position.
- Heterogeneity of data: data rates and accuracy requirements vary a lot between different types of application in a medical WBAN. Table 2 outlines the bandwidth required for some typical data types in this environment.

Application	Data rate	Bandwidth (Hz)	Accuracy (bits)
ECG (12 leads)	288 kbps	100–1000	12
ECG (6 leads)	71 kbps	100–500	12
EMG	320 kbps	0–10,000	16
EEG (12 leads)	43.2 kbps	0–150	12
Blood saturation	16 bps	0–1	8
Glucose monitoring	1600 bps	0–50	16
Temperature	120 bps	0–1	8
Motion sensor	35 kbps	0–500	12
Cochlear implant	100 kbps	–	–
Artificial retina	50-700 kbps	–	–
Audio	1 Mbps	–	–
Voice	50-100 kbps	–	–

Table 2 Data rates required for various medical applications (Latré et al. 2010)

2.3.2 IEEE WBAN specification - 802.15.6

In 2012 the IEEE standard on WBANs was published. This standard defines MAC and PHY protocols for devices in an area about the size of the human body. It makes up for some shortcomings in the current Personal Area Network (PAN) specification such as interference with medical devices and acceptable power levels at close proximity to human tissue.

2.3.3 Network Topology

A BAN comprises nodes and a single hub. In a one-hop star BAN, data is exchanged between nodes and the hub. In a two-hop extended star BAN, the hubs are also able to communicate via a relay-capable node.

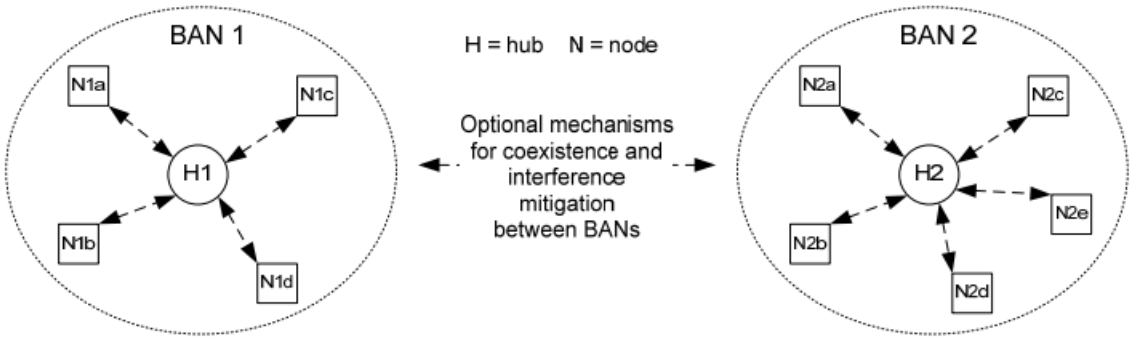

Figure 1 WBAN Network Topology (IEEE 2012)

2.3.4 Reference Model

Nodes and hubs are internally partitioned into a physical (PHY) layer and a medium access control (MAC) layer similar to the OSI model.

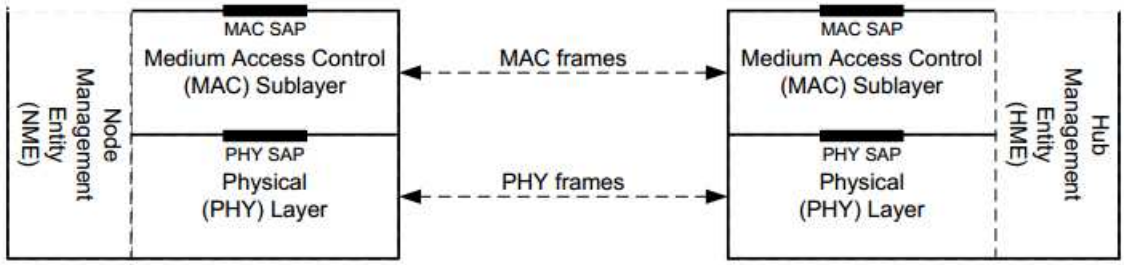

Figure 2 WBAN Reference Model (IEEE 2012)

2.3.4.1 Physical Layer (PHY) Summary

A standard for WBAN PHY is defined in IEEE standard 802.15.6 (IEEE 2012). Three physical specifications are included:

Narrowband PHY

The Narrowband PHY is defined for data rates of up to 1Mbps. A compliant device should be able to support transmission and reception in at least one of the following frequency bands: 402 MHz to 405 MHz, 420 MHz to 450 MHz, 863 MHz to 870 MHz, 902 MHz to 928 MHz, 950 MHz to 958 MHz, 2360 MHz to 2400 MHz, and 2400 MHz to 2483.5 MHz.

Note the use of unlicensed bands e.g. 2400MHz; the WBAN has been defined so that it can be used without license in public, and indeed, medical environments.

Ultra Wideband PHY

The Ultra Wideband PHY is designed to provide a large scope for implementation opportunities for high performance, robustness, low complexity, and ultra low power operation in BANs. It is defined for data rates of up to 12Mbps.

Human Body Communications (HBC) PHY

HBC communication is performed using the body as a conductor. Devices in a WBAN that use HBC do not need a wireless component. The user simply touches the devices, and the devices are connected to each other via touch-and-play (TAP) technology.

This form of communication should have useful applications for authentication or service discovery of WBAN nodes that are in low duty cycles.

2.3.5 MAC Layer Summary

The 802.15.6 standard specifies the requirements of the MAC layer including provisions for:

- medium access including management, control and data frames,
- an indication of capability of supporting different types of access e.g. CSMA/CA or ALOHA,
- security: message authentication and encryption, based on the Advanced Encryption Standard (AES) forward cipher function,
- clock synchronisation for medium access,
- creation of BANs by hubs,
- the connection of a node to a hub .

Clock synchronisation in a WBAN using MAC layer time-stamping was explored by (Maróti et al. 2004).

2.3.6 Network/Transport Layer

The use of Ethernet/TCP for a WBAN in a medical setting (hospital) was considered by (Khan, Yuce, and Karami 2008). They concluded that using TCP to transfer the large amount of small packets generated by medical devices over a wireless medium is inefficient. One approach to resolving this issue would be to combine packets of data before transmission. This approach was adopted in the work done in this dissertation.

Other studies have been conducted into routing in WSNs, one of the most famous being CodeBlue (Malan et al. 2004) - a WSN developed in Harvard. CodeBlue is an implementation of a scalable software infrastructure for wireless medical devices. CodeBlue is designed to provide routing, naming, discovery, and security for wireless medical sensors, PDAs, PCs, and other devices that may be used to monitor and treat patients in a range of medical settings. It uses Adaptive Demand-driven Multicast Routing as described in (Jetcheva and Johnson 2001).

2.3.7 Application Layer

(Chen and Shih 2012) describes an architecture for an electronic medical records (EMR) system which includes streaming media. It is an XML document-based system that is derived from the Taiwan Electronic Medical Record Template (TMT) standard. It describes an approach to include static and dynamic medical images into these documents using a Flash Media Server which uses RTMP. Most other network streaming formats use RTSP e.g. Windows Media Series, RealNetworks Realsystem and Apple QuickTime technologies.

Health Level-7 (HL7)

HL7 is a not-for-profit organisation that develops specifications; the most widely used being a messaging standard that enables disparate healthcare applications to exchange clinical and administrative data. The 7 in HL7 stands for layer 7 or application layer in the OSI model. HL7v3, the latest version, uses XML messaging as its foundation. A gateway between HL7v3 and CodeBlue has been successfully implemented by (Baird and Dawson-Haggerty 2006).

ECG Formats

In this dissertation a prototype using a single sensor ECG monitor is developed. For simplicity the sampled ECG data is encoded as a 16 bit int and no compression techniques are used to reduce the size of the captured data. There are however many formats that encode ECG data in more efficient ways. The reader is referred to (Bond et al. 2011) for a detailed survey of these.

2.4 Summary

In this chapter the state of the art in mHealth monitoring and Wireless Body Area Networks was evaluated. The Holter monitor was discussed and the growing need for health monitoring applications was highlighted. The architecture and components of a WBAN were outlined and some of the main characteristics of a WBAN were highlighted.

In the following chapter the focus moves to the more specific requirements of a suite of protocols that would support real-time monitoring of mHealth data from a WBAN.

Chapter 3 Design

In this chapter the specific requirements for a real-time medical WBAN monitoring system are considered. In particular, some existing transport protocols are evaluated as possible solutions for the application domain under consideration in this work.

3.1 Specific Requirements of a Medical WBAN to be used for Real-Time and On-Demand Monitoring

In this study the objective was to develop a framework for the distribution of real-time heterogeneous data collected in a WBAN. To that end the following considerations must be added to the previously outlined description of the WBAN:

Transfer/monitoring of data

It is desirable that the data captured in the WBAN may be viewed live or after the event (on-demand) from a remote site. Utilizing existing standardised transport protocols is an obvious way to achieve this. For example, using media transport protocol suites such as RTSP/RTP or HTTP based streaming protocols like HTTP Live Streaming (HLS) would allow the transfer and synchronisation of different streams of data. Furthermore, the use of a standardised protocol like the ones mentioned, instead of a new one, can bring with it additional benefits such as support by routers and firewalls on the internet.

Live/on demand requirement

Devices in a WBAN are typically low power and are not going to be constantly connected to each other or to the internet so there is a requirement for offline storage and playback or "on-demand" streaming of data.

Operational Modes/End-to-End Control

Again, in order to save power it may be desirable for the nodes in a WBAN to operate in different modes, for example accelerometer data may be captured at a low sample rate until some particular

event is detected on an ECG sensor. An end-to-end control mechanism may be needed to achieve this.

Synchronisation

Data collected from different sensors in the WBAN should be synchronised for playback either live or at a later date.

Recording

The system should facilitate recording of data either within the WBAN or to a remote site.

Contextualisation

Contextualisation of data collected would generally assist physicians, allowing them to provide a better service and, hence improving patient outcomes. Some examples of levels of collected patient data indicating that the patient is in an urgent state are shown in Table 3.

Acquired patient data	**Data levels indicating an urgent state**
ECG (Electrocardiogram, 3 leads)	ST Wave elevation & depression T-wave inversion
BP (Non invasive Blood Pressure)	90 mm Hg > systolic > 170 mm Hg
PR (Pulse Rate)	50/min > PR > 110/min
HR (Heart Rate)	50/min > HR > 110/min
SpO2 (Haemoglobin oxygen saturation)	< 90 (%)

Table 3 Patient data and data levels indicating an urgent status (Doukas, Maglogiannis, and Kormentzas 2006)

Security/Privacy

Medical data is highly sensitive and must not be made available to third parties without permission. There may be legal ramifications if this privacy is compromised. Any framework for transmission of this information needs to be resilient to attack and provide for the encryption of the data.

Reliability/Integrity

By its nature medical data that is to be used by physicians for diagnosis or monitoring in a clinical setting, or otherwise, must be delivered intact. Misdiagnosis based on an incorrect representation due to data corruption would be a failure of the system.

Low Overhead

Ideally any protocol used to transmit data in a WBAN should be capable of operating with minimal overhead. i.e. it should involve a simple setup mechanism and the use of minimal additional header data.

The use of a media transport protocol seems a good fit for the project given the requirements for real-time transmission/playback, synchronisation and recording identified as being necessary in a WBAN monitoring system. Moreover, a media transport protocol meets the requirements outlined in the previous chapter and builds on the work done in (Chen and Shih 2012).

3.1.1 Existing Protocols

The use of an existing protocol is preferred because it has already gone through extensive testing and is, or is at least expected to be, well supported in the wider world.

A good comparison of media control protocols can be found in (Ravesteijn 2009) however the focus of that study is mainly on the transfer of medical documents rather than on real-time data.

Some of the key existing standards for media transmission are summarised below:

3.1.1.1 HTTP Live Streaming (HLS)

HLS is a media streaming protocol implemented by Apple and proposed as a draft standard in (Pantos and May 2010).

The server breaks the media stream into a sequence of files which are downloaded using HTTP requests by the client which then reassembles them for playout. A playlist file is used to define the sub-streams that are available.

It has the benefit of using HTTP requests which will be automatically passed by any firewall or proxy that allows through standard HTTP traffic.

However it is only defined for MPEG-2 transport streams or MPEG-2 audio streams and the draft does not currently include a method for including other types of streams.

3.1.1.2 RTSP/RTP

RTSP is a control protocol for media streaming. It is similar to HTTP and uses requests and response codes. A typical transaction between a client C and a media server M is given in Figure 3.

The protocol is widely supported and is an open standard described in RFC 2326 (Schulzrinne, Rao, and Lanphier 1998).

Resources such as audio or video files or live streams are identified by URIs. Once a media session has been established through the use of the SETUP request, the client and server use the same session ID for all correspondence (until a pre-determined session expiration time).

The DESCRIBE method shown Figure 3, combined with the SETUP method allows a client to negotiate a session with a server. This feature would be useful if a server had, for example, media encoded at different bit rates, as it allows the client to choose which one is best suited for its application.

RTSP is not actually used to deliver the data; this is typically done using the transport protocol Real-time Transport Protocol (RTP) together with the Real-time Control Protocol (RTCP) described in RFC 3550 (Schulzrinne et al. 1996).

RTP and RTCP are used to actually packetize and stream the live, or on-demand media. The data is transmitted within the RTP packets while control data is sent in RTCP packets. The RTCP packets allow multiple streams to be synchronised at the receiving end. They also contain reports on packet loss and jitter, allowing the sender to modify its transmissions based on this feedback and to calculate the transmission delay.

```
C->M: DESCRIBE rtsp://foo/twister RTSP/1.0
      CSeq: 1
M->C: RTSP/1.0 200 OK CSeq: 1
      Content-Type: application/sdp Content-Length: 164
      v=0
      o=- 2890844256 2890842807 IN IP4 172.16.2.93 s=RTSP Session
      i=An Example of RTSP Session Usage
      a=control:rtsp://foo/twister t=0 0
      m=audio 0 RTP/AVP 0
      a=control:rtsp://foo/twister/audio m=video 0 RTP/AVP 26
      a=control:rtsp://foo/twister/video
C->M: SETUP rtsp://foo/twister/audio RTSP/1.0
      CSeq: 2
      Transport: RTP/AVP;unicast;client_port=8000-8001
M->C: RTSP/1.0 200 OK
      CSeq: 2
      Transport: RTP/AVP;unicast;client_port=8000-
      8001;server_port=9000-9001
      Session: 12345678
C->M: SETUP rtsp://foo/twister/video RTSP/1.0
      CSeq: 3
      Transport: RTP/AVP;unicast;client_port=8002-8003 Session:
      12345678
M->C: RTSP/1.0 200 OK
      CSeq: 3
      Transport: RTP/AVP;unicast;client_port=8002-8003;
      server_port=9004-9005
      Session: 12345678
C->M: PLAY rtsp://foo/twister RTSP/1.0
      CSeq: 4
      Range: npt=0- Session: 12345678
```

Figure 3 Typical RTSP interaction

RTP packets include a Payload Type (PT) field that identifies the format of the contained payload. For audiovisual presentations this is usually chosen from a pre-defined list of payload types managed by the IANA and originally defined in (Schulzrinne and Casner 2003). However, it could also be used to identify non-standard payloads such as live medical data. Receivers are designed to ignore payload types that they do not understand and so the data could easily be added to existing RTP streams without causing a problem for standard decoders.

Privacy is implemented in RTP through encryption of the RTP and RTCP packets. Authentication and data integrity are not implemented in the basic RTP standard but can be achieved either at a lower level using IPsec, as described in RFC (Kent and Atkinson 1998), or by using the related protocol SRTP as described in RFC 3711 (Mcgrew and Carrara 2004). RTP and RTCP typically run over UDP so if the integrity of the data is to be maintained, one of these techniques must be used to deal with lost or corrupted packets.

RTSP sends messages in plain text. There is no compact form for keywords, however RTP does include a mechanism for header compression as defined in RFC 2508 (Casner and Jacobson 1999).

3.1.1.3 SIP

Session Initiation Protocol (SIP) is a media transmission control protocol primarily used for setting up and managing voice and video calls over the internet. It is described in (Rosenberg, Schulzrinne, and Camarillo 2002)

It is similar to HTTP in that it uses request and response methods called transactions. Like RTSP, it does not transmit the media data, rather it is just used to create and manage the session. The data is typically transmitted using a transport protocol like RTP.

End-points are referred to as User Agents (UAs). Typically proxy servers are used to route SIP packets. A registration function is used so that UAs can register their current locations for use by proxy servers to route calls. This effectively allows a UA to roam. It is also possible to set up a session directly between two user agents, which is a useful feature for remote monitoring applications.

SIP uses SDP to describe the session type, or profile, and so different media types may be used in a session depending on the capabilities of the receiver. SDP is extensible, so it is possible to define non-standard media types in SDP e.g. ECG data.

3.1.1.4 H.323

H.323 is an ITU-T recommendation defined in (ITU-T 2009). It is an audio-visual transfer protocol designed to run over any packet network.

The streams are transferred using RTP and RTCP in a similar way to RTSP and SIP above.

H.323 uses a Gatekeeper. This third party to a call manages permissions such as who can and can't call another end system and whether the end system can answer the call. In this way the Gatekeeper manages the resources of the network.

The Gatekeeper is also used to interconnect different types of networks. It can provide address translation where different addressing schemes are used on the different networks and can provide translations between the different messages on each network type, both control and media messages.

3.1.2 Summary of Existing Control/Transport Protocols

	HLS	RTSP/RTP	SIP/RTP	H.323/RTP
Extensible	No - media files must be MPEG-2 transport streams or an MPEG-2 audio stream	Yes - Payload Type header and SDP both extensible	Yes - SDP extensible	Yes
Latency - time to start playback	Client must download playlist and delay also depends on segment's duration and bit rate	Fast - playback requires 3 messages between client and server: SETUP, 200 OK, PLAY.	Fast - playback requires 3 messages between client and server: INVITE, 200 OK, ACK	Complicated - many messages between gatekeepers/ gateways
Lightweight - Overhead	Yes. No headers etc	RTSP uses text for messages - not so efficient. RTP has a 12 byte header for data however header compression an option	SIP uses text for messages - not so efficient. Same as RTSP for RTP	Control messages in binary representation - efficient
Open Standard	Yes but closely linked with Apple	Yes	Yes	Yes
Source Origin Authentication /Privacy of Data	Uses HTTP features such as HTTPS	Yes via SRTP	Yes via SRTP	Yes via H.235
Integrity of Data	Over TCP so reliable	Yes via SRTP	Yes via SRTP	Yes via H.235
Ease of Support by Firewalls/proxies	Yes - HTTP ports usually open	RTSP OK as usually runs over TCP. NAT issues for RTP	SIP OK as usually runs over TCP. NAT issues for RTP	Not easy - firewalls must act as application level proxies
Synchronisation of multiple streams	Data must first be encoded as MPEG2	Yes	Yes	Yes
VCR Type Commands - Play, Seek etc	Yes	Yes	No	Yes

Table 4 Summary of existing control/transport protocols

In order to compare the different protocols outlined above, Table 4 is provided, summarising their relevant characteristics.

Analysis of the existing protocols led to the choice of RTSP with RTP for this dissertation. Out of the evaluated protocols, RTSP's relatively low overhead coupled with its VCR-type functionality appear to make it the most suitable for the control and management of a remote monitoring application. RTP's ability to synchronise multiple streams of data makes it suitable for the transport of the sensor data. A more detailed analysis of the features of RTSP and RTP, and how they suit this particular application is given in the following section.

3.2 RTSP as a Multi-Sensor WBAN Monitoring Control Protocol

3.2.1 Overview

Clients and Servers communicate in RTSP by way of Requests and Responses.

The Request

A Request message from a client to a server contains a Method and a Resource (URI) to which the method will be applied. This is shown in Figure 4, where C is client and M is media server.

```
C->M: DESCRIBE rtsp://foo/twister RTSP/1.0
      CSeq: 1
```

Figure 4 RTSP request message example

The defined methods are:

"DESCRIBE"

"ANNOUNCE"

"GET_PARAMETER"

"OPTIONS"

"PAUSE"

"PLAY"

"RECORD"

"REDIRECT"

"SETUP"

"SET_PARAMETER"

"TEARDOWN"

The Response

A Response is sent in response to a Request and always contains a Status Line which is made up of a Status Code and a textual phrase associated with the Status Code, see Figure 5.

```
C->M: SETUP rtsp://foo/twister/audio RTSP/1.0
      CSeq: 2
      Transport: RTP/AVP;unicast;client_port=8000-8001

M->C: RTSP/1.0 200 OK
      CSeq: 2
      Transport: RTP/AVP;unicast;client_port=8000-8001;
      server_port=9000-9001
      Session: 12345678
```

Figure 5 Example of RTSP request and response

Status codes are formatted similar to HTTP status codes and are classified as follows:

The first digit of the Status-Code defines the class of response. The last two digits do not have any categorisation role. There are 5 values for the first digit:

- 1xx: Informational — Request received, continuing process,
- 2xx: Success — The action was successfully received, understood, and accepted,
- 3xx: Redirection — Further action must be taken in order to complete the request,
- 4xx: Client Error — The request contains bad syntax or cannot be fulfilled,
- 5xx: Server Error — The server failed to fulfill an apparently valid request.

An RTSP server implements a simple to implement finite state machine. This is shown in Figure 6.

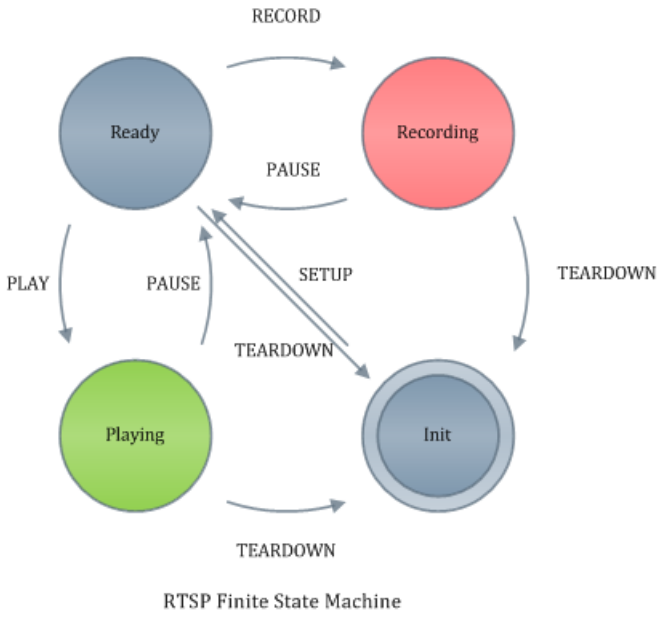

Figure 6 The RTSP Finite State Machine

3.2.2 Methods

In the RTSP standard there are many features that would be useful in a WBAN monitoring system.

Some of the main methods with respect to this requirement are outlined below.

OPTIONS

The OPTIONS method is sent from a client to a server. The server returns the methods that it supports. This is shown in Figure 7.

```
C->S: OPTIONS * RTSP/1.0
      CSeq: 1
      Require: implicit-play
      Proxy-Require: gzipped-messages
S->C: RTSP/1.0 200 OK
      CSeq: 1
      Public: DESCRIBE, SETUP, TEARDOWN, PLAY
```

Figure 7 Options request and response

This would be useful in an embedded environment because not all methods may need to be implemented. A client could tailor their requests with respect to how full an implementation of

RTSP is in place. For example, in the exchange above, there is no PAUSE method returned. This might be due to the fact that an embedded mHealth monitor only supports playback of live data.

DESCRIBE

The DESCRIBE method retrieves the description of a presentation, or a media object, identified by the request URL from a server.

```
C->S: DESCRIBE rtsp://server.example.com/fizzle/foo RTSP/1.0
      CSeq: 312
      Accept: application/sdp, application/rtsl, application/mheg
S->C: RTSP/1.0 200 OK
      CSeq: 312
      Date: 23 Jan 1997 15:35:06 GMT
      Content-Type: application/sdp
      Content-Length: 376

      v=0
      o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
      s=SDP Seminar
      i=A Seminar on the session description protocol
      u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
      e=mjh@isi.edu (Mark Handley)
      c=IN IP4 224.2.17.12/127
      t=2873397496 2873404696
      a=recvonly
      m=audio 3456 RTP/AVP 0
      m=video 2232 RTP/AVP 31
      m=whiteboard 32416 UDP WB
```

Figure 8 A DESCRIBE request/response

In the request shown in Figure 8 the client asks the server to DESCRIBE the resource rtsp://server.example.com/fizzle/foo. The (optional) Accept line of the request informs the server of the formats that the client can understand.

Again, in an mHealth context, this feature would allow different sensors or gateways to support different types of media, for example continuous ECG data, Blood Pressure or contextual information. A client can use the DESCRIBE method to ascertain which resources it wants to play,

then by only selecting the ones it is interested in, the sensors and gateway can save system resources, such as battery life, by not having to transmit unwanted data.

The information in the response after the Content-Length header is a Session Description Protocol (SDP) of the media identified by the URL in the DESCRIBE request.

A full description of SDP can be found in the RFC (Handley, Perkins, and Jacobson 2006). The elements that may be applicable to an mHealth monitoring application are described in Appendix A. Of particular interest is the use of dynamic payload definitions which allow a client to specify the details of the media in a dynamic way. This would allow nodes to advertise the fact that they are sampling data at a different rate or have disabled particular features. The client would then know what data or data format to expect based on these SDP descriptions.

SETUP

Once a client has decided what transport mechanism it can accept, for example after a DESCRIBE request, it can then issue a SETUP request. The SETUP request specifies which mechanism to use for the resource (URI).

By using the Transport header the client specifies all of the transport parameters that are acceptable to it, e.g. unicast/multicast, UDP/TCP, ports, etc.

A typical SETUP request from a client to a server is show in Figure 9 along with the "200 OK" response.

```
C->S: SETUP rtsp://example.com/foo/bar/baz.rm RTSP/1.0
      CSeq: 302
      Transport: RTP/AVP;unicast;client_port=4588-4589
S->C: RTSP/1.0 200 OK
      CSeq: 302
      Date: 23 Jan 1997 15:35:06 GMT
      Session: 47112344
      Transport: RTP/AVP;unicast;
      client_port=4588-4589;server_port=6256-6257
```

Figure 9 A SETUP request/response

The Session ID is created by the server on receipt of a SETUP request. From that point on all interactions from the client with respect to this session will be done using this ID.

If multiple streams are required, for example ECG and Accelerometer streams then a session will be created using a SETUP command for each stream. Each stream will be broadcasted via the transport parameters agreed in the respective SETUP requests.

PLAY

Once a SETUP request has been received and responded to a client may send a PLAY request to a server.

The PLAY request tells the server to start sending data using the transport details agreed in the SETUP request.

If present, the "Range" header specifies a section of the media file to play.

In Figure 10, the client requests the server to play from 10 seconds from the start of the file to 15 seconds from the start of the file.

```
C->S: PLAY rtsp://audio.example.com/audio RTSP/1.0
      CSeq: 835
      Session: 12345678
      Range: npt=10-15
      Scale: 2
```

Figure 10 Typical PLAY request from client to server

If the PLAY method is requested without the Range header from a live stream, the data is delivered live from the server.

In order to save battery life, and for mobility reasons, an mHealth monitoring application will not necessarily be connected to the internet all the time. Storage of data collected by nodes will be necessary and so the Range feature of the PLAY method would allow review of the media at specific points of interest.

The Scale header field can be used to change the rate and/or direction of playback of the media content. This header field would be of use when reviewing recorded content.

RECORD

The RECORD method allows a client to tell the server to record a session. If no Range header is provided the server will start recording immediately

ANNOUNCE

The ANNOUNCE method allows a server to update the SDP of a session in real-time. The newest SDP settings are used in place of the ones previously chosen in a SETUP request. This could be used as a way for embedded sensor nodes to update other devices. For example, if an embedded system wanted to change to a lower duty cycle (say with a lower sample rate) while it is being monitored by a monitoring application, it could send an ANNOUNCE request with its new settings to the monitoring application so an adjustment could be made for the new sample rate and playback could continue.

OTHER METHODS

PAUSE pauses playback of a session and TEARDOWN ends a session. Examples are given in (Schulzrinne, Rao, and Lanphier 1998).

GET_PARAMETER, SET_PARAMETER and REDIRECT are of no particular relevance to the work described in this dissertation but are required to provide a full implementation of RTSP.

3.2.3 The URL

Identification of resources in RTSP is by URL. With the right design this could have additional benefits in an mHealth WBAN. Apart from simply identifying the resources the format of the URL could be leveraged to add control features over and above the scope of RTSP.

For example the request:

```
SETUP rtsp://ecg.mywan.com/low_sample_rate RTSP/1.0
```

could be used to set up an RTSP connection to a node but also to tell the node to use a low sample rate when collecting the data.

3.3 RTSP 2

RTSP 2 seeks to solve some of the issues found in the first version of RTSP. A mechanism for NAT traversal is defined using the ICE protocol. Different transport options such as multiplexing RTSP and RTP streams and transporting RTP over TCP are defined. Transporting the data over TCP means that no data will be lost but may introduce additional challenges, e.g. due to TCP flow control or the loss of the ability to optimise the different streams of data from the different sources when multiplexing.

3.4 The Realtime Transport Protocol (RTP) as a Multi-Sensor WBAN Transport Protocol

Typically when RTSP is used as a session control protocol, RTP is used as the transport protocol to deliver the data. RTP is fully described in (Schulzrinne et al. 1996). An overview of the protocol and a description of some of the features that are of interest to a real-time mHealth monitoring system follows.

RTP is a text based protocol. RTP uses two packet types: an RTP data packet to send media data and a Real-time Control Packet (RTCP) control packet. Control packets provide for periodic reporting of reception quality, participant identification and other source description information as well as notification of changes in session membership and the information needed to synchronise media streams. Typically a server allocates 5% of the bandwidth to RTCP packets, although this can vary with different payload types.

3.4.1 The RTP Packet Header

The RTP packet header encapsulates the media data. Multiple samples can be included in a single RTP packet. In a session with multiple streams, each stream will send its own RTP and RTCP packets.

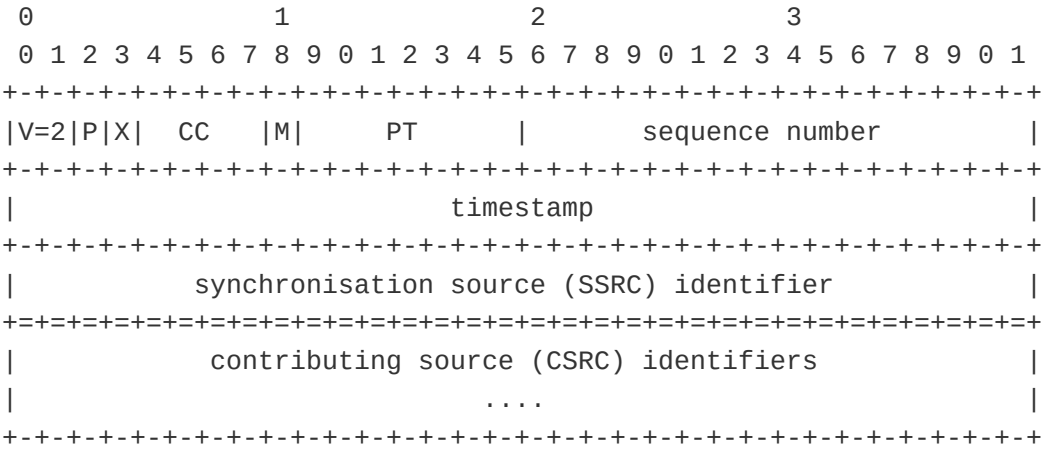

Figure 11 The RTP Header Format (Schulzrinne et al. 1996)

Some of the relevant fields from the packet are outlined below:

Payload Type (PT)

The Payload Type (PT) field identifies the type of media contained in the RTP packet. There is a table of standard mappings between the PT field defined in RFC 1890 (Schulzrinne and Casner 2003) but out of band signaling of the media format using RTSP can be done as described in the section on the SETUP request in 3.2.2 above and in the section on SDP in Appendix A. In the case of transmission of mHealth data, for example ECG data, this would be done using SDP.

Synchronisation Source (SSRC)

The synchronisation source (SSRC) identifies a stream within a session. All packets received must be grouped by SSC for playout.

RTP Sequence Number

The RTP sequence number is used to identify packets, and to provide an indication to the receiver if packets are being delivered in the wrong order or out of sequence.

RTP Timestamp

The RTP timestamp is a 32-bit unsigned integer that increases at a media dependent rate.

As an example audio payload formats typically use the sample rate as their media clock, so the timestamp typically increases by 1 per sample.

The timestamp is monotonically increasing and must form a continuous sequence even if the content of the packets is not a continuous stream. Figure 12 provides an example of this: Even though the two streams are from different sections of a media resource, the timestamp used in the RTP stream created continues to increase according to the media clock rate specified.

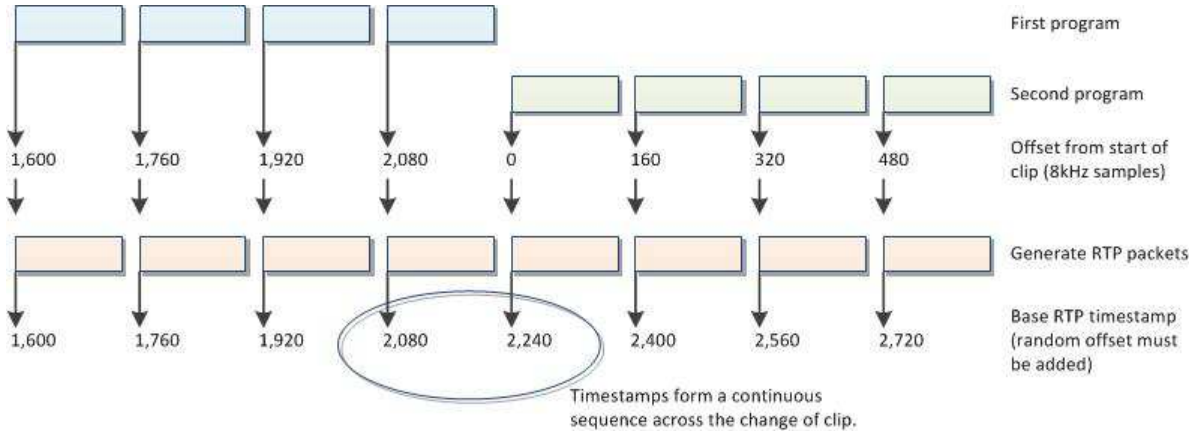

Figure 12 Continuous timestamp sequence across two spliced together clips (Reproduced from Perkins 2008)

Essentially this means that a player must generate the timestamp on the fly and, for example, an on-demand server could not save the RTP packets as they are captured complete with RTP timestamps for later transmission. If the server was requested to play the content in reverse the previously generated timestamps would no longer be increasing as required.

3.4.2 The RTCP Packet Header

RTCP is used to control the flow of data from the sender to the receiver. There are a number of types of RTCP packet that perform different roles in the control of the stream. The different types of RTCP packets are identified and summarised below. Figure 13 shows a typical RTCP SR (Sender

Report) type packet and the other types are similar. The payload type octet is used to identify the types e.g. "200" denotes a Sender Report as shown in Figure 13.

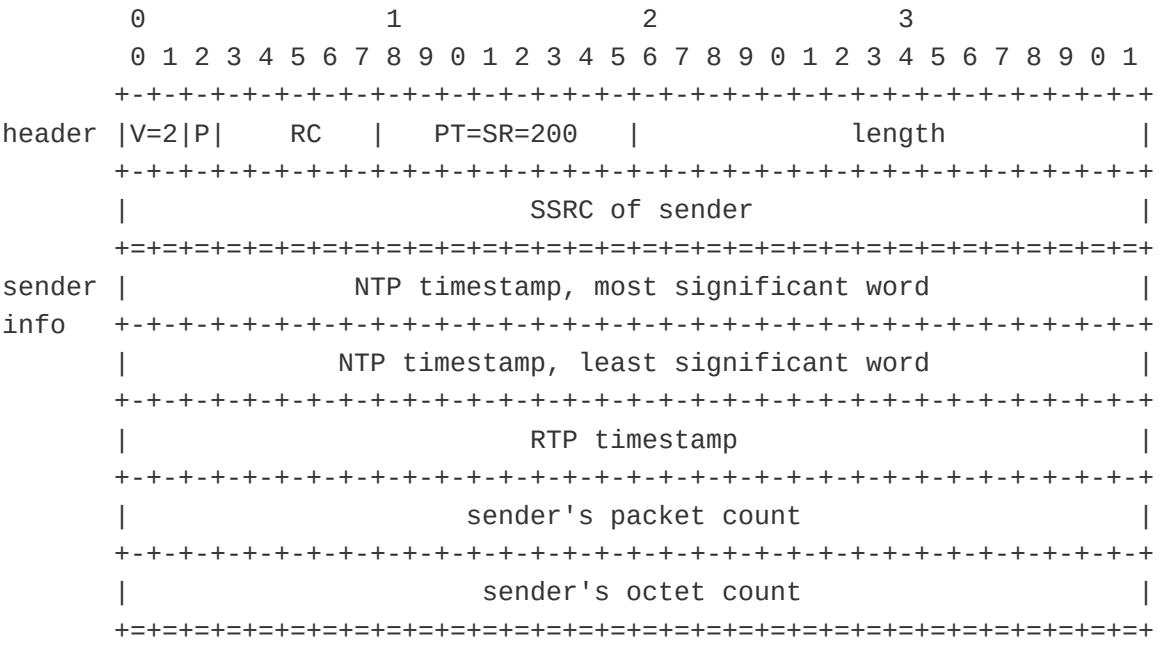

Figure 13 The RTCP Sender Report header format-(excluding reception report blocks) (Schulzrinne et al. 1996)

RTCP RR: Receiver Reports

Receiver Reports are sent from the client device to the server. RRs include information on the quality of reception of the packets. They include, amongst other details, the cumulative number of packets lost, the loss fraction, the inter-packet arrival jitter, the LSR (last Sender Report arrival time) and the DLSR (the delay since the last Sender Report arrived). The DLSR and the LSR allow the sender to compute the network round trip time between the sender and receiver which may be of use in optimising the data for transport. The packet loss, loss fraction and jitter values may be used by the sender to detect congestion and allow the sender to take remedial action such as reducing the data rate or changing to a more lossy media format.

RTCP SR: Sender Reports

Sender Reports are sent by participants that have recently sent data via RTP. SRs are primarily used for synchronisation of multiple streams. They include fields for NTP timestamp and RTP timestamp as shown in Figure 13.

The NTP timestamp is the time at which the RTCP SR packet was sent. It is a 64 bit unsigned value in the format of an NTP timestamp, counting seconds since 1/1/1900 in the upper 32 bits with the lower 32 bits representing fractions of a second.

The RTP timestamp represents the same moment as the NTP timestamp but is expressed in the units of the RTP media clock. This process is illustrated in Figure 14.

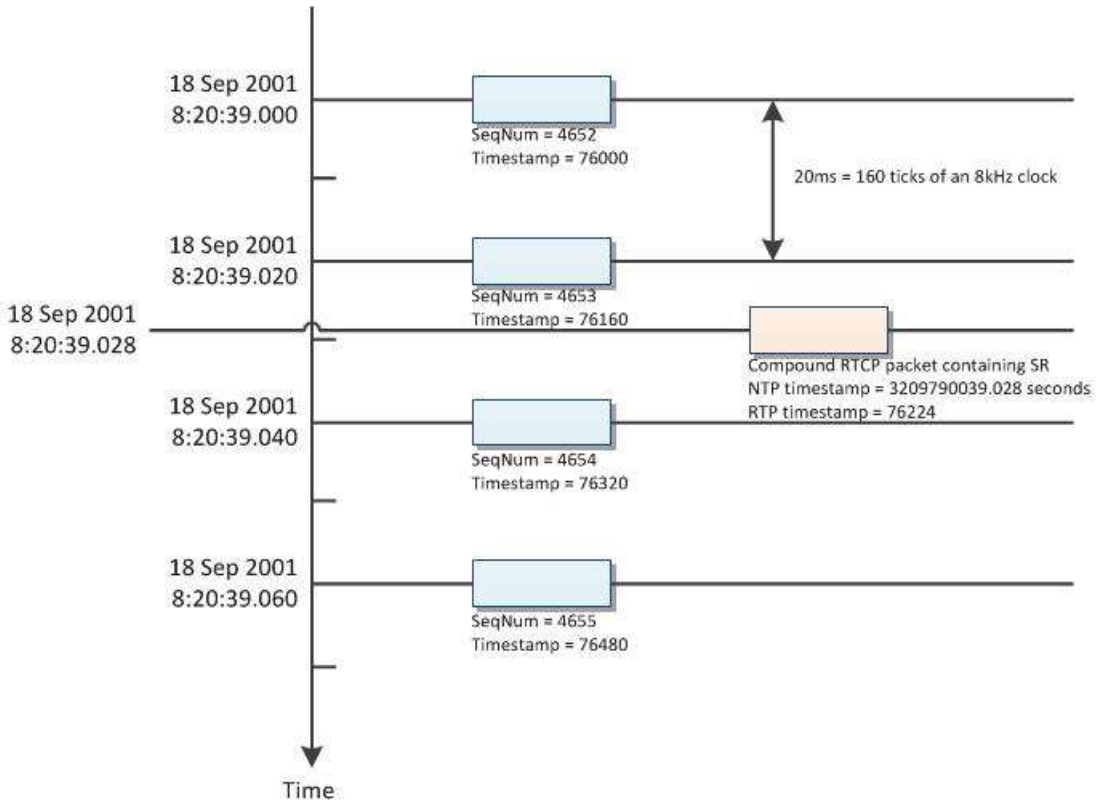

Figure 14 RTCP timestamp usage (reproduced from (Perkins 2008 page 109)

Because all media streams sent by a single sender will use a single "reference clock" to create the NTP timestamps in the SRs, a receiver can synchronise the RTP timestamps of the different streams to each other for playout. This is commonly used for lip synchronisation in video transmissions. Of course if two senders wish to synchronise streams they must synchronise their NTP time stamps. The Network Time Protocol would be one way to do this.

Clearly the lip-synchronisation aspects of RTP/RTCP are of interest when designing a protocol for use with multiple sources such as an mHealth WBAN. The process by which the receiver can

synchronise playout of multiple streams is illustrated in Figure 15. The use of RTCP in the implementation of the prototype is described further in 4.3.3.

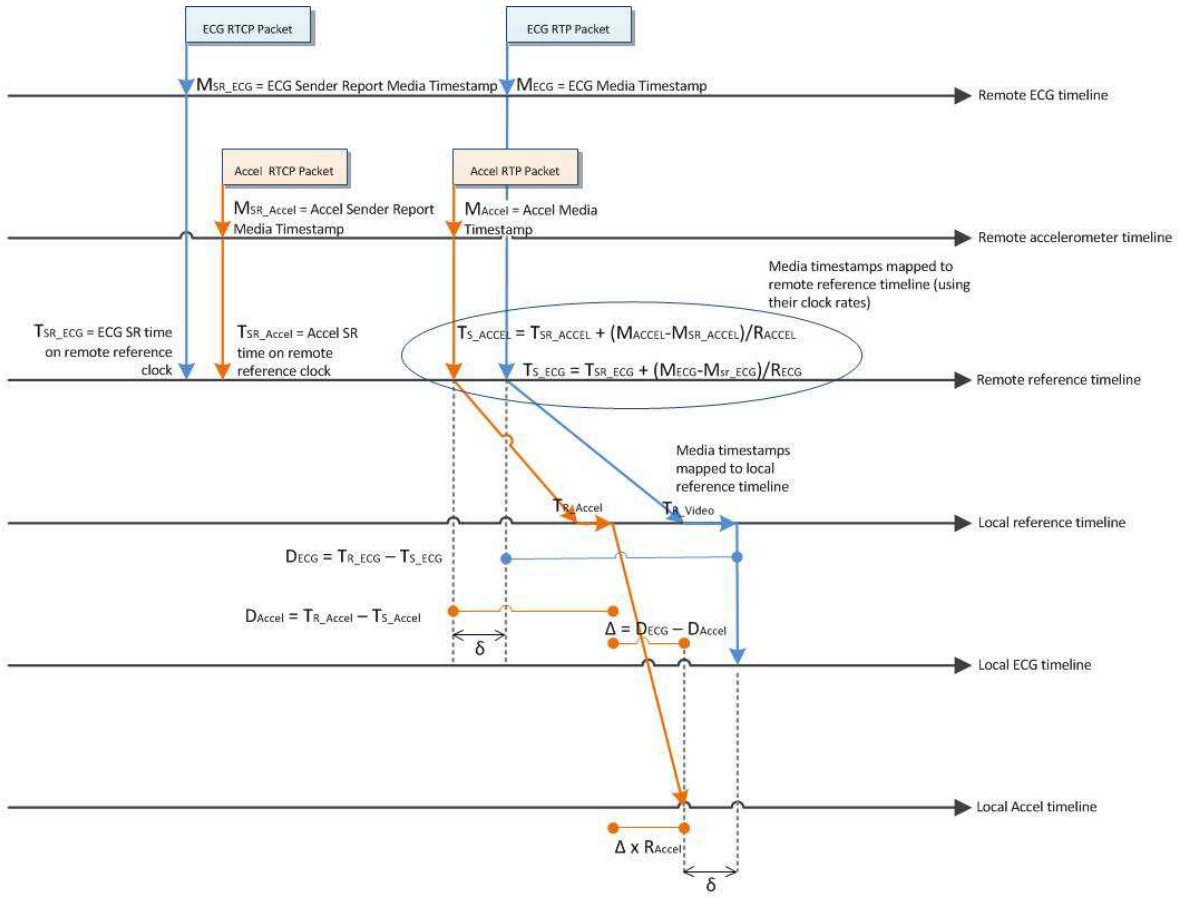

Figure 15 Mapping between timelines to achieve synchronisation at the receiver (adapted from (Perkins 2008)).

RTCP SDES: Source Description

In a conference environment the RTCP SDES packet sends information about the participant such as location, e-mail address, telephone number etc. While this is not particularly applicable to an mHealth WBAN the attributes could easily be overridden to provide other implementation details such as model or software version numbers of sensor nodes.

The SDES packet does however provide a mapping between the SSRC of a session and the CNAME.

In a presentation a sender can have multiple sessions with different synchronisation source (SSRC) numbers. If synchronisation of streams is required (such as between contextual and ECG data) then

the sender should set the canonical name (CNAME) of each of the synchronised sessions to be the same. The sender then lets any receivers know that these SSRCs are related to this particular CNAME using the RTCP sender description packets (SDES).

For example, if the sender is sending an ECG stream and an accelerometer stream via two RTP sessions that should be played "in sync", each session would have its own SSRC (a 32bit int) but they would share a CNAME e.g. a string like "mHEALTH_SENSOR_DATA".

3.4.3 Playout Delay

To compensate for clock skew, processing time on the player (and in the case of this dissertation processing time on the gateway), network jitter (packets arriving in the wrong order) and packet loss/recovery, playout of the media received in an RTP stream must be delayed.

In practice this is calculated by working out each component separately. Some strategies for approaching these calculations are fully described in (Perkins 2008). The clock skew and network jitter strategies are outlined below:

Adjusting For Relative Clock Skew Between Sender And Receiver.

Relative clock skew between a sender and a receiver must be detected and measured. The receiver can then compensate for the amount of skew in one of two ways. The first is to tune the receiver clock to match the sender clock and the second is to periodically adjust the playout buffer occupancy.

Tuning the receiver clock can result in very good results however this is not always possible due to hardware constraints. Adjusting the playout buffer can be done at the RTP layer and entails either adding empty samples to delay the playout buffer or removing samples to bring it forward.

Measuring the clock skew is done by comparing a weighted running average of the network transit delay with an active delay estimate. Increasing divergence between the active delay estimate and measured average delay denotes the presence of clock skew.

Adjusting For Network Jitter.

As for the clock skew, the jitter must first be measured before adjustments can be made. (Moon, Kurose, and Towsley 1998) suggest that the distribution of inter-packet arrival times versus frequency is similar to a Gaussian distribution. Choosing a playout delay equal to three times the standard deviation of this distribution would therefore cover over 99.5% of packets.

3.4.4 Header Compression

Most of the fields in the RTP header do not change or change in a constant way from one packet to the next. This makes it possible to define a header compression mechanism, Compressed RTP (CRTP) (Casner and Jacobson 1999). For example, from packet to packet, the Payload Type field does not change, while the first order difference in the RTP sequence number is constant meaning that the second order difference for the sequence number is zero.

The first packet sent is a full packet but subsequent packets are compressed and the link layer must send a message to the transport layer to route these packets to a decompression algorithm.

This mechanism is of benefit in an mHealth WBAN because much of the initial gathering of information will take place over lower bandwidth links such as Bluetooth or Zigbee.

3.4.5 Security

Authentication of the identity of the sender (source origin authentication), confirmation of the integrity of the data and maintaining the privacy of the transported data (encryption) are the three types of security that are required in this application.

Data integrity is not implemented in standard RTP. The Secure RTP (SRTP) standard defines data integrity authentication using shared keys. Source origin authentication is more difficult requiring the sender to be identified in the signature. This is defined for SRTP using TESLA (Carrara and Baugher 2005).

RTP defines a data encryption method using a shared key mechanism such as Advanced Encryption Standard (AES) but when used, the full RTP packet is encrypted and so RTP header compression will not function.

These security shortcomings of RTP can be addressed by implementing authentication and confidentiality at a lower level through the use of IP security extensions (IPsec) (Kent and Atkinson 1998).

Alternatively, SRTP can be used to provide confidentiality by encrypting the data using AES, data authentication using shared keys and source origin authentication using TESLA. SRTP is defined in RFC 3711 (Mcgrew and Carrara 2004) and a summary can be found in Appendix B.

RTSP does not have any security functionality built-in but if it were transported over TCP then, for example, TLS could be used.

3.5 Summary

The specific requirements of a real-time monitoring system have been set out above. Some of the existing standard real-time transport protocols were evaluated and RTSP together with RTP was proposed as a technology to support such a system.

In Chapter 4 the implementation details of a prototype built to test RTSP and RTP as a means to transmit and synchronise real-time mHealth data from a WBAN is described.

Chapter 4 Implementation

In order to test the efficacy of using RTSP/RTP as control and transport protocols for real-time mHealth data from a WBAN a prototype was designed and implemented. This chapter provides a detailed description of this implementation.

4.1 Topology

Three applications were developed. A Holter Device (HD), an ECG Gateway and a Remote Monitoring Application (RMA).

A visualisation of the topology of the system is provided in Figure 16.

An Arduino Uno was used to develop the HD. The ECG Gateway was developed as an app on an Android smart phone.

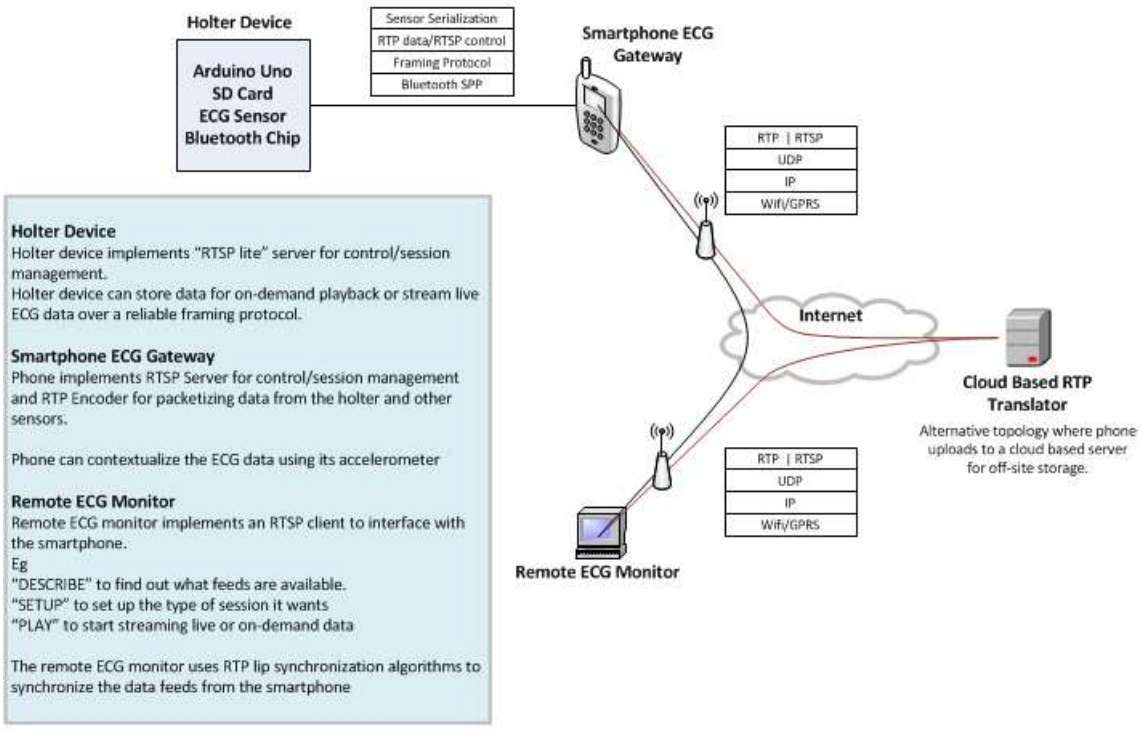

Figure 16 Prototype Topology

The Arduino HD communicates over a Bluetooth serial port (SPP protocol) with the smart phone ECG Gateway application. The ECG Gateway can packetize the data from the HD along with other sensor data, e.g. the accelerometer, and send it to the Remote Monitoring Application (RMA).

The RMA was developed in Java and run on a laptop. The RMA communicates with the ECG Gateway over an 802.11g Wi-Fi network using both RTSP and RTP protocols.

In a real world deployment the RMA would be installed on a physician's computer allowing them to view a patient's cardiac output in real-time. The application would connect to the smart phone over the internet. Potential transport issues with NAT/firewall traversal are discussed in Section 5.1.

4.2 Arduino Holter Device

4.2.1 Hardware Specification

The Holter device is built using the following hardware components:

- Arduino Uno R3 with 9V dc power supply
- Olimex ECG Shield with three sensors: left arm, right arm and leg for reference.
- RN42 Bluesmirf class 2 Bluetooth module running at 115200bps baud
- MicroSD Shield and 2GB micro-SD card for recording offline content

The Olimex ECG sensor samples data from three ECG sensors at 256Hz. The Arduino program is written in C++.

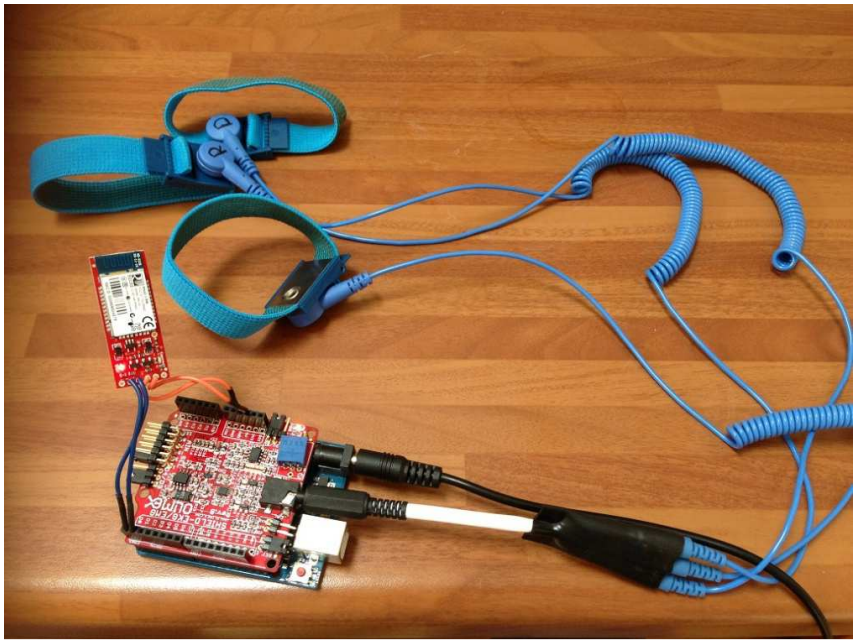

Figure 17 The Holter Device

4.2.2 Functional Description

Functionally the Holter device application is based on the RTSP finite state machine (see Figure 6). It implements an RTSP server for interaction with the ECG Gateway smart phone application and so responds to RTSP requests delivered over the Bluetooth serial port.

RTSP messages are delivered to the Holter device over the serial port using a framing protocol (see Section 4.2.3).

The RTSP PLAY and PAUSE requests were used to start and stop the packetizer. Selection between live and pre-recorded content was by choice of URL in the RTSP SETUP request. The SETUP request was required before any PLAY request could be sent. The server generates RTSP responses including "501" responses for unimplemented functionality and "400" responses for malformed requests.

The flow of data in the application is described in Figure 18, Figure 19 and Figure 20. A future enhancement would be the addition of the record feature. For the purposes of testing the playback of recorded content was simulated using generated data.

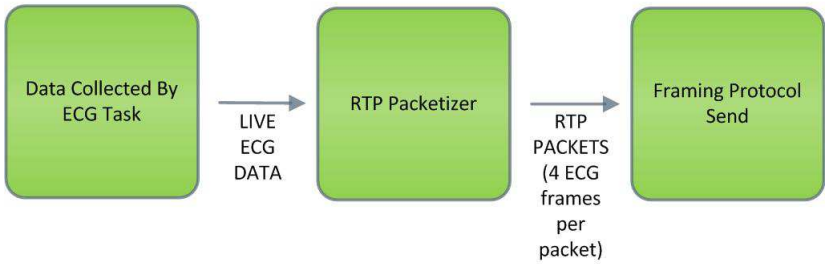

Figure 18 Holter Monitor playing captured data live while in the RTSP PLAY state

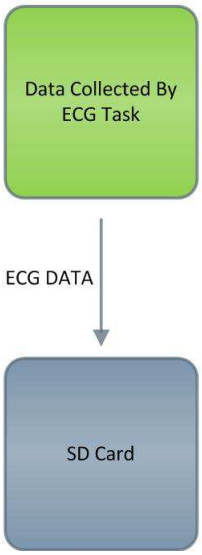

Figure 19 Holter Monitor capturing data to the SD card while in the RTSP RECORD state

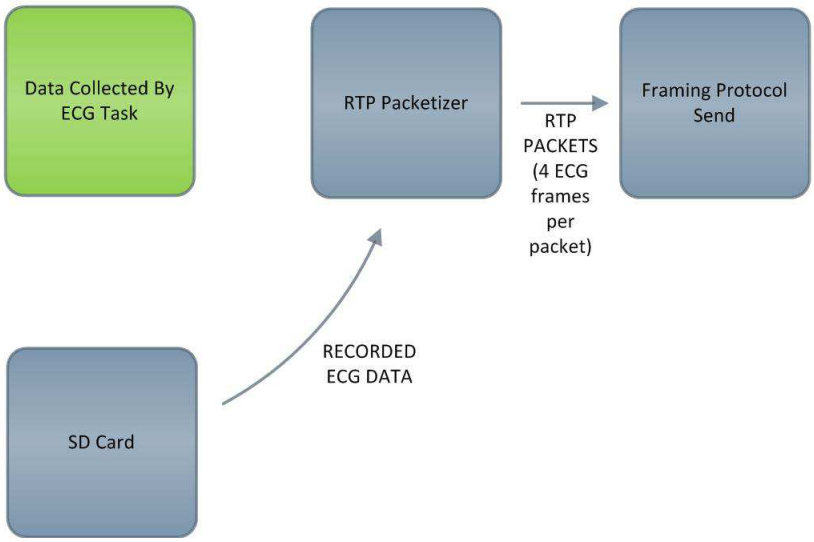

Figure 20 Holter Monitor playing saved data while in the RTSP PLAY state

4.2.3 Framing Protocol

In a typical RTSP/RTP server the application communicates with clients on multiple UDP ports. For example RTP and RTCP streams would be delivered over UDP on separate contiguous ports while RTSP control would communicate over yet another port over UDP, or possibly TCP. On the Holter device there is only a serial port to communicate with the Android ECG Gateway and so there was a requirement to multiplex the different streams of control and data information.

Additionally, given the sensitive the nature of medical data and the fact that this application is designed to be used for clinical diagnosis, there was a requirement for guaranteeing the integrity of the transmitted information in the presence of corrupted packets.

Finally it was considered critical that the control messages sent via RTSP were not lost. This necessitated the introduction of a mechanism for keeping track of lost packets.

In order to meet the above requirements a framing protocol was developed to sit between the RTP packetizer and the Bluetooth serial port of the Holter device.

The framing protocol uses a Type character to identify what type of payload in encapsulated (RTP, RTSP, message ACK etc) thereby meeting the multiplexing requirement. It uses a checksum algorithm to ensure data integrity. This was achieved using the CRC-16 algorithm. Lost frames are detected through the use of a sequence number. The framing protocol module for Arduino, and for Java on the phone side, can be used in a reliable or unreliable mode. In reliable mode lost and corrupted packets are resent. It might be desirable to use the unreliable version where reducing latency is to be prioritised over data integrity. The sequence number, type, data and CRC bytes are byte-stuffed to ensure any control characters appearing in the frame are not mistaken for actual control characters. Figure 21 shows the frame format of the designed framing protocol.

Figure 21 Framing Protocol Frame Format

4.2.4 Technical Challenges Faced

The protocol induced overhead is significant. If 4 frames of ECG data are packed into one frame the total number of bytes transmitted is 17 (with the potential for more bytes to be transmitted if characters are byte-stuffed). If an ECG sensor is sampling data at 256hz then this gives a total of 34,816 bps.

This is already well beyond the standard 9600bps bit rate of a serial connection so it was necessary to set the bit rate of the connection to 115200bps in order to cope with the amount of data coming from the Holter device. Looking at Table 2 in Chapter 2 it is clear that it may be necessary to encapsulate more frames of sampled data in each frame or to introduce some compression techniques for other data types if they are to be transmitted over the serial connection.

Another challenge encountered was the lack of UTF encoding on the serial connection from the Arduino. It was discovered after debugging that the Android serial port was expecting UTF-8 encoded serial data and so a small algorithm had to be introduced on the Holter device to deal with this.

4.2.5 On-Demand Content - Recording

It is desirable to provision for the fact that the Holter monitor will not be connected to the ECG gateway at all times. Reasons for this may include the smart phone being out of range, the preservation of battery life on the Holter monitor and the phone and also because there is no need to transmit data that is not being monitored. It is therefore necessary that the output from the ECG sensor may be recorded to some offline storage for review at a later date.

The prototype design meets this requirement by saving live data to the SD Card when in the RTSP RECORD state. The data is recorded to files created hourly in the format YYYY-MM-DD-HH. For example if the monitor starts recording at 12:05 on the 15th August 2013 the file will be called 2013-08-15-12. If the recording rolls into the next hour a new file is created 2013-08-15-13 and the data continues to be written in this file. The recording frame format is given in Figure 22.

```
DLE | STX | TimeStamp | Data | DLE | ETX
```

Figure 22 Recording Frame Format

An index is kept of all the recorded content available in the existing files. This can be used to generate a playlist for distribution to the smart phone application.

While playlist distribution is outside of the scope of RTSP in this implementation, a playlist can be included in a response to an RTSP describe request for a known URL, e.g. rtsp:ecg.example.com/playlist.

Parts of the recorded content can then be accessed using the range header of the RTSP PLAY request. This is given in Figure 23.

```
C->S: PLAY rtsp://ecg.example.com/ecg RTSP/1.0
      CSeq: 835
      Session: 12345678
      Range: clock=20131108T142300Z-20131108T143520Z
```

Figure 23 RTSP command to play previously recorded content

If a client requests unavailable content the Holter monitor returns a "`457 Invalid Range`" response.

4.3 Smart Phone ECG Gateway

4.3.1 Hardware/Software Specification

The smart phone application, show in Figure 24, is an Android App that runs on a Samsung Galaxy S2 mobile phone running Android version 4.2 Jelly Bean. It was developed in Eclipse in Java using Google's Android API level 16. It uses the built-in accelerometer and categorises vertical and horizontal movement as a proxy for contextual information.

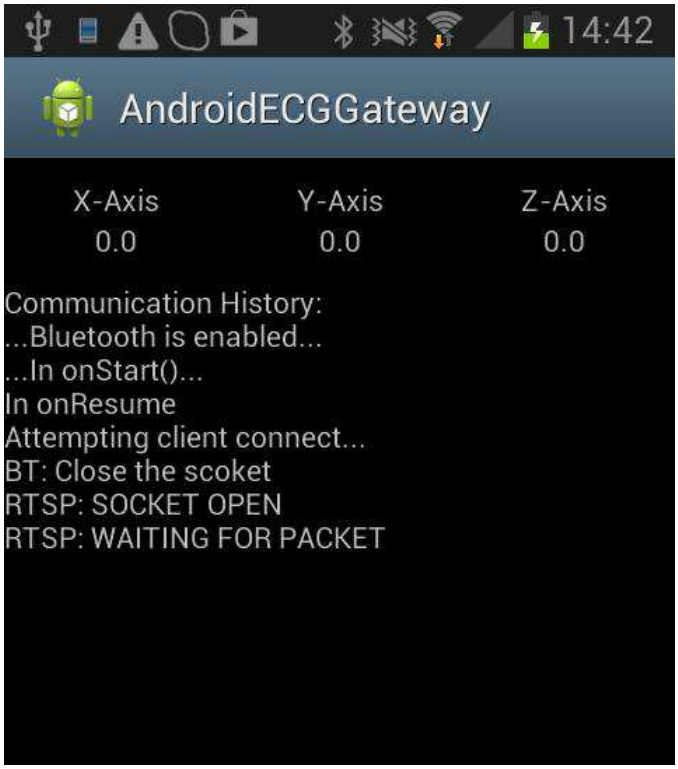

Figure 24 ECG gateway running on Android smart phone

4.3.2 Functional Description

The ECG Gateway application has to act both as an RTSP client to control the Holter monitor and as an RTSP server to send data to the remote monitoring application.

When interacting with the Holter monitor, all the RTSP control messages and data to and from the Holter monitor must be passed through the framing and de-framing algorithm, see section 4.2.3 above.

The gateway interacts with the remote monitoring application through an implementation of an RTSP server.

Classes are defined for the RTSP session, RTP session and the different requests, responses, packets etc. Figure 25 provides a (partial) illustration of these.

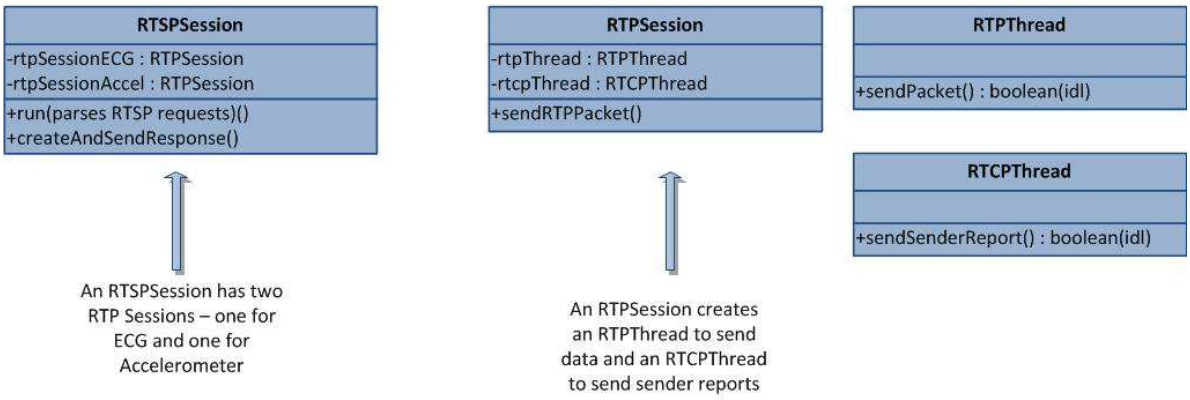

Figure 25 Android ECG Gateway RTSP Server Classes (only some of those involved with RTSP interaction with remote monitoring application shown)

When a SETUP request is sent from the remote monitoring application, the gateway sets up an RTP session to send data. Either a session for ECG data or accelerometer data is set up depending on the URL in the request. If both are required by the remote monitoring application it must send a SETUP request for each stream. This sequence of messages sent from the remote monitoring application through the gateway to the Holter device and back is shown in Figure 26. This demonstrates the end-to-end control capabilities of the prototype.

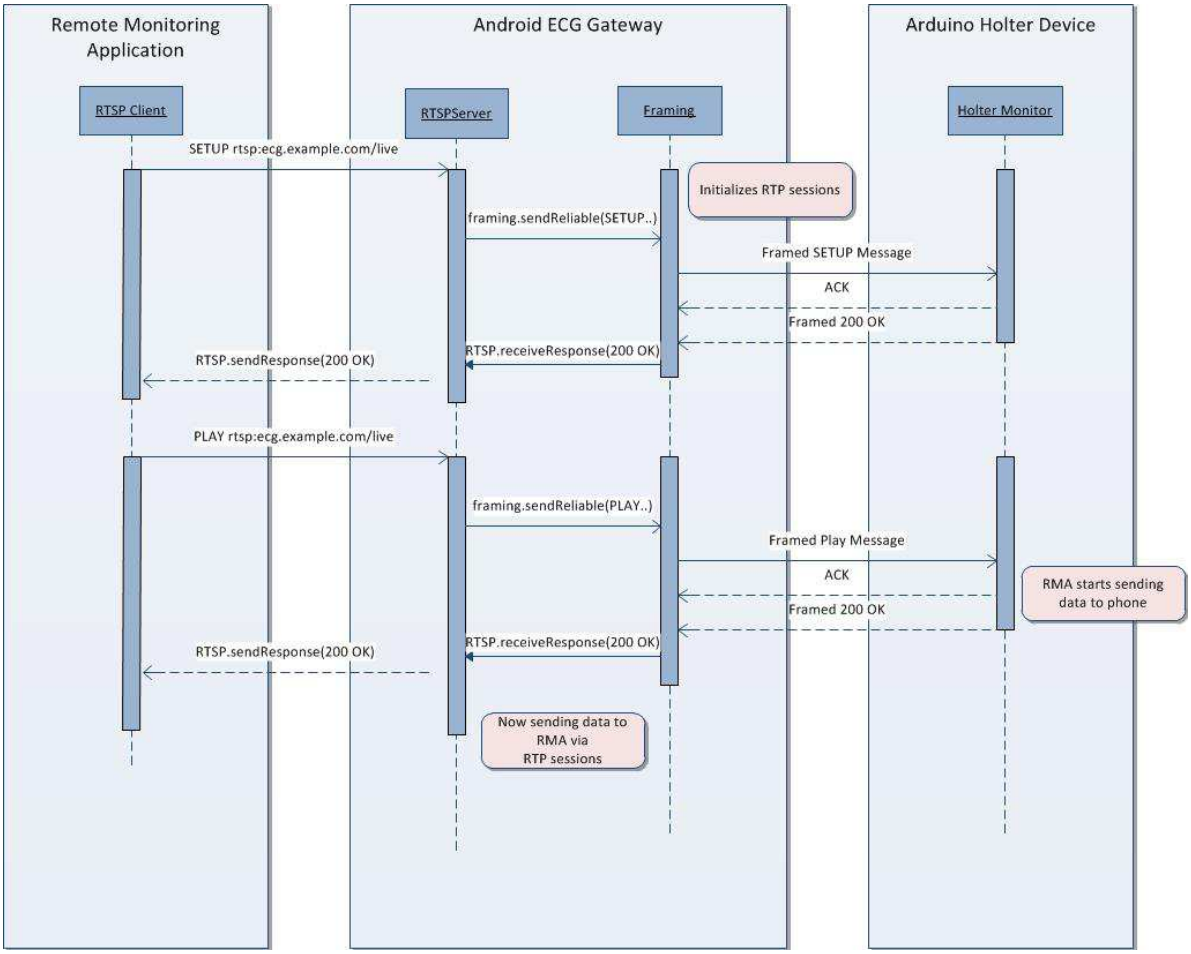

Figure 26 Session setup and play on ECG Gateway and Holter monitor using RTSP

4.3.3 Synchronisation

The RTCP Sender Report (SR) packets are periodically sent from the gateway application to the remote monitoring application for each RTP session to enable synchronisation of the ECG and accelerometer streams. The NTP timestamp of the SR packet is set to the current time of the ECG Gateway. The RTP timestamp in the SR packet is set to the same time but with reference to the RTP clock of the stream as shown in Figure 27.

```
now = System.currentTimeMillis();
SR.rtpTimestamp =
  sampleCount*(1000/sampleRate)+(now-lastSampleTime)*(sampleRate/1000);
```

Figure 27 Setting the Sender Report timestamp

This allows the Remote Monitoring Application to work out the relationship between the RTP timestamp sent with the RTP packets and the reference clock of the ECG gateway for each RTP stream.

4.4 Remote Monitor

4.4.1 Hardware/Software Specification

The Remote Monitoring Application (RMA) was built using Java Swing in Java SE 7.

4.4.2 Functional Description

The RMA comprises an RTSP client, RTP clients for ECG and Accelerometer data and an additional client for simulated/pre-recorded sine wave content from the Holter device for testing, and a user interface that includes a playout window.

Figure 28 The Remote Monitoring Application showing captured ECG and accelerometer data

Figure 28 shows the RMA user interface. The SETUP, DESCRIBE, PLAY, PAUSE and TEARDOWN buttons are used to send these RTSP requests to the ECG Gateway. Samples extracted from the received RTP packets are displayed in the black playout area. Samples received

in the previous five seconds, i.e. after the playout delay, are shown in the window with the newest samples appearing from the right hand side.

In Figure 28 the waveform shown is a sample being generated on the ECG gateway and sent via an RTP stream to the RMA. A second RTP stream is being sent with values from the accelerometer on the phone. Vertical bars represent vertical movement on the phone and horizontal bars represent horizontal motion on the phone. The RMA is able to play these streams out in sync using the information in the timestamp of the RTP packets and the NTP timestamp in the RTCP sender reports of each stream. This demonstrates use of the synchronisation features built into RTP.

4.5 Summary

In this chapter an implementation of a real-time mHealth monitoring system for use with WBANs was described. RTSP and RTP were used as control and transport protocols respectively. The three programs were described at a high level and some of the challenges overcome during the implementation were highlighted.

The next chapter evaluates this prototype and its design with respect to the specific requirements of a real-time mHealth WBAN identified in Section 3.1.

Chapter 5 Evaluation/Discussion

5.1 Evaluation/Discussion

The main objective of this dissertation was to investigate the use of a media transport protocol as a means of distributing mHealth data. The working prototype built and described in Chapter 4 was implemented as a proof of concept of this goal. This implementation is now evaluated using the specific requirements of a protocol for an mHealth real-time WBAN monitoring system as stated in Section 3.1.

Transfer/monitoring of data

The implementation showed that RTSP and RTP work well for the transport of the different live streams of data from the embedded Holter device through to the remote monitoring application.

One could imagine a situation where a physician might need to view their patient's health sensor output very quickly. Latency is inherent in the design and is a combination of the playout delay described in Section 3.4.3 and the setup time of the RTSP session.

A theoretical minimum for the setup time can be established for the prototype. If it is assumed that the system is running over UDP and if time expended on DNS lookups etc. is ignored, the minimum comprises: two round trips from the Remote Monitoring Application to the Android ECG Gateway (SETUP, 200 OK response and PLAY, 200 OK + Data) plus two round trips from the Android ECG Gateway to the Holter device (SETUP, 200 OK response and PLAY, 200 OK + Data) plus any time taken to process the requests and create the responses on the three programs. This exchange is illustrated in Figure 26.

In tests, the time measured between sending a play request and reception of the first RTP packet was, on average, 1100ms. An improvement could be achieved by increasing the polling rate of the RTSP server on the Holter monitor or by moving to an interrupt-based implementation.

Jitter was measured by calculating a running standard deviation of the inter-packet arrival time. It was measured at anywhere between 11ms and 70ms, averaging at 20ms. This measurement takes into account differences in transmission time due to network jitter but also, and more likely in this case, different processing times on the gateway and Holter monitor. This suggests setting the jitter component of the playout delay to 60ms based on (Moon, Kurose, and Towsley 1998).

The playout delay used in the application was 100ms. This worked well in practice and no dropouts due to a short playout delay were noticed. It would be interesting to explore the impact of simulated network jitter or packet drops in further work.

Data can be played as soon as it starts arriving but if multiple sessions are to be synchronised this will not start until the first RTCP packets arrive from each stream. This issue is a design feature of RTP and could be resolved by sending an RTCP packet as soon as an RTP session starts streaming.

Another potential issue with RTSP/RTP is NAT traversal. A transport protocol like HLS has the advantage of using port 80 which is almost always passed by firewalls. However RTSP and RTP can use any unspecified port so either firewalls in deployments must be aware of this requirement or a mechanism needs to be employed to solve the issue. RTSP does not define extensions for NAT traversal but this is addressed in RTSP 2 as it uses the ICE protocol described in (Rosenberg 2010).

Live/on demand requirement

The prototype demonstrated the use of the RTSP/RTP protocols for playback of live data from the sensor in the WBAN. A design was proposed in Section 4.2.5 for the play back of recorded content from the Holter device. Playback of different resources using different URLs in the SETUP and PLAY requests was implemented and could be used to play content from a list of URLs provided out-of-band. A recorded ECG waveform sample was played back from the Holter device using this feature.

This requirement for out-of-band transmission of the playlist is out of the scope of this dissertation but it would be possible to transmit something like an M3U playlist as part of a response to a DESCRIBE request to meet this requirement.

It might also be worth considering persisting data to the gateway when possible to allow the content to be reviewed without a connection to the Holter device.

Operational Modes/End-to-End Control

Again, the URL in RTSP has shown itself to be a very flexible tool and the server implementation on the Holter device could easily be modified to change the operational mode, for example changing the sampling rate or switching on/off features, such as header compression, where appropriate in order to make savings on battery life, latency etc. Furthermore the use of the Session Description Protocol allows the nodes in the WBAN to change their settings and to dynamically update the sink or the monitoring application via ANNOUNCE requests. This flexibility is very desirable in an mHealth WBAN because it would allow nodes to react to situations such as a low-battery level or high levels of radio interference.

Synchronisation

Synchronisation of two streams of mHealth data from a WBAN using the RTP protocol was demonstrated in the prototype (see Figure 28). In order to test this further a sine wave was generated on the Holter monitor and streamed to the RMA. It is played out on the RMA along with a locally generated sine wave.

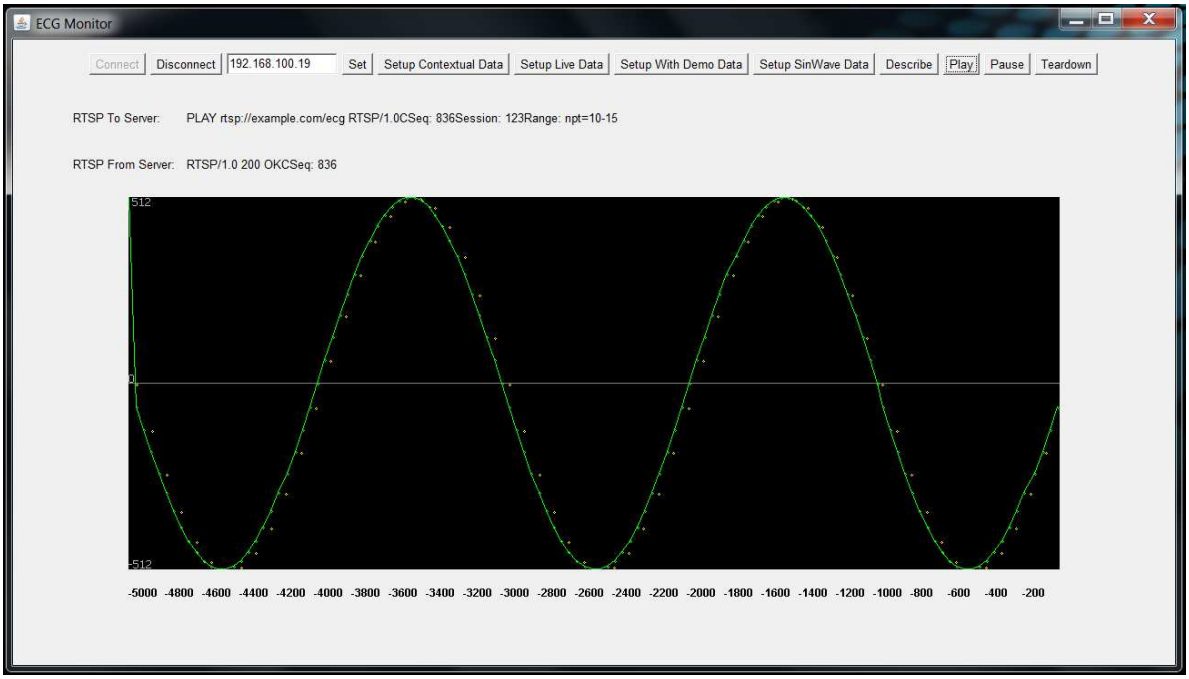

Figure 29 Sin wave streamed from Holter (green) compared with locally generated sine wave (yellow dots) after 30 mins playback

The system was left running for 30 mins and was shown to be running in sync; see Figure 29.

A question that arose during the demonstration was whether RTSP and RTP could be used to synchronise the actual hardware clocks themselves in a WBAN. While it would be possible to use the skew calculations available in RTP to work out drift in the clocks of devices this method is not distributed and would require a central point of control, with the phone being the most likely candidate for this role. A way to synchronise the data coming from devices exhibiting drift was outlined in Section 3.4.3. Alternatively there are other protocols specifically designed to synchronise the clocks in a resource limited wireless network, such as the Flooding Time Synchronisation Protocol (FTSP). The FTSP is described in (Maróti et al. 2004).

Recording

Recording is a native request method in RTSP. Clearly it is a suitable protocol to initiate recording and the headers can be used to specify whether to start recording immediately, at a specified time or periodically. In Section 4.2.5 a design for recording mHealth data in a WBAN for on-demand review was given.

If recording of multiple streams of data on different devices for future on-demand playback was required then the clocks on the different devices would need to be synchronised in order to ensure that the timestamps saved in the stored content were consistent. Again, the methods described above could be used.

Contextualisation

Contextualisation was achieved in the prototype through the use of a separate RTP stream for the categorised accelerometer data. The monitoring application used a very simple vertical and horizontal bar to illustrate the vertical and horizontal movement of the phone. The vertical and horizontal bars were displayed in the playout window synchronised with the captured ECG data.

Further work could be done to categorise the motion of the phone in a more meaningful way and other sources of contextual data could be added such as GPS coordinates. This raises the issue of overhead as the amount of data is relatively small compared to the RTP/UDP/IP headers. Possible solutions include header compression or bundling the data.

Security/Privacy and Reliability/Integrity

Security, privacy and data integrity will be considered together first for the Bluetooth connection in the WBAN and then for the connection from the gateway to the remote monitoring application over an IP network.

The prototype did not implement any security features however potential techniques for securing the privacy and integrity of the data as well as authentication of the sender and receiver were discussed in the design in Section 3.4.5.

The integrity of the data sent across the Bluetooth connection from the Holter device to the Android ECG gateway was ensured using the framing protocol designed and outlined in Section 4.2.3. The protocol implemented a sequence number to track lost packets and a checksum to ensure the validity of the data. Security was not considered in the design, however there are security features built in to Bluetooth which can be used e.g., 128bit encryption of the data using the e0 cipher. Other

precautions that can be taken include only allowing pairing with certain devices and only using long randomly generated passkeys e.g., not using "0000". More techniques for security and privacy over Bluetooth links for users and developers are outlined in (NSA United States).

In order to provide security across the IP portion of the application for the sensor data, SRTP and IPsec were advanced as possible design solutions. The advantages and disadvantages of these methods are summarised in Table 5.

IPsec		**SRTP**	
Advantages	Disadvantages	Advantages	Disadvantages
Confidentiality, integrity and authentication supported	RTP header compression not possible in IPsec transport mode. (possible in tunnel mode)	Confidentiality and data integrity supported	RTP header is not encrypted so traffic analysis possible
	Many firewalls block unrecognised traffic - IPsec hides the UDP header so the traffic may be blocked	Protects against replay attacks	In a large multicast group managing the keys is resource intensive. Not suitable for mobile devices with limited resources. (however multicast is not part of the spec of this project)
	NAT impossible if UDP port numbers encrypted	Out of order packets can be decrypted	No source origin authentication
		Header compression possible	

Table 5 Security and data integrity advantages and disadvantages of IPSec and SRTP

Low Overhead

In this prototype only 4 frames of ECG data were bundled into each RTP packet. This is 8 octets of data being packed in 12 octets of RTP data plus the 28 octet UDP/IP header.

Clearly this is a relatively large overhead. Bundling more frames of data into each packet is one solution but, of course, this will mean further delay at the playout point. In a typical audio application 200ms of audio are bundled into an RTP packet. So if 200ms of ECG data were encapsulated into an RTP packet the ratio of data to overhead would be roughly 100 bytes data:40 bytes overhead; plus marginally more overhead from RTCP. Additional overhead would accrue from the need to maintain the privacy of the data in RTP in the form of padding bits and maintaining a cryptographic context. Data integrity and source authentication in SRTP also add authentication data to each packet.

It should be mentioned that this is not unique to RTSP/RTP, as all the other protocols considered suffer from similar issues.

5.2 Evaluation

An evaluation of the prototype against the given requirements is provided in Table 6.

Requirement	**Evaluation**
Transfer/monitoring of data	Worked well for transfer of ECG and contextual data
Live/on demand requirement	Live and record features built-in to RTSP and RTP suitable for playback of live or previously recorded content
Operational Modes/End-to-End Control	RTSP URL fulfils this role
Synchronisation	RTP/RTCP synchronisation provide this functionality
Recording	RTSP RECORD method implements this
Contextualisation	Accelerometer on smart phone
Security/Privacy	Confidentiality built-in to RTP but can also use SRTP and IPsec. IPsec required for sender authentication
Reliability/Integrity	SRTP required for data integrity over network. Framing protocol designed for WBAN part
Low Overhead	Moderate to high overhead - 100:40 byte ratio for ECG data in 200ms chunks. Header compression can be used to alleviate if using SRTP.

Table 6 Evaluation of design/implementation based on identified requirements of a real-time mHealth monitoring system

Chapter 6 Conclusions

The RTSP and RTP protocol suite are a good fit for an mHealth monitoring system. The features and design of the protocols meet many of the requirements stated in Section 3.1. Synchronisation of streams is achieved along with real-time monitoring over the network.

Identified issues such as security, authentication and data integrity can be resolved by using SRTP or IPsec. Further work would be needed to investigate the suitability and performance of these protocols in an embedded mobile health environment.

NAT and firewall traversal remain an issue and it would be instructive to test the prototype on the internet to evaluate the magnitude of this problem.

There is an explosion of biometric monitoring devices appearing in the marketplace. Healthcare costs are increasing worldwide and people's awareness of their own health data is on the rise. In this context, demand for mHealth information systems, such as the one prototyped in this dissertation, is expected to grow rapidly. RTSP and RTP prove themselves worthy of consideration for any developer attempting to address these concerns through the use of real-time monitoring applications.

Appendix A. Session Description Protocol (SDP)

The SDP attributes of use in an mHealth application are described below with regard to the RTSP DESCRIBE request/response exchange shown in Figure 30.

```
C->S: DESCRIBE rtsp://server.example.com/fizzle/foo RTSP/1.0
      CSeq: 312
      Accept: application/sdp, application/rtsl, application/mheg
S->C: RTSP/1.0 200 OK
      CSeq: 312
      Date: 23 Jan 1997 15:35:06 GMT
      Content-Type: application/sdp
      Content-Length: 376

      v=0
      o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
      s=SDP Seminar
      i=A Seminar on the session description protocol
      u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
      e=mjh@isi.edu (Mark Handley)
      c=IN IP4 224.2.17.12/127
      t=2873397496 2873404696
      a=recvonly
      m=audio 3456 RTP/AVP 0
      m=video 2232 RTP/AVP 31
      m=whiteboard 32416 UDP WB
```

Figure 30 DESCRIBE request/response exchange showing SDP description

Connection Data ("c=")

If the session is multicast, the connection address will be an IP multicast group address. In the context of this dissertation, the connection address contains the unicast IP address of the expected data source or data sink.

It is possible to create a hierarchical encoding scheme where the encoding from a single media source is split into a number of layers. The receiver can choose the desired quality (and therefore bandwidth) by only subscribing to a subset of these layers. Each layer is transmitted in its own multicast group where each group has an address contiguously increasing from the base address given in the Connection Data header e.g.

`c=IN IP4 224.2.1.1/127/3` states that addresses 224.2.1.1, 224.2.1.2, and 224.2.1.3 are to be used. Each group would have a different quality and a subscriber could just join the one at the quality level they require thereby reducing network traffic from the gateway. This would be desirable on a lower bandwidth uplink such as 3G.

Media Descriptions (`"m="`)

`m=<media> <port> <proto> <fmt>`

Single or multiple media descriptions can be included in an SDP to describe each media stream available in a session.

Each description comprises:

`<media>` The type of media e.g., video, audio, etc. For the purposes of this work, we could define health as a media type.

`<port>` This is the port to which the data will be sent and depends on the transport protocol used. For RTP data it is usually an even port number and the RTCP data is assumed to be sent on the next contiguous port.

`<proto>` The transport protocol e.g. RTP/AVP denotes RTP Profile for Audio and Video Conferences with Minimal Control over UDP.

`<fmt>` The format of the media. If RTP/AVP is the protocol type above then this is the Payload Type number of the media defined in (Schulzrinne and Casner 2003) (H.261 video, MPEG video, etc.). It would be possible to define a new payload type for mHealth data such as ECG and denote its use in this way. The payload type will define important characteristics of the media stream that the monitor will use such as clock rate and precision.

Example:

```
m=audio 49234 RTP/AVP 0
```

represents a stream with media type of audio, to be delivered on port 49234 using the payload type 0 of the RTP/AVP Profile for Audio and Video Conferences which is PCMU at 8000Hz.

RTP Map ("a=rtpmap:")

Payloads can be defined using static assignments from the RTP Audio/Video profile as described above but can also be defined dynamically by the application.

The "a=rtpmap:" attribute provides a mapping from the payload type number given in the "m=" media description attribute to an encoding name denoting the format of the payload to be used along with information on the clock rate and encoding parameters.

If a payload is fully specified by the static payload type code given, then there is no need for this attribute. It is common however for an assignment to be done dynamically and indeed this would be desirable for an mHealth monitoring system as nodes might be sampling data at different sample rates depending on different situations or as a means of saving battery life.

As an example, for a typical video payload completely defined by payload type 32 in the RTP Audio/Video profile (MPV) the "a=rtpmap" attribute is unnecessary and the payload is fully specified using

```
m=audio 49234 RTP/AVP 0
```

Figure 31 Example of a payload type fully described in the RTP Audio/Video profile (MVP)

The receiving application knows that the payload format will be PCMU with a clock rate of 8000Hz.

Payload types 96-127 are reserved for dynamic assignments. In the case of an mHealth monitoring application if a dynamic payload type was defined as, say, 96 then the clock rate would be defined in the following "a=rtpmap" attribute as follows:

```
m=health 49234 RTP/AVP 96
a=rtpmap:96 L8/256
```

Figure 32 Example of rtpmap attributes in a dynamic assignment of a payload type

This rtpmap denotes 8 bit precision sampled at 256Hz.

In this way it is clear that the use of the Session Description Protocol would facilitate the creation of a WBAN sink that can advertise the various types of sensor data that it can transmit to a monitoring application via the DESCRIBE method. Similarly SDP could be used by sensors to advertise details such as the current sample rate to the sink.

Appendix B. Secure Real-Time Transport Protocol (SRTP) - Authentication and Privacy

Source origin authentication, data privacy and data integrity for streaming media are addressed through the use of SRTP.

Authentication takes two forms: integrity protection and source origin authentication. Integrity protection is knowing that the received data hasn't be tampered with and source origin authentication is confirmation that the sender of the data was who they say they were. SRTP provides integrity protection by sharing a secret key between sender and receiver which can be used to verify the data. This ensures the data has not been tampered with and has not been corrupted.

Source origin authentication is implemented in SRTP using the TESLA standard. (Carrara and Baugher 2005).

Data privacy, which is probably the most important security consideration in the context of this dissertation, is provided in SRTP by encrypting the payload of the RTP packets and encrypting the RTCP packets, including headers. A cryptographic context must be maintained between client and sender and the keys must be exchanged out of band, e.g. using RTSP. RTP header compression continues to function because only the payload of the RTP packet is encrypted.

Bibliography

Ahmadi, M M, and G Jullien. 2009. “A Wireless-implantable Microsystem for Continuous Blood Glucose Monitoring.” *IEEE Transactions on Biomedical Circuits and Systems* 3 (3) (June): 169–80.

Alwan, Dr Ala et al. 2011. “WHO Global Status Report on Noncommunicable Diseases.”

Baird, S, and S Dawson-Haggerty. 2006. “Communicating Data from Wireless Sensor Networks Using the Hl7v3 Standard.” *… Sensor Networks, …*: 4–7.

Baum, Peter, and Fabienne Abadie. 2013. “Strategic Intelligence Monitor on Personal Health Systems, Phase 2: Market Developments – Remote Patient Monitoring and Treatment, Telecare, Fitness/Wellness and mHealth.” *European Commission JRC Scientific And Policy Reports*.

Bond, Raymond, Dewar Finlay, Chris Nugent, and George Moore. 2011. “A Review of ECG Storage Formats.” *International Journal of Medical Informatics* 80 (10) (October): 681–97.

Carrara, and Baugher. 2005. “The Use of TESLA in SRTP.” *Internet Engineering Task Force <draft-ietf-msec-srtp-tesla-05.txt>*.

Casner, S, and V Jacobson. 1999. “Compressing IP/UDP/RTP Headers for Low-speed Serial Links.” *Network Working Group RFC 2508*.

Chen, Wei, and Chien-Chou Shih. 2012. “Architecture of Portable Electronic Medical Records System Integrated with Streaming Media.” *Journal of Medical Systems* 36 (1) (February): 25–31.

Danbury Hospital. 2010. “Holter Monitor Diary Instructions.” *Www.danburyhospital.org*. http://www.danburyhospital.org/~/media/Files/Patient Education/patiented-english/pdf_Diagnostic/HolterMonitorDiaryInstructions.ashx.

Doukas, Charalampos, Ilias Maglogiannis, and George Kormentzas. 2006. “Advanced Telemedicine Services through Context-aware Medical Networks.” In *5th International IEEE EMBS Special Topic Conference on Information Technology in Biomedicine*.

Handley, M, C Perkins, and V Jacobson. 2006. “SDP: Session Description Protocol.” *Network Working Group RFC 4566*.

IEEE. 2012. *IEEE Standard for Local and Metropolitan Area Networks Part 15.6: Wireless Body Area Networks*. IEEE.

ITU-T. 2009. “H.323 - Packet-based Multimedia Communications Systems.” *H.323* Series H.

Jetcheva, Jorjeta G., and David B. Johnson. 2001. “Adaptive Demand-driven Multicast Routing in Multi-hop Wireless Ad Hoc Networks.” In *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing - MobiHoc 2001*, 33. New York, New York, USA: ACM.

Jurik, Andrew D., and Alfred C. Weaver. 2012. “Bidirectional ECG Monitoring with an Event Detection Policy Engine.” In *2012 21st International Conference on Computer Communications and Networks (ICCCN)*, 1–8. IEEE.

Karter, A J, L M Ackerson, J A Darbinian, R B D’Agostino, A Ferrara, J Liu, and J V Selby. 2001. “Self-monitoring of Blood Glucose Levels and Glycemic Control: The Northern California Kaiser Permanente Diabetes Registry.” *The American Journal of Medicine* 111 (1) (July): 1–9.

Kent, S, and R Atkinson. 1998. “RFC 2401 - Security Architecture for the Internet Protocol.” *Network Working Group RFC 2401*.

Khan, Jamil Y, Mehmet R Yuce, and Farbood Karami. 2008. “Performance Evaluation of a Wireless Body Area Sensor Network for Remote Patient Monitoring.” *Conference Proceedings : Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference* 2008 (January): 1266–9.

Latré, Benoît, Bart Braem, Ingrid Moerman, Chris Blondia, and Piet Demeester. 2010. “A Survey on Wireless Body Area Networks.” *Wireless Networks* 17 (1) (November 11): 1–18.

Lau, SL, I Konig, and Klaus David. 2010. “Supporting Patient Monitoring Using Activity Recognition with a Smartphone.” *ISWCS 2010 7th Symposium* (April): 810–814.

Lusignan, Simon De, Sally Wells, Paul Johnson, Karen Meredith, and Edward Leatham. 2001. “Compliance and Effectiveness of 1 Year’s Home Telemonitoring . The Report of a Pilot Study of Patients with Chronic Heart Failure.” *European Journal of Heart Failure* 3: 723–730.

Malan, David, Thaddeus Fulford-Jones, Matt Welsh, and Steve Moulton. 2004. “CodeBlue : An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care.” In *International Workshop on Wearable and Implantable Body Sensor Networks*.

Maróti, Miklós, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. 2004. “The Flooding Time Synchronization Protocol.” In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, 39–49. ACM.

Mcgrew, D, and E Carrara. 2004. “RFC3711 - The Secure Real-time Transport Protocol (SRTP).” *Network Working Group RFC 3711*.

Moon, Sue B., Jim Kurose, and Don Towsley. 1998. “Packet Audio Playout Delay Adjustment: Performance Bounds and Algorithms.” *Multimedia Systems* 6 (1) (February 1): 17–28.

National Security Agency United States. 2013. “Systems and Network Analysis Center Information Assurance Directorate Bluetooth Security.”

Pantos, R, and W May. 2010. “HTTP Live Streaming Draft-pantos-http-live-streaming-04.”

Perkins, C. 2008. *RTP - Audio and Video for the Internet*. 2nd ed. Addison Wesley.

Ravesteijn, Wim Van. 2009. “Connecting Distributed E-health Applications by Means of a Generic Control Protocol”. University of Twente, Enchede - The Netherlands.

Rosenberg, J. 2010. “Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/answer Protocols.” *Internet Engineering Task Force RFC 5245*: 1–118.

Rosenberg, J, H Schulzrinne, and G Camarillo. 2002. “SIP: Session Initiation Protocol.” *Network Working Group RFC 3261*.

Schulzrinne, H, and S Casner. 2003. “RTP Profile for Audio and Video Conferences with Minimal Control.” *Network Working Group RFC 3551*.

Schulzrinne, H, S Casner, R Frederick, and V Jacobson. 1996. “RTP: A Transport Protocol for Real-time Applications.” *Network Working Group RFC 3550*.

Schulzrinne, H, A Rao, and R Lanphier. 1998. “Real Time Streaming Protocol (RTSP).” *Network Working Group RFC 2326*.

WHO. 2013. “WHO Factsheet 317.” http://www.who.int/mediacentre/factsheets/fs317/en/#.