Ad Hoc Networks 36 (2016) 332–348

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

# Location and mobility-aware routing for multimedia streaming in disaster telemedicine

Fraser Cadger, Kevin Curran*, Jose Santos, Sandra Moffett

*School of Computing and Intelligent Systems, Faculty of Computing and Engineering, University of Ulster, Londonderry, Northern Ireland BT48 7JL, United Kingdom*

ARTICLE INFO

*Article history:*
Received 21 February 2014
Revised 20 August 2015
Accepted 23 August 2015
Available online 29 August 2015

*Keywords:*
Geographical routing
Location aware routing
Network streaming
QoS

ABSTRACT

Disaster telemedicine leverages communications networks to provide remote diagnosis of injured persons in areas affected by disasters such as earthquakes. However, telemedicine relies heavily on infrastructure, and in a disaster scenario there is no guarantee that such infrastructure will be intact. In an ad-hoc network, devices form a network amongst themselves and forward packets for each other without infrastructure. Ad-hoc networks could be deployed in a disaster scenario to enable communications between responders and base camp to provide telemedicine services. However, most ad-hoc routing protocols cannot meet the necessary standards for streaming multimedia because they do not attempt to manage Quality of Service (QoS). Node mobility adds an additional layer of complexity leading to potentially detrimental effects on QoS. Geographic routing protocols use physical locations to make routing decisions and are typically lightweight, distributed, and require only local network knowledge. They are thus less susceptible to the effects of mobility, but are not impervious. Location-prediction can be used to enhance geographic routing, and counter the negative effects of mobility, but this has received relatively little attention. Machine Learning algorithms have been deployed for predicting locations in infrastructure networks with some success, but such algorithms require modifications for us in ad-hoc networks. This paper outlines the use of an Artificial Neural Network (NN) to perform location-prediction in an ad-hoc network.



## 1. Introduction

Millions of people are affected every year by both natural and manmade disasters. These lead not only to death and injury, but also the devastation of communities and sometimes entire nations. A common feature of such events is people trapped in an area, either those who physically cannot be moved or are cut-off from the outside world, and who require treatment. Even when such people can be reached it may not always be possible for the appropriate medical services to reach them on time. The explosion of the Internet and other communications networks, has seen the field of telemedicine go from a relatively obscure military system, to a disruptive service that millions of people around the world use to access medical care remotely. Today applications of telemedicine range from performing appointments over video conferencing to remote operation of medical equipment in surgery [20]. Telemedicine can therefore be of great benefit in providing services to people who are unable to access them directly, or allowing institutes to provide treatments they normally wouldn't be able to. In essence telemedicine seems perfect for use in disaster recovery scenarios. If doctors cannot attend an injured person then they can consult remotely, perform a diagnosis, provide instructions on treatment to others, and monitor the patients' condition. Telemedicine has been utilised during disasters, but its use so far is limited. After an earthquake in North Pakistan a field hospital was set up and webcams and computers used to communicate with

* Corresponding author. Tel.: +2871 375565.
*E-mail address:* kj.curran@ulster.ac.uk (K. Curran).

http://dx.doi.org/10.1016/j.adhoc.2015.08.016
1570-8705/

remote specialists. Of the patients seen at the hospital a total of 28 patients were treated in some way via telemedicine [9]. Another example of telemedicine being utilised in the wake of a natural disaster is the University of Texas' existing telemedicine being damaged by a hurricane and a temporary telemedicine system being set up in its place [25]. However both of these systems, although being deployed in disaster recovery scenarios, made use of existing infrastructure albeit limited infrastructure in the case of the latter. Unfortunately communications infrastructure is not always available after a disaster. Cell towers and wired connections are all prone to damage. Even when some infrastructure is intact, it may not always be suitable for performing telemedicine. When you are treating a patient remotely a certain quality of video is required, and a damaged network may not always be able to deliver it.

Where communication between people in the disaster area or the vicinity of it is desired then ad-hoc networks could be a potential solution. Ad-hoc networks are an infrastructure-free model for networking in which devices wishing to communicate with each other form a network amongst themselves. Routing is then performed on a multi-hop basis where nodes forward a packet to each other until it reaches the destination. Thus all connected nodes are not only end-users, but also routers. Ad-hoc networks are considered to be distributed and decentralised as there is no infrastructure or servers. This can be seen as either an advantage or disadvantage, as the lack of control can lead to issues in ensuring all nodes behave correctly, but it can also prevent attackers from being able to destroy the network by targeting infrastructure. While an ad-hoc network cannot bridge a divide between the disaster area and the outside world, it can facilitate communication within it. Even if medical personnel are present they may not be able to attend directly to every injured person. If a first responder with some basic medical training could communicate with a doctor located at base-camp, the doctor could then relay instructions to the responder on how to handle the patient. Where some infrastructure is intact this could be incorporated into the network to allow devices who are able to connect to the ad-hoc network to access the outside world via it. The traditional ad-hoc model does not make any provision for this, but a sub-type the Hybrid Wireless Mesh (HWM) does. HWMs are similar to ad-hoc networks in that devices form a multi-hop network, but where they differ is their ability to incorporate infrastructure that can then be accessed through the multi-hop network by devices. In a disaster recovery scenario this would allow devices able to connect to the Internet to share their connection with other devices in the network. A significant problem limiting the use of ad-hoc networks for disaster telemedicine is QoS. To provide a suitable streaming service, strict levels of packet loss, delay, and jitter must be maintained. Ad-hoc network protocols are typically best effort, with the primary aim being to forward every packet to the destination. As such, ad-hoc networking protocols do not typically mechanisms such as classification and resource reservation found in infrastructure networks. As ad-hoc networks are not centrally managed implementing such policies is fraught with a number of organisational difficulties. Similarly, another technique used to manage QoS in infrastructure networks is inappropriate; overprovisioning, whereby significantly more capacity than is typically required is installed to provide redundancy. That is not to say that managing QoS and achieving standards suitable for streaming media in ad-hoc networks is impossible. However existing paradigms developed for infrastructure networks may be inappropriate, and thus new techniques must be devised.

Such approaches must overcome not only the challenge of decentralisation, but also other factors that make ad-hoc networks unique. One such factor is the potential for dynamic behaviour. While individual devices in an infrastructure network may fail, it is highly unlikely that such devices will be removed at random, and there will probably be some form of contingency measure. In an ad-hoc network nodes may leave or join at any point. This can have a disastrous effect, as the loss of one node can leave a node without a path, resulting in potentially wasted transmissions and the need to find another path. This problem is compounded in instances where mobility is permissible. Such networks are typically referred to as Mobile Ad-hoc Networks (MANETs). In contrast to static ad-hoc networks, MANETs are significantly more dynamic as node mobility can have a huge effect on connectivity. Even where a user does not wish to leave the network, if they move outside the range of another node the connection is lost. If ad-hoc networks are deployed in disaster recovery scenarios then it is likely they will take the form of a MANET. Even if the users of the stream remain static, there is no guarantee that the users of the other device that comprise the network connecting them will remain immobile. Thus mobility is liable to play a significant factor in the performance of any disaster recovery ad-hoc network.

If end-user applications are able to make use of a device's location and mobility data then it is logical to consider the possibility of using such information at the network-layer. Geographic routing covers a broad range of protocols that make use of such information varying extents. Geographic routing originates from a technical paper published by Finn [8] that suggest the use of physical location in forwarding decisions. In its most basic form, greedy geographic forwarding, geographic routing forwards packets to neighbours based on their proximity to the destination. In addition to making use of physical locations, greedy routing is also lightweight as nodes do not store routing tables or topology. Instead nodes maintain a list of directly connected neighbours and perform forwarding on per-hop basis, selecting the neighbour closest to the destination and dropping the packet if no neighbour closer to the destination than the node itself can be found. This is done so as to avoid the possibility of routing loops where a packet travels backwards. Other approaches to geographic routing include face routing based on the Compass II protocol [11] where nodes traverse a planar graph and which theoretically guarantees delivery, but is considerably less efficient than greedy routing, as well as hybrid greedy—face protocols that combine the two approaches such as Greedy Perimeter Stateless Routing (GPSR) [10]. Geographic (or location-aware) protocols can also utilise location information to optimise specific criteria, such as in [22] where location is used to compute the connection time between two nodes, and [21] where mobility serves as an indicator of delay and jitter. These two protocols are interesting applications of how physical locations can be incorporated into QoS decisions. Both of these protocols are also novel in

their use of location-predictions. Instead of merely using existing information about neighbours' locations, they actively try to determine where their neighbours will be in the future, and thus what effect this will have on routing. Location-prediction is therefore a potentially powerful tool for geographic routing protocols, and can also be of benefit to other areas such as the MAC by reducing transmission power if all neighbours are located nearby and are expected to remain so. Despite this, there has been relatively little attention paid to location-prediction in the area of geographic routing and ad-hoc networking in general. Considerably more attention has been paid in infrastructure wireless networks such as WLANs and cellular networks, where Machine Learning has been deployed to predict the future locations of neighbours and thus assist in hand-offs and capacity management. Examples of this include the application of a Hidden Markov Model (HMM) to predict future connectivity based on mobility [18] or the use of Bayesian Networks to monitor mobility with regards to managing hand-offs. These approaches boast high accuracy and success rates. However from an ad-hoc networking perspective they are unsuitable as they perform location-prediction in terms of the infrastructure itself, by predicting what Access Point (AP) or cell a node will connect to, and not the actual geographic location of a node. There is great potential therefore, for a geographic routing protocol that is able to accurately predict the future of locations of other devices in a MANET scenario. Such a protocol could also incorporate other context information about the user and environment, and use this to anticipate their future behaviour and how such behaviour would impact the network. In a disaster recovery scenario containing a large level of dynamic behaviour this protocol may be able to counter such behaviour and route packets in the best possible manner so as to maximise QoS and thus enabling the possibility of performing telemedicine in such scenarios.

We present here a Geographic QoS Peer-to-Peer Streaming framework (GQP2PS) designed to allow for disaster telemedicine to be performed over ad-hoc networks. GQP2PS aims to utilise the capabilities of modern mobile devices such as WiFi connectivity and GPS to form an ad-hoc network of such devices. This network will then be used to facilitate multimedia streaming using a P2P model. The main challenge in achieving this is managing QoS in a highly dynamic environment. GQP2PS will therefore make use of location and other context information, to observe the local state of the network, and act on this knowledge to make QoS-optimal routing decisions.

## 2. Geographic routing

In its most general form, the term geographic routing entails some form of routing (or forwarding) using physical locations as the primary criteria. Geographic routing therefore encompasses a wide range of individual protocols and algorithms, but does not necessarily constitute a well-defined philosophy. While some of the protocols described in this section are explicitly identified as being geographic routing protocols, many do not use that term (although they may use a similar term such as location-aware/based) but clearly make use of location in the routing process, and for the purpose of this review are considered geographic routing protocols. Greedy geographic forwarding/routing was previously introduced as the most basic form of geographic routing. As greedy forwarding only considers 1-hop forwarding it only needs state information about its' immediate neighbours and is unconcerned with the larger topology of the network. This provides nodes with a certain sense of 'freedom' from topological dependencies found in conventional ad-hoc routing protocols; an event occurring on the opposite side of the network will not directly affect a node at the other. However, by using the distance between a neighbour and the destination as the sole criterion for forwarding it can give rise to a number of problems.

The most serious of which is the local maximum wherein a node receives a packet and is unable to forward it because none of its own neighbours are closer to the destination than itself, and as a result of the simple forwarding rule the packet cannot be passed backwards (this is to prevent rooting loops) and must be dropped. Thus situations could exist where there is a path from source to destination, but because of greedy routing's reliance on minimising physical distance, the packet cannot be delivered. Although this is the most widely noted drawback of greedy routing it is not the only one. By using distance as the criterion for forwarding, greedy routing ignores numerous other factors which can affect packet delivery; for instance hop count, congestion, and neighbour's energy levels can all have a significant effect on whether a packet can be delivered or not and the time it might take. Thus a neighbour which is physically closer to the destination than another could be experiencing extreme congestion and may queue the packet for a longer time (or drop the packet), or may have low energy levels and be unable to drop the packet, or simply be in a worse position hop-wise than the other neighbour. Greedy routing however, makes no distinction between neighbours other than on their physical distance to the destination and thus would be unable to prefer a congestion-free node that was only a few centimetres away from a congested node nearer the destination. As such there have been several proposed variations and alternatives to geographic routing which aim to address these shortcomings while still making use of physical location in the routing process. Several of these protocols use a variant of geographic routing known as face routing, which exists as an alternative to greedy forwarding. In face routing nodes perform traversal of a planar graph consisting of all of their directly-connected neighbours, normally using a method known as the right-hand rule. This graph is intersected by an imaginary line connecting the source and destination, and every time a node that is located on this line is encountered the algorithm keeps track of this node. When the entire face has been traversed the node closest to the destination that lies on the aforementioned line is selected as the next hop, and the algorithm continues. Although the face routing algorithm solves the local maximum problem, it has been demonstrated to be less efficient than greedy routing as it requires a total of $O(n)$ messages to route a packet to the destination where $n$ is the number of nodes in the network [13]. In contrast, greedy forwarding has a worst-case complexity of $O(d^2)$ where $d$ is the distance between source and destination [13]. Several protocols which combine elements of greedy forwarding have been developed such as GOAFR+ [12] or GPSR [10] which alternate between greedy and face

modes, typically starting in greedy mode and then using face routing as a recovery mechanism whenever the local maximum is encountered. Similarly, while basic face routing solves the local maximum in common with greedy routing, it only considers physical locations (albeit in the context of a network graph) and thus is potentially vulnerable to sub-optimal routing decisions arising from other factors.

In addition to greedy, face and hybrid protocols there exist a number of other geographic protocols that make use of information other than physical location or which use physical location to augment existing approaches. Several of these are explicitly categorised as geographic routing protocols and may be based on a greedy/face/hybrid approach, while others simply use physical location as one factor in the routing process. These can include protocols which attempt to address a particular issue related to geographic routing such as improving security [4,14] or reducing power consumption [7]. There are protocols which use location or geographic routing techniques as a means of solving another problem such as countering/utilising the effects of mobility [15] or improving QoS through the use of location information [21]. Geographic routing protocols can also be considered as examples of context-aware protocols, as location is an important aspect of context. This section provides an overview of geographic routing initially in general terms and then in the context of QoS in particular. Although there have been a number of different approaches taken to geographic routing there are some common characteristics which most protocols share (in addition to their use of location). They are:

- Localised; protocols typically only store information on their directly connected neighbours.
- Lightweight; store only limited state information about their neighbours and do not store end-to-end routes.
- Distributed; forwarding decisions are made without any coordination between nodes with packets simply being passed to the next hop.
- Best-effort; packets are sent based on the forwarding criteria without any attempt to guarantee QoS (although the forwarding criteria may try to optimise QoS).

Geographic routing protocols typically only store information about their immediate 1-hop neighbours which are obtained through a series of beacon (hello) messages sent periodically which contain physical locations (typically GPS coordinates, but other forms can be used). Nodes then store this information in a neighbour table. There is typically no routing table as nodes do not create end-to-end routes but instead forward packets on a per-hop basis; hence only local information is required. Although QoS is an emerging topic in ad-hoc network research, there has been only a small number of geographic routing protocols which explicitly deal with QoS. Two of the most notable of these are [22] and [21] both of which make use of location-prediction in order to estimate the level of QoS neighbours can offer. While there has only been limited exploration of location-prediction in ad-hoc networks, greater attention has come from wireless infrastructure networks. Several methods using ML algorithms have been proposed. The advantage of these approaches is that they can learn from previous information (i.e. mobility traces) and adapt themselves to deal with future interactions. However, although these algorithms are able to boast high-rates of prediction accuracy they are unsuitable for use in ad-hoc networks due to their reliance on infrastructure. Although these algorithms are described as location-prediction algorithms, they typically view location in terms of infrastructure (i.e. APs or cells) and therefore view the task of location-prediction as determining which AP or cell the device will be nearest when it moves. In contrast to the ML-based approaches employed in infrastructure networks, ad-hoc location-prediction algorithms such as those of [21] and [3] predict locations in the form of continuous coordinates. For location-prediction a ML algorithm could be used to predict future locations as geographic coordinates (such as GPS). This would provide nodes with the most accurate possible view of where their neighbours would be located. In addition to being used for geographic routing, an algorithm which could accurately predict future GPS coordinates, would have numerous potential applications in location-aware systems as diverse as smartphone apps and cognitive radio algorithms. This is the approach used by GQP2PS and which will be discussed in future chapters. At the time of writing, no such location-prediction algorithm exists in literature except the one used by GQP2PS.

## 3. Location-awareness in peer-to-peer streaming

Although the focus here is on wireless mesh networks, the number of P2P applications intended for wireless mesh or ad-hoc networks is relatively small in contrast to the number available over the Internet, as was evidenced in the last section. However, given the similarities between ad-hoc and P2P networks some research of note has been carried out using P2P streaming in ad-hoc networks. Within this group of protocols a very small number have sought to look at the potential for using location information in ad-hoc P2P streaming. Before looking at this category, applications of location information to conventional (i.e. Internet based) P2P systems will first be considered as there has been more research in this area. Tu et al. [24] address the issue of geographically disparate nodes being selected as peers due to being logically close in a P2P overlay. They argue that because commonly used peer selections do not consider the physical network topology, a problem they refer to as topology mismatch can occur in which unsuitable nodes are selected as peers [24]. The solution proposed by Tu et al. [24] is Nearcast a P2P overlay construction algorithm that uses physical location to avoid topology mismatch and which was found to reduce overhead and latency when compared with two other non-location-aware protocols [24]. It is important to note however, that Nearcast like the majority of P2P streaming and P2P research in general is focussed on the Internet as the underlying network [24]. However the ideas and algorithms used by Nearcast could potentially be modified for application to ad-hoc networks.

MStream is a hybrid P2P architecture that retains some features of the client-server architecture (a streaming server sends files to the root of a streaming tree and from there all streaming is P2P) and uses position data to determine location-based streaming policies (Liu et al., 2005). Another hybrid P2P streaming architecture of interest is MeTree [17] which also uses a hybrid tree-mesh architecture and takes into account underlying physical topology when building the

overlay. MeTree is intended for use in live VoD streaming scenarios where the content is provided by a content server, carried by the ISP to the peer network and then distributed between peers [17]. In MeTree, geographically close peers form subnets with these subnets in turn forming a mesh which forms part of an ISP sub-tree which is the tree structure connecting the content provider, ISPs, and peers [17]. In addition to taking into account physical location, MeTree also uses contribution to determine the 'ranking' of clusters (i.e. where it is located on the sub-tree) so that clusters that contribute more resources get higher priority (by being placed near the top of the virtual sub-tree) than those that do not [17]. Thus MeTree is able to make use of both physical location (when forming clusters) and resources contributed (when deciding where to place these clusters on the logical sub-tree). A brief mention is made of using physical location and underlay network information in the Anysee protocol [16] however this is only discussed in very general terms and does not appear to be a major part of the protocol.

Although it does not specifically deal with ad-hoc networks, [2] have analysed the performance of the Chord, Tapestry, and Kelips P2P routing protocols on a wireless network and found that the various protocols performed better than others in some categories, and that overall there was no clear 'winner' that could be said to be most suitable to wireless networks. Regarding the application of location-aware P2P streaming technologies to ad-hoc or mesh networks, there has been very little research in this area to date, although some promising early research has laid suitable foundations for future work. Qin et al. [19] present a GPS-free means for calculating the amount of time a link between two peers will exist based on the movement of both nodes. As mentioned, it is GPS-free and does not use exact position data such as coordinates but rather the distance between nodes which makes it suitable for instances and environments where GPS is not available or unusable (i.e. indoors or with poor line of sight) [19]. This is actually somewhat similar to the mechanisms used by some location-predictive geographic routing protocols that will be discussed later, although all of the location-predictive geographic routing protocols assume the ability of GPS (or similar) and used actual coordinates. Nevertheless, the approach taken by Qin et al. [19] is interesting not only because it is one of the few location-aware P2P streaming proposals for ad-hoc networks but also because of its ability to improve the performance of the aforementioned MStream framework in terms of decreased number of link breaks, increased number of nodes that do not experience breaks, increased average continuous streaming duration, and decreased waiting time caused by breaks [19]. Such an approach is potentially very suitable for use in disaster recovery scenarios due to it not relying on underlying infrastructure and its ability to estimate link duration without GPS.

At present only a few P2P streaming protocols make use of location data and of those a large number are Internet-based and most likely unsuitable for use in ad-hoc or mesh networks (although adaptation may be possible). This is in spite of several authors stressing the benefits of using physical location information and closer integration of overlay and underlay networks in general. This indicates that there is significant scope in the areas of P2P streaming and ad-hoc networking in general for the design of a protocol that is able to make use of location data (and other underlay network information) and that is suitable for deployment on an ad-hoc or mesh network where Internet connectivity is not guaranteed. Communications technology has the potential to play a significant role in recovering from disaster recovery scenarios. From a telemedicine perspective, communication with remote parties allows injured persons and responders assisting them to communicate with doctors and other medical professionals who can consult, diagnose, and observe treatment of them. More specifically, multimedia communications can allow both audio and visual interaction, widening the level of information available to clinicians. In the first section, two applications of telemedicine in disaster recovery scenarios were discussed. In both of these instances communications and medical infrastructure were affected, but a telemedicine system was employed. In the first [25] there was already a telemedicine system in place, and the emphasis was on how the system responded to dealing with the needs of affected persons, and resuming operations. On the other hand, the telemedicine approach discussed in [9] did not feature an existing telemedicine system but did make use of existing infrastructure outside of the disaster area. Both of these instances show the significance of being able to establish contact with external facilities unaffected by the disaster. However, in many scenarios this is impossible or only limited communication is available.

An alternative approach to the problem of disaster recovery telemedicine is to consider the deployment of an ad-hoc network. The devices would form a network between themselves, allowing responders to communicate with each other no matter where they are located. Responders could then communicate directly with each other, or with a base station. If external communications infrastructure exists then devices in the ad-hoc network could connect to these networks and share it with other users of the ad-hoc network. Although ad-hoc networks are an exciting prospect for disaster telemedicine there are a number of technical challenges that limit their application. One of the most significant of these is their ability to handle multimedia traffic such as voice or video, particularly interactive traffic. Previous attempts at running streaming applications such as VoIP over ad-hoc networks have not been particularly successful [1,23]. However, when considering the failure of these experiments it is important to recognise, that in both of these instances ordinary ad-hoc protocols with no support for QoS were used. Protocols such as AODV and DSR may be unsuitable for heavy multimedia traffic without any modification, but that does not mean that ad-hoc networks in general are. While the majority of ad-hoc routing protocols focus on the traditional client-server model of streaming in which the source unicasts to the destination, for the purposes of streaming in disaster recovery scenarios it is worth considering an alternative approach. Conceptually, P2P networks and ad-hoc networks can be considered similar in a number of ways, as they are both networks of end-users who share resources and operate in a distributed manner. Just as ad-hoc networks make sense for situations where there is limited or no infrastructure, P2P streaming could be appropriate for use over a distributed network where end-users are scattered across a physical area. P2P streaming removes the reliance on servers

and by harnessing the resources of the network helps reduce bottlenecks.

## 4. GQP2PS

It is not enough to merely provide a means of communicating without infrastructure, but to ensure that such communications are able to be understood by human beings as anything else is just noise. Providing a means of doing so over an ad-hoc network is a nontrivial challenge and this is where the novelty of this work lies. The focus of this research project has therefore been on designing a mechanism that allows ad-hoc networks to deliver streaming multimedia of a suitable quality for telemedicine. This is achieved through the use of GQP2PS. Because the environment in which it is deployed will have a great impact on its operation, and in turn this will have affect the quality of video delivered to the user, GQP2PS must be able to understand the environment in which it operates, and make the right routing decisions so as to maximise QoS at every step of the journey. GQP2PS must therefore be aware of the context it operates, how this affects its operations, and how to modify its operations to fulfil its requirements. The purpose of GQP2PS is to deliver streaming multimedia over an ad-hoc network. This requires some means of acquiring and then distributing the multimedia content. Traditionally, this would be achieved through the use of a streaming server that was fed a content stream (i.e. video) from a file or live (from the application creating the stream), which it would then place into packets and send over the network. Depending on the nature of the stream, it could be broadcast, multicast or unicast. The stream may be delivered to a specific destination (or group) or could simply be broadcast over a specific port. Server streaming typically requires some form of connection between the receiving clients and the server, where the client requests the stream and the server sends it.

The main alternative to server-based streaming is P2P streaming. In P2P streaming the initiating node plays a role similar to the server as it is responsible for acquiring the streaming content and transmitting it to other nodes. This is where it differs from server-based approaches, as these are responsible for directly sending the stream to clients, whereas portions of a P2P stream can be acquired from any peer that has them. Although both systems require a node that creates the stream, P2P solutions allow peers to acquire portions of the stream from any other peer in the network, instead of having every node acquire the stream directly from the server. This means that if the originating node goes down, peers can still obtain previously transmitted chunks of a stream from other peers, whereas when a server goes down the stream becomes unavailable. Conversely, a client-server solution requires less configuration, as clients simply need to contact the server and initiate a stream, whereas a P2P solution must first create a network of peers and then perform some form of peer selection to decide which peers the stream is obtained from. As GQP2PS uses a hybrid overlay network, its streaming method is based on the P2P streaming with some modifications to take into account the underlying ad-hoc network. Like P2P streaming, GQP2PS does not use a server and the originating node is responsible for preparing the stream and transmitting it to the network. GQP2PS is intended to support both one-to-one and one-to-many streaming of both live and pre-recorded media. However, at present only one-to-one streaming has been implemented. This is ostensibly similar to client-server streaming, however while the originating node is responsible for procuring the stream and sending it to the receiving node, it is performed over an overlay network. A one-to-many implementation would follow a more conventional P2P streaming approach with peers tracking each other to determine which peers have which portions.

A major difference between this approach and client-server streaming is that the originating node does not so much serve a stream, so much as transmits portions of a stream from the receiving devices to reassemble. Using the example of a video stream, this means that the originating node will receive a video stream, split the stream into frames, split these frames into packets, and then transmit them over the network for the receiving node to piece together independently. This requires that both devices are able to understand the format that frame segments come in, and are therefore able to reassemble them and play them in sequence.

### 4.1. Context-aware routing

In addition to requiring a suitable network and streaming protocol, GQP2PS also requires a routing protocol that is able to understand the needs of the application (and by implication, its users) so as to ensure its packets are delivered to a suitable level of QoS. The literature review highlighted the lack of research in the direction of ad-hoc routing protocols intended to support multimedia. While there have been a few novel protocols (some of which address streaming) presented, these constitute a small minority of ad-hoc routing research. Similarly, studies that have sought to evaluate the performance of existing ad-hoc routing protocols supporting multimedia traffic have generally found them lacking. Given that infrastructure networks almost always have some form of traffic management and prioritisation intended to provide time-sensitive, interactive services with QOS and preferential treatment, it is not unsurprising that ad-hoc routing protocols that treat multimedia traffic as though it is standard non-interactive traffic perform poorly from a multimedia QoS perspective. There is a need for ad-hoc routing protocols designed to handle interactive multimedia traffic, however techniques applied in infrastructure networks such as classification and prioritisation alone are not necessarily sufficient to provide suitable QoS in ad-hoc networks. This is because ad-hoc networks are often tightly-constrained in terms of available bandwidth and other resources; therefore simply giving greater priority to QoS-sensitive packets may not be enough. Thus, even when enough bandwidth is available to support the stream, simply prioritising certain types of traffic does not prevent nodes from being overwhelmed and creating bottlenecks. Load-balancing can be applied to reduce bottlenecks, as can resource reservation which guarantees specific QoS requirements will be met (i.e. by explicitly reserving portions of bandwidth). However, when considering these options it is important to take into account the dynamic nature of ad-hoc networks – particularly those

deployed in disaster areas. Device mobility is likely to be high, and from an ad-hoc network perspective this can be a source of significant disruptions as whole routes can be rendered useless due to the movement of one device. Therefore, any attempt to deliver QoS in ad-hoc networks must take into account the innate characteristics that make ad-hoc networks unique.

GQP2PS proposes to do so by using an approach to routing that only takes into account the uniqueness of ad-hoc networking, but that attempts to take into account all possible factors that affect routing. The approach used for routing by GQP2PS can therefore be categorised as context-aware, in the sense that GQP2PS attempts to assimilate as much information as possible about the device and its environment. This includes obvious characteristics such as signal strength/range and energy levels, but also other less obvious factors such as those pertaining to mobility and device usage, that can have significant effects on network performance. Mobility is of particular interest to GQP2PS, and is the primary reason GQP2PS's routing is based on geographic routing. Mobility in ad-hoc networks can lead to the breakdown of routes, resulting in lost packets, as well as an increase in traffic as nodes seek to alert others of this damage and find backups. Generally speaking, mobility is a serious threat to end-to-end routing, and even alternatives such as the location-aware hop-by-hop approach of geographic routing are also susceptible to the negative effects of mobility. From a QoS perspective, routing has an obvious impact on reliability by disrupting routes leading to lost packets, but it can also negatively impact delay by causing packets to be buffered while trying to recover a route, or new bottlenecks emerging as a result of nodes switching to backup routes.

Mobility is therefore at the heart of GQP2PS's context-awareness, primarily taking the form of a location prediction algorithm that allows GQP2PS to predict the future state of its neighbours. This allows GQP2PS to perform an enhanced version of geographic routing that takes into account factors other than location. Location predictions allow nodes to determine where their neighbours are, or will be, at a particular time based on their previous behaviour. Nodes are then able to make routing decision based on this information so as to avoid sending packets to out-of-range neighbours, and anticipate the need to select an alternative next-hop in advance of movement. More specifically, by predicting how mobility will affect routing, nodes are able to determine what effect it will have on QoS and take steps to mitigate or avoid negative effects. Although mobility is treated as a significant factor affecting ad-hoc QoS, it is not the only factor, and GQP2PS therefore seeks to use a context-aware approach that takes into account other factors.

#### *4.1.1. System architecture*

GQP2PS is best described as a framework, in the sense that it more than a single protocol, algorithm, or piece of software. It is important to emphasise that while the test bed implementation of GQP2PS takes the form of an Android app, GQP2PS is not in itself an Android app. The purpose of the test bed implementation is to provide a means of testing the general principle behind GQP2PS; that context-awareness, specifically location-awareness, is capable of optimising ad-hoc QoS to such an extent that it can support streaming multimedia. All three of the functional areas discussed earlier relate to this, with the hybrid overlay network providing the quasi-infrastructural underpinning that the routing and streaming services will build on. Routing and streaming themselves are closely intertwined as both need to share information about their state with each other. Although there are three identified functionality areas, GQP2PS can ultimately be considered to consist of two main components; streaming and networking. While the hybrid overlay network and context aware routing are two distinct functional areas, it is best to consider them as part of a wider networking component. This is often the case in ad-hoc networking, where protocols described as routing protocols are also responsible for neighbour discovery and topology determination. The network portion of GQP2PS is therefore responsible for building an ad-hoc hybrid-overlay network on top of WiFi broadcasts, presenting this network to the streaming application, performing routing of all GQP2PS traffic, and maintaining the network. The streaming portion of GQP2PS is responsible for acquiring the streaming content, preparing the stream, and management of both the stream and its playback. The two portions of GQP2PS are Location-Aware Peer-to-Peer Streaming Environment (LAPSE) and Geographic QoS Predictive Routing (GQPR). 8 provides a high-level diagrammatic overview of the two components and their relationship. In this diagram, GQPR sits above a cloud representing the physical WiFi network GQP2PS operates on, while LAPSE is linked with a circle representing contact with the user interface and by extension the user. GQPR is also enclosed by a circle representing the hybrid overlay, although GQPR is responsible for managing this, from a conceptual point of view it makes sense to consider this as a sphere that GQPR operates within. This circle does not contain LAPSE, but it does overlap with it, signifying that LAPSE operates on top of the network, as well as below the user. LAPSE can also be considered as the link between the user and the functionality of GQP2PS, so by extension it is also the link between the user and the hybrid-overlay network GQP2PS runs upon. The arrows between LAPSE and GQPR represent the sharing of context information between these two components. As GQPR represents the network, and LAPSE represents the users' interaction and data, both are responsible for obtaining information relevant to their purpose and sharing it with the other proportion. The cooperation of these two components is key to the context-aware nature of GQP2PS. By interacting with and receiving information from LAPSE, GQPR is able to make QoS predictions that not only take into account the state of the network and the behaviour of other devices, but also the state and behaviour of the end-user that is providing it with data to route. Similarly, GQPR provides LAPSE with a view of the network, and allows it to make streaming decisions based on QoS. These unique interactions are in addition to common data flows such as LAPSE sending pieces of a stream to GQPR to send over the network (and vice versa).

Creating and managing a streaming session is largely handled by LAPSE which is also responsible for performing all telephony-related functions. GQPR provides LAPSE with information about the node's it can connect to, and LAPSE translates this information into a directory-like view of the network, with peers viewed as contacts in a phonebook. When initiating a stream, LAPSE is responsible for identifying

the peer and passing the information to GQPR, which then creates its own record of the sessions and routes the packets. The reason that GQPR creates a session record is to ensure that packets are treated as part of the same stream, and to avoid the possibility of mix-ups between sessions. When the user chooses to end a stream, a teardown request is sent in a similar manner, and any video packets received after this will be discarded. All functions pertaining to the receiving and playback of a stream are handled by LAPSE, however this is done using information from GQPR to adjust parameters to take into account network conditions (for instance, by setting the threshold for how long a frame can be buffered based on predicted delay). On the other hand, all network management is performed by GQPR. This involves the transmission of messages to discover other devices. From these messages GQPR then discovers indirectly connected nodes, and uses this information to create the hybrid-overlay network which it then shares with GQPR. Through regular update messages, GQPR is able to learn about the state of its neighbours and their neighbours; this information is also passed to LAPSE. The information GQPR sends includes information that it has obtained from LAPSE, as well as network information and information obtained from lower layers such as radio environment. Routing is performed by using a modified geographic routing protocol that retains the hop-by-hop nature, and uses QoS predictions to determine the best possible route. These predictions use previous information from neighbours to determine the suitability of each neighbour for forwarding a packet. As GQPR takes into account the state of its neighbours, it can help avoid bottlenecks by recognising that a neighbour is experiencing a large level of delay and deciding to route via another suitable neighbour. GQPR could also use neighbour information to conserve energy, by avoiding using neighbours that were far away, or choose neighbours with high power levels over those with lower ones. Fig. 1 shows some of the interactions between LAPSE and GQPR when performing two common tasks – initiating a stream and serving a stream. Note that although these two tasks are significantly different, one abstract diagram has been used to model the flow of information for these two tasks. The reason for doing so is that despite the actual information differing, the flow of control between LAPSE and GQPR remains the same and thus can be abstracted.

For the case of stream initiation, the user initiates a stream by selecting a recipient from a directory provided by LAPSE. This directory is obtained from GQPR and regularly updated as old peers become unreachable and new peers join. After the user selects a recipient, LAPSE formats an initiation message and passes it to GQPR to create the connection. This interaction between the user and LAPSE is modelled by the two arrows depicting the follow of information between the user and LAPSE. The arrow on the right hand-side represents the directory presented to the user by LAPSE, while the arrow on the left represents the user selecting a peer and initiating a stream via LAPSE. Similarly, LAPSE itself obtains information about the network from GQPR and this is represented by the arrow going from GQPR to LAPSE, while the arrow from LAPSE to GQPR represents LAPSE passing the stream initiation request to be routed. GQPR receives the request and first determines whether the peer is directly connected or not. If the peer is directly connected then GQPR sets the packet to be sent directly, otherwise GQPR determines a route. GQPR determines this route by using QoS predictions to determine the most suitable next-hop. When the initiation request is at the other end, GQPR identifies the initiation request and forwards it to LAPSE. LAPSE determines that a stream is being initiated and alerts the user via dialogue. If the user accepts, LAPSE creates a new instance of the streaming session, and sends a message to GQPR, which routes the packet as described previously. The initiating node then receives this packet and begins the streaming session. Again, arrows depict the flow of information. An arrow from GQPR to LAPSE shows the stream initiation request being received, which LAPSE processes and in turn presents to the user. The user then makes a decision and this is passed back to LAPSE which must determine whether to begin streaming or deny the stream initiation request, either way LAPSE does so by sending a message via GQPR. The GQP2PS architecture is important because it provides a single, unified vision of GQP2PS that can be implemented in different ways and also because it provides the guidance necessary to devise a platform-specific technical design. This will be discussed in the next chapter, which details how experimental GQP2PS implementations of GQP2PS were created, first in the ns-2 simulator and then on a test bed of Android devices. The functionality of GQP2PS is provided by two components and most importantly the interaction between them. LAPSE is responsible for creating and managing streaming sessions, which involves providing an interface with the host device to obtain video and then to playback received video. The design of LAPSE is inspired by P2P overlay networks, which are distributed and decentralised, and therefore similar to ad-hoc networks. GQP2PS differs from this traditional separation by providing a novel hybrid-overlay network that integrates these two approaches into one. By taking advantage of the nature of wireless network in which all transmissions are effectively broadcasts, GQP2PS is able to implement an overlay network on top of UDP broadcasts. This network is presented to LAPSE in the form of a list of connected peers. The responsibility for maintaining the network lies with GQPR. When GQPR receives data from LAPSE it must determine how to route the packet so as to ensure the best possible QoS. There are a number of factors which GQPR must take into account when doing so, one of the most prominent of these is mobility. GQPR is based on geographic routing, which forwards packets based on physical locations to

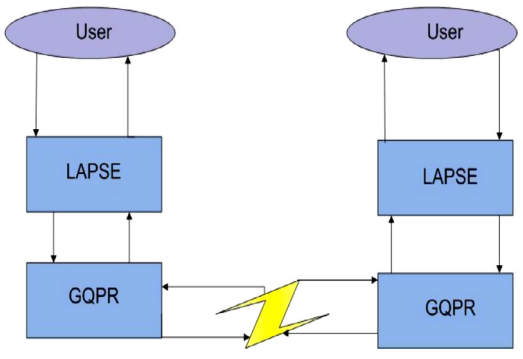



**Fig. 1.** Overview of relationship between LAPSE and GQPR.

minimise the distance between the packet and destination. Geographic routing is more distributed and lightweight than other approaches, and reduces the amount of network information to be maintained and distributed. While geographic routing does have a degree of resilience towards mobility, it can still experience negative effects. GQPR counters this through the use of an ANN to predict the future locations of neighbours. This allows GQPR to make routing decisions based on where other devices will be at the time of transmission and not where they were previously located. Location-awareness is therefore part of a wider strategy of context-awareness which tries to understand the local state of the network and use this to determine the best routing strategy to maximise QoS. As a whole GQP2PS has been designed to achieve the goal of maximising QoS and deliver high-quality to its end-users, and this is also reflected in the design of its components.

*4.1.2. LAPSE architecture*

LAPSE can be considered as responsible for all non-networking tasks, a remit that can be broken down into streaming and telephony-related functions. While most of the focus in this chapter has been on LAPSE's streaming functions, it is important to note that telephony is an essential part of GQP2PS's operation, as without it, conducting a stream would be extremely difficult. Telephony involves identifying a target/destination node (or contact from the user's perspective), sending an initiating request, accepting or denying an initiation request, ensuring that a session remains 'alive', and ending a session. The session in encapsulated by a session object. The use of the term telephony may seem confusing here, and it may also seem more appropriate to place these operations under stream management (or more specifically, streaming session management), however there are several reasons for keeping session management separate from the actual streaming. The first is an implementation issue, arising from the fact that GQP2PS is implemented on top of the Serval Mesh app and that Serval already possesses suitable telephony functionality. Discarding this functionality would be wasteful, and with minor modifications Serval telephony can be adjusted to manage streaming sessions. Doing so also allows GQP2PS to retain the functionality of Serval's VoIP telephony, alongside its own video streaming. The exact details of how this works will be covered in greater depth in the Implementation chapter. Streaming and telephony are also separated for design purposes, so as to provide the necessary abstraction that allows streaming to focus on all multimedia elements of the stream, while telephony handles the session. This approach means that both the streaming and telephony components can function independently or in different systems. LAPSE also acts as the interface between the user and GQP2PS, by virtue of the fact that all user interaction goes through it. Through its interaction with GQPR, LAPSE provides the user with a directory-like view of the network showing all connected peers, and allowing the user to initiate contact with them. Similarly, by acquiring, processing, managing, and then presenting streams to the user LAPSE is also responsible for all handling of multimedia content. However LAPSE is not a UI in the classical sense, and while the test bed implementation takes the form of an Android app, the interface presented to the user should not be considered a part of LAPSE, but a layer running on top of LAPSE. Fig. 2 shows the system architecture of LAPSE.

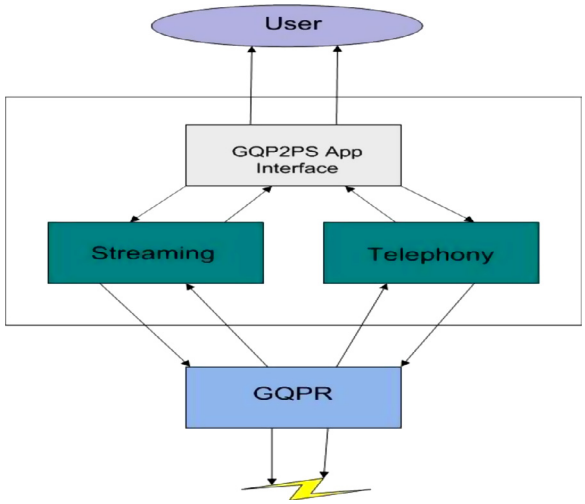



**Fig. 2.** LAPSE architecture.

The user is represented by a circle above LAPSE accessible through the GQP2PS app interface, while both the physical network and hybrid-overlay network are represented as a single cloud, accessible through GQPR. Both the GQP2PS app interface and GQPR can be seen as black boxes through which LAPSE interacts with the end-user and network. Although LAPSE (and GQPR) are implemented as part of an Android app, from a design perspective the separation between LAPSE and the interface is intended to reinforce the difference between LAPSE (and by extension GQP2PS as a whole) as a conceptual design and its implementations, whatever form they take.

Arrows denote the flow of information and control within LAPSE components and between LAPSE and other functional areas. On the right hand side of the diagram, an arrow indicates the app presenting the user with a peer list, this is in turn provided to the app by the telephony module which itself receives the network state from GQPR and presents it to the user in a suitable format. The arrow coming from the user to the app indicates the user controlling (initiating and ending) a streaming session, which is processed in the telephony layer with the transmission of the relevant packets being handled by GQPR. The reverse takes place when the other end responds. The left hand side of Fig. 2 shows the streaming component receiving streams from the user app and the network. Streams received from the network are processed and pieced together, before being buffered pending playback, all of which takes place in the streaming component except playback itself which is the responsibility of the app with LAPSE providing an input stream. Streams received from the app (i.e. frames sent from the camera) are processed, split and sent (via GQPR) by the streaming component. A two-way arrow between the streaming and telephony components indicates their interaction. When a session is initiated telephony informs LAPSE to start streaming, and when a session is ended, telephony alerts LAPSE.

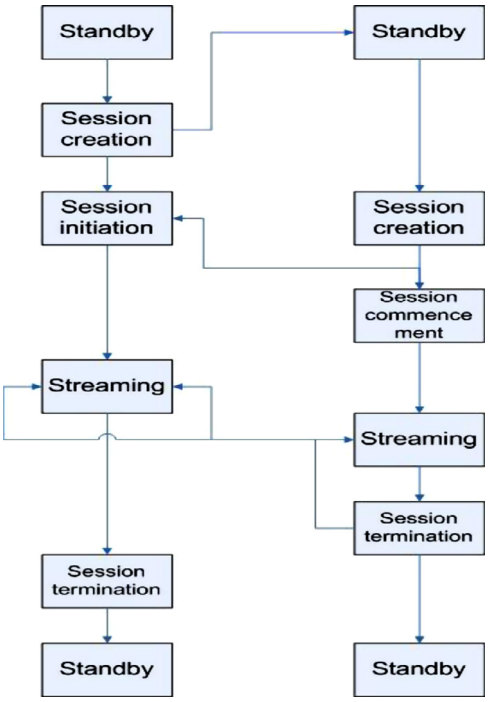



**Fig. 3.** Telephony states.

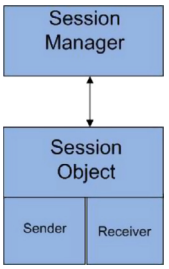



**Fig. 4.** Telephony architecture.

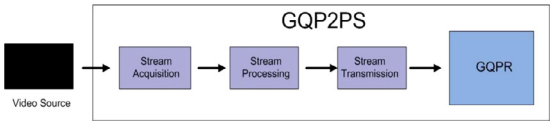



**Fig. 5.** Process for acquiring and sending a stream.

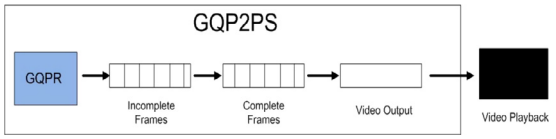



**Fig. 6.** Process for receiving and outputting a stream.

*4.1.3. Telephony*

While much of the operation of the telephony component is dependent on implementation, there are several main stages through which session initiation must go through. These stages can be abstracted as requesting a call, acknowledging receipt of a call, accepting a call, and beginning a call. When a call is to be ended, the peer wishing to end the call sends a hang-up message to the other peer, which terminates its session, before sending an acknowledgement of the hang-up. Fig. 3 illustrates the life cycle of a streaming session. To perform these tasks, the telephony module requires a means of managing the interactions between other components that are necessary to create, maintain, and end a session. For instance, once the user has decided to create a session, the telephony module is then responsible for sending (and receiving) the appropriate messages via GQPR, maintaining the current state of a session, and determining when the session has started and informing the streaming module to start streaming. Performing these tasks therefore requires a means of recording the state and status of call, in order to allow effective session and avoid issues such as never-ending or overlapping sessions. Fig. 4 shows the internal architecture of the telephony module.

Fig. 4 shows two main blocks, the session manager and the session object, with the session object being split into two types; sender and receiver. The role of the session manager is to act as a gatekeeper and point of contract for the app, the streaming portion of LAPSE, and GQPR/the network to request the performance of tasks related to the streaming session. All information related to the state of the session is held in the session object, with a new object being created for each session. The diagram depicts the session object as being split into sender and receiver so as to model the states of the two peers. Although only one session object is created, the object itself is divided into three parts one modelling the general state, and the other two modelling the state of the sending and receiving peers. The reason for this is that some session management tasks rely on knowledge of the other peer's status in relation to its own status.

*4.1.4. Streaming*

When discussing the architecture of LAPSE's streaming operations, it is again important to recognise that many features are implementation-dependent. For instance, in order to perform streaming LAPSE requires some means of obtaining streaming media from the source device. Depending on whether the stream is live or on-demand, this would be either a recorded video file or a means of accessing the device's camera directly. In the case of the latter, the exact means of doing so are heavily dependent on the device and operating system in question. The high-level design of LAPSE therefore does not focus on these issues, but instead presents an overview that assumes LAPSE is able to obtain the stream and does not require knowledge of how to do this. For the Android prototype of GQP2PS, the exact details of stream acquisition will be described in the Implementation section. Similarly, a means of playing received streams is also required, and from a design point of view, LAPSE assumes that there will be some means of providing this functionality in any implementations. Figs. 5 and 6 provide an overview of the process of sending and receiving a stream using LAPSE. In both diagrams, the large rectangle depicts the elements of the process belonging to GQP2PS and GQPR is represented by a sky blue square. Fig. 5 shows the internal components of LAPSE represented by lavender boxes, while Fig. 6 shows the internal buffers. The telephony components of LAPSE are omitted

from both. A black box is used to represent both the source of the stream as well as the destination. This is because both the source and destination are not part of GQP2PS and so long as LAPSE receives the stream in a suitable format and has an appropriate destination, how the stream is created or used is not relevant to the operation of LAPSE.

Fig. 5 depicts sending and assumes that a session has been created and initiated between two peers, as described in the Telephony section. The first stage of streaming is stream acquisition, where LAPSE receives a portion of the stream from the host device. This example assumes that it is a live stream, and that LAPSE is receiving a continual stream of frames from the camera. Upon receiving a portion of the frame LAPSE performs processing to create a suitable format for transmission. Depending on the stream and its source, this may involve compressing the frames to reduce bandwidth, encoding it in a particular format, or splitting each individual frame into a number of smaller segments. The latter is likely to be performed in most implementations both to reduce the size of packets being transmitted and to decrease the impact of packet loss by avoiding the possibility of an entire frame being lost due to a single packet drop. Once processing has been completed, the frame or its segments are then prepared for transmission over the network. This does not directly modify the frame or segments, but instead consists of gathering metadata about the frame/stream. The exact nature of the metadata will vary depending on implementation, but assuming the frame is split into segments then at a minimum the following fields will be used; frame number, frame size, and segment number. Both frame number and segment numbers are used for sequencing purposes, to allow the receiving peer to determine what order the complete frame should be in and then when re-building a frame from segments to determine where each segment should be placed. More information on how this is performed will be provided in the discussion of receiving a stream. Frame size details the size of the complete frame and is used by the receiving peer to determine the size of the frame, and construct an object of suitable size. It is included as a field in all segments, because due to the nature of ad-hoc networking there is no guarantee that all segments will be received in the correct order, therefore the first segment of a frame that a peer receives may not be the first segment sequentially, and it will need the information to build the frame object. After preparation has taken place, the frame/segments along with the relative metadata are passed to GQP2PS for routing, and LAPSE begins the process with the next frame.

Upon receiving an in-bound segment, LAPSE uses the above metadata to determine what to do with it. First, LAPSE checks the incomplete frame buffer, one of two buffers it maintains, to determine whether it has received any segments from that frame. Although the implementation of this will vary from platform to platform, it is likely that the frame buffer will be represented as a basic data structure such as an array or linked list. Similarly, individual frames will most likely be objects or structs depending on the language used. Assuming the implementation uses an array of frame objects, LAPSE will traverse the buffer to determine whether there is an object representing the frame the segment belongs to. If the object exists in the buffer LAPSE will place the segment into the correct position and update the statistics for the frame. When there is no corresponding frame object this means one of two things, either the segment is the first part of the frame to be received, or the frame has been removed from the buffer (either to be played, albeit with a segment missing, or because it has expired). To determine which of these is responsible, LAPSE will inspect the sequence numbers of other frames in the buffer and estimate whether it is possible the frame has already been handled. If this is the case, then the segment will simply be dropped. Once a frame object/struct has been created in the incomplete buffer, LAPSE will continue to receive segments and piece them together, while updating the frame's statistics. LAPSE will also perform regular maintenance to ensure that frames that are no longer viable for playback are removed from the buffer. This was discussed briefly in the Functional Architecture section, and the exact means of performing this are implementation-specific. When a frame is completely reassembled LAPSE will attempt to transfer it from the incomplete buffer to the complete buffer. The complete buffer uses the same structure as the incomplete buffer, but consists of complete frames (or frames that have been deemed complete). If the complete frame buffer is full, LAPSE needs to determine whether to remove a frame from it or keep the newly-completed frame in the incomplete buffer. Assuming there is no space in the incomplete buffer LAPSE will inspect the complete frame buffer to first determine if there are any frames that have expired, and if not will attempt to perform a trade-off to decide whether or not to remove an existing frame. The trade-off will involve trying to determine the effect that removing an existing frame from the complete buffer will have on playback, for instance if the frame is part of a complete sequence then it is less-likely to be dropped then a frame that does not presently fit into a sequence.

Although LAPSE is not responsible for playback itself, and does not provide a media player, it does provide an interface between the media player and GQP2PS. LAPSE continuously monitors the complete buffer to determine whether a sequence of frames is ready for playback. A sequence is identified as a set number of frames (representing x seconds of video playback) that are available in the buffer. For instance, assuming a frame rate of 15 fps, a one second sequence would consist of 15 frames in the complete buffer. Identifying a sequence also involves determining the relationship between a sequence and other frames in the buffer. For instance, if a complete sequence is identified but there are frames in the buffer that have an earlier number, but do not fit into a sequence (i.e. there are gaps between a sequence and earlier frames) then LAPSE needs to determine whether to play the sequence and drop the earlier frames (so as to avoid playing frames in an incorrect order) or delay playback of the sequence. When a sequence has been deemed fit for playback, LAPSE passes it to the media player (or interface) responsible for playback. Depending on the implementation, this could take the form of a data-type such as an input stream, or may consist of LAPSE handing raw images to the media player. With regards to the exact implementation of tasks such as buffer monitoring and how LAPSE determines whether a frame should be dropped or delayed, these are not only dependent on the implementation but also use. For instance, depending on the application of GQP2PS, there may exist the possibility of the user defining acceptable playback quality

levels, which could include a maximum buffer time, therefore the duration a frame could be held by LAPSE would be dependent on this. Similarly, an important aspect in the design of LAPSE is its interaction with GQPR, therefore information obtained from GQPR about network conditions can also play a part in managing streaming. From a receiving perspective, network information could be used to estimate the level of delay caused by the network, and adjust buffering policy to take this into account. On the other hand, if the network is congested then LAPSE could attempt to reduce its transmission volume by applying extra compression or switching to a lower frame rate to avoid saturating the network.

## 5. Testing and evaluation

The configurations described in this section refer to the evaluation of GQPR using ns-2. Note that the CMUPriQueue queue was used for simulations of DSR due to a bug in DSR. For GQPR a beacon period of 10 s a congestion control alpha value of 0.001 was used. While ten seconds may seem like a high value this is based on GQPR's ability to predict future locations and thus reduce the number of beacons required, while the 0.001 alpha value was arrived at after experimenting with other values. Simulations of 10, 30, and 50 nodes using the RWM, RPGM, and GM models were performed giving a total of 9 unique scenarios. All simulations use a maximum velocity of 2.5 m/s and a maximum pause time of 20 s. The RPGM scenario used a join probability of 0.75 meaning nodes have a 75% chance of joining a group, while the GM scenarios used an update frequency of 1, angle standard deviation 0.5, and a speed standard deviation of 0.5 to allow a suitable mix of random and non-random mobility. These mobility models were chosen because in addition to being used for earlier experiments they reflected different and diverse aspects of mobility modelling. RWM is purely random, whereas RPGM incorporates both individual and group mobility, while GM exhibits varying degrees of random and non-random mobility. While there are mobility models based on disaster scenarios, it was decided not to use these for two main reasons. The first is that there is a great diversity between disaster scenarios, and attempting to capture all of the necessary characteristics would have led to an unreasonably large number of simulation scenarios. The second reason is that while GQPR is intended for use in disaster recovery scenarios, it is also suitable for other purposes, therefore it was decided to simulate general mobility using the aforementioned models. For traffic the following configurations were used. For the 10 node scenario 1 video call and 1 video stream, for the 30 node scenario 2 video calls and 4 video streams and for the 50 node scenario 3 video calls and 4 video streams. Each video call consisted of two nodes sending CBR packets of size 512 bytes and with a send rate of 58 packets per second. Video streams also use 512 byte packets but have only one node sending and use a higher send rate of 128 packets per second and is intended to reflect the streaming of 360-480 p traffic. These scenarios are intended to realistically model video calling/VoIP and on-demand video streaming based on figures from Cisco Systems [5]. It was decided to use traffic characteristics based on these applications instead of the applications themselves, as simulating real VoIP and video streaming traffic in large topologies would take a great deal of time. To evaluate the performance of GQPR the three standard QoS metrics of reliability, delay and delay variation were used. Reliability is the rate of data packets successfully delivered, while delay is the duration between a packet being created and received, and delay variation is the standard variation of packet delays at a node. While there are no fixed QoS parameters and the amount of visible disruption a user will be willing to tolerate varies on the individual, there are some good practices with regards to QoS. For instance, Cisco recommends delay not exceed 150 ms and delay variation no more than 30 ms (Cisco, 2014) with no recommendations for packet loss. Other sources state up to 300 ms (VBrick, 2014) is an acceptable level of delay for streaming video. Joshi and Rhee (2000) state that a loss of 10% may be acceptable, while some other sources state that loss of more than 3% will lead to noticeable detrimental effects [6]. It is important to recognise that these metrics are intended for streaming over infrastructure networks or the Internet, which will have greater resources available than that of an emergency MANET. However, if the video being streamed by GQP2PS is not of adequate quality from the user's perspective, it will be of little use. Therefore, the users may have reduced perceptions of the quality available minimum standards must be adhered to. Thus delay should typically be below 300 ms and packet loss below 10% and preferably 3%. Note that due to DSR continually freezing on the 50 node RWM scenario there are no statistics for its performance here.

**Table 1**
Reliability for RWM scenarios.

| Protocol | 10 nodes | 30 nodes | 50 nodes |
|---|---|---|---|
| GQPR | 77.6 | 83 | 73 |
| AODV | 90.3 | 75.2 | 71.2 |
| DSR | 43.8 | 80.3 | – |
| DSDV | 48.8 | 73.5 | 67 |

### 5.1. Random waypoint mobility

#### 5.1.1. Reliability

Table 1 contains the reliability results for the RWM simulations. In the 10 node scenario, only AODV is able to attain a standard close to the requirements for streaming QoS. Although GQPR comes second, 77.6% packet delivery would generally be considered unsuitable. Both DSR and DSDV perform extremely poorly in this scenario. All protocols except AODV show a marked improvement in the 30 node scenario, and GQPR comes close to reaching a level suitable for multimedia streaming, but falls short by 7 p.p. GQPR again outperforms AODV (and DSDV) in the 50 node scenario, but again the result obtained here is unsuitable for streaming QoS.

Regarding the overall performance, it is interesting to note that all protocols (except AODV) experience an increase in packet delivery between 10 and 30 nodes, but then a decrease at 50 nodes. The mobility created by the RWM is most likely a factor in the poor performances seen here. As the RWM is purely random, it is to be expected that routing will be disrupted by the constant and unpredictable mobility. Although the NN location-prediction algorithm used by GQPR was often able to accurately predict future locations in RWM that does not necessarily mean that it will always be able to

**Table 2**
Delay for RWM scenarios.

| Protocol | 10 nodes (ms) | 30 nodes (ms) | 50 nodes (ms) |
|---|---|---|---|
| GQPR | 4 | 5.9 | 8.6 |
| AODV | 2060 | 2200 | 1798 |
| DSR | 12.4 | 1910 | – |
| DSDV | 9.6 | 310 | 556 |

**Table 3**
Delay variation for RWM scenarios.

| Protocol | 10 nodes (ms) | 30 nodes (ms) | 50 nodes (ms) |
|---|---|---|---|
| GQPR | 2.9 | 20 | 22 |
| AODV | 2269 | 2940 | 2711 |
| DSR | 110 | 4144 | – |
| DSDV | 580 | 934 | 914 |

utilise this information to improve routing. Mechanisms such as motion stability are also of little use if all motion is purely random, as a neighbour that may have previously been relatively stable could suddenly make a 'random' and unforeseen change. While the RWM is not intended as an accurate model of human mobility, the results are still useful as they enable GQPR to be observed in differing contexts.

#### 5.1.2. Delay

The results for delay in all RWM scenarios are presented in Table 2. In the 10 node scenario GQPR, DSDV and DSR all perform well while AODV incurs an unacceptable 2 s of delay. These results should however be considered in the context of the reliability results, and as both DSR and DSDV had less than 50% packet delivery it is hardly surprising that they experienced low levels of delay. GQPR's performance can be seen as a positive, but with reliability only 77.6% it comes at a price. GQPR again achieves the lowest level of delay for the 30 node scenario, with all other protocols exceed the informal limit of 300 ms. The performance by GQPR is particularly notable in comparison with AODV as GQPR achieves a slightly higher level of reliability, and a significantly higher lower level of delay. This suggests that GQPR is able to handle trade-offs between reliability and delay well when conditions are favourable. However the strong performance by GQPR in the 30 node scenario, must be considered alongside the 10 and 50 node scenarios where GQPR achieves very low levels of delay, but relatively poor reliability. This may be as a result of GQPR prioritising reduced delay over packet delivery and making routing decisions that lead to routable packets being dropped. When evaluating the results for the 50 node it is necessary to take into account that all other protocols performed poorly in this scenario as well, and that GQPR was the best performer in terms of both reliability and delay.

#### 5.1.3. Delay variation

Delay variation results are presented in Table 3. Comparing the results of delay variation with delay shows that while DSR and DSDV achieve good levels of delay, they experience a high level of delay variation; GQPR has only a minor variation and is the only protocol within the 10–50 ms window of acceptable jitter. In contrast, given that DSR and DSDV both had extremely low packet delivery levels, the large levels of delay variation experienced are likely a consequence of this. GQPR experiences a slight increase in the 50 node scenario, but still outperforms the other protocols. Although GQPR does not predict delay variation and does not explicitly try to manage it, GQPR achieves acceptable levels of jitter when the other protocols fail to do so. This is particularly interesting given the randomness of the RWM scenario is likely to create a continuously changing environment, that could be a potential source of a high jitter. This may be the reason that the other protocols struggle in this area, as a low level of delay is not a guarantee of low jitter. This would be a logical explanation for DSR and DSDV experiencing low delay but high delay variation in the 10 node scenario, given that the low delay was likely a result of frequent packet drops.

**Table 4**
Reliability for RPGM scenarios.

| Protocol | 10 nodes | 30 nodes | 50 nodes |
|---|---|---|---|
| GQPR | 99.9 | 97.1 | 99.3 |
| AODV | 99.9 | 98.2 | 87 |
| DSR | 85.5 | 80.3 | 76.8 |
| DSDV | 84.7 | 73.5 | 82.8 |

### 5.2. Reference point group mobility

#### 5.2.1. Reliability

From Table 4, it can be seen that GQPR performs significantly better for the RPGM scenarios than it does for RWM. The lowest reliability level experienced by GQPR is 97.1% in the 30 node scenario. Overall the reliability results are better here than for RWM, but GQPR is the only protocol to attain a minimum 90% reliability in all scenarios. AODV also performs strongly in these scenarios, and outperforms GQPR in the 30 node scenario by a small margin. DSR and DSDV both perform better than in the RWM scenarios, but fall short of 90% reception in all scenarios. A possible explanation for GQPR's improvements is that RPGM is a less random form of mobility than RWM and as a result GQPR is able to take advantage of the more predictable mobility. Therefore while all protocols experience an improvement in their results, and AODV competes closely, GQPR is able to gain an edge through its use of location-predictions. Similarly, as mobility is less disruptive nodes may remain together for longer periods of time, thus allowing GQPR to build a more accurate picture of its neighbours' context than it would in random RWM environment. Considering nodes in RPGM scenarios will often move in groups this is a plausible explanation. While responders in disaster recovery scenarios may not always be moving in groups, they are also unlikely to be moving randomly.

#### 5.2.2. Delay

The improvement in GQPR's reliability in the RPGM scenarios is even more remarkable when considered alongside its delay results shown in Table 5. GQPR has managed to not only retain low levels of delay despite increased packet reception, but in the 30 and 50 node scenarios achieves a lower level of delay than in the corresponding RWM scenarios. The general decrease in delay also holds true for the other protocols, with AODV and DSDV achieving the same result in the 10 node scenario, and DSDV only having 1ms more. GQPR does

**Table 5**
Delay for RPGM scenarios.

| Protocol | 10 nodes (ms) | 30 nodes (ms) | 50 nodes (ms) |
|---|---|---|---|
| GQPR | 1.8 | 5 | 3 |
| AODV | 1.8 | 133 | 59 |
| DSR | 1.9 | 32 | 894 |
| DSDV | 1.8 | 41 | 162 |

**Table 6**
Delay variation for GM scenarios.

| Protocol | 10 nodes | 30 nodes | 50 nodes (ms) |
|---|---|---|---|
| GQPR | 4 | 13 | 2.9 |
| AODV | 1176 | 1159 | 263 |
| DSR | 443 | 3141 | 1338 |
| DSDV | 68 | 919 | 111 |

however experience significantly better levels of delay in the 30 and 50 node scenarios. Contrasting the results of GQPR with AODV, it can be seen that while their packet delivery rates are almost the same, in the 30 and 50 node scenarios GQPR performs substantially better in terms of delay. Again, the role of the mobility model being simulated is likely to play a large role in GQPR's performance. While the delay predictions used by GQPR do not explicitly incorporate mobility, it can still have a significant effect. GQPR is still a geographic routing protocol and therefore places emphasis on physical location, mobility can therefore have a detrimental effect on delay by leading to a packet being forwarded through many hops due to mobility, therefore result in a high level of delay. As the improvement in both delay and reliability is experienced by the other protocols, that GQPR is able to achieve the overall best performance is further proof of its ability to make successful trade-offs between the competing demands of delay and reliability.

#### 5.2.3. Delay variation

In the RWM scenarios, GQPR managed to maintain a relatively low level of jitter regardless of delay or reliability. As Table 6 shows, in the RPGM simulations GQPR is again able to maintain this level of jitter. While the 4 ms for the 10 node scenario is slightly higher than 2.9 ms in RWM, both the 30 and 50 node scenarios are significant improvements. More stable mobility may be a factor here, however as the other protocols also benefit from improved jitter. However GQPR is still the only protocol to manage an acceptable level of delay variation in all scenarios. For GQPR there does not appear to be an obvious link between delay and delay variation, as delay decreases between 10 and 30 then again between 30 and 50, but for delay variation there is an increase followed by a decrease. In contrast, in the RWM scenarios there is a continuous rise in both delay and jitter as the number of nodes increase – although the increase is not proportional. That GQPR exhibits its lowest levels of delay at and delay variation at the 50 node scenario is interesting. Although the increased traffic may be expected to have a negative effect on delay, the increased number of nodes may provide more options for forwarding, thus enabling GQPR to make better decisions.

**Table 7**
Reliability for GM scenarios.

| Protocol | 10 nodes | 30 nodes | 50 nodes |
|---|---|---|---|
| GQPR | 93.3 | 88 | 98.2 |
| AODV | 98.5 | 80 | 97 |
| DSR | 52.6 | 74.3 | 93.8 |
| DSDV | 82.6 | 76.34 | 99 |

**Table 8**
Delay for GM scenarios.

| Protocol | 10 nodes (ms) | 30 nodes (ms) | 50 nodes (ms) |
|---|---|---|---|
| GQPR | 5.3 | 4.9 | 3.2 |
| AODV | 372 | 436 | 49 |
| DSR | 1112 | 1539 | 28 |
| DSDV | 8 | 361 | 25 |

### 5.3. Gauss–Markov model

#### 5.3.1. Reliability

While not as positive as the RPGM results, the GM results in Table 7 still show GQPR achieving strong levels of reliability in two out of three scenarios. In the 30 node scenario, GQPR does dip below the 90% level, but only by 2 percentage points – 8 percentage points more than the second best protocol AODV, although AODV does perform better in the 10 node scenario. Whether or not 88% reception is acceptable depends on the context and the degree of packet loss a user is willing to accept. Given the nature of GQPR's deployment – as an emergency system in an unstable environment – the users may be willing to tolerate some noticeable packet loss if they are still able to perform a consultation, however this cannot be guaranteed.

The GM model is interesting as it incorporates both random and 'memory' based mobility, unlike the RWM which is purely random. Thus while it does not purport to be a realistic model of human mobility, the GM allows for the simulation of scenarios with varied degrees of random and correlated mobility. Although GQPR does not perform continuous learning, through its use of the stable mobility metric alongside location predictions, it is possible that GQPR is able to adjust its behaviour to adapt to the GM model whereas doing so is infeasible for the RWM. The GM approach may also explain why results similar, but with some differences, to the RPGM are obtained. There is again a pattern of decrease followed by increase as seen in the RPGM scenarios, suggesting that this may be a feature of GQPR's handling of less random mobility.

#### 5.3.2. Delay

The results achieved for delay by GQPR are again low, and fall somewhere in the middle of GQPR's range of delay across scenarios. The most notable observation about these results is that they follow a neat pattern of decrease as the number of nodes increase, and that there is very little variance between the results from different numbers of nodes (see Table 8). There also appears to be no direct correlation between the level of delay and packet delivery rate, as the lowest level of delay is found in the 50 node scenario which also has the highest rate of reliability.

**Table 9**
Delay variation for GM scenarios.

| Protocol | 10 nodes (ms) | 30 nodes (ms) | 50 nodes (ms) |
|---|---|---|---|
| GQPR | 4 | 13 | 2.9 |
| AODV | 1176 | 1159 | 263 |
| DSR | 443 | 3141 | 1338 |
| DSDV | 68 | 919 | 111 |

### 5.3.3. Delay variation

Unlike the delay results, the jitter results shown in Table 9 does not share the same pattern of continual decrease found in the former, instead showing a pattern of increase and then decrease to a level lower than the first scenario. These results can also be seen as more consistent than the other scenarios with less of a difference between the smallest (2.9 ms) and highest (13 ms) than the RPGM and RWM scenarios. With regards to the other protocols, DSDV comes close to acceptable jitter levels in the 50 node scenario, but other than only GQPR is able to remain within the 10–50 ms range.

## 5.4. Evaluation summary

GQPR's performance can be considered on the whole as positive. Although GQPR did not always meet the 90% packet delivery criteria, the only mobility model where it completely failed to achieve this was RWM. In RPGM GQPR was able to attain at least 90% delivery in all scenarios, and only failed to do so in one GM scenario (where it achieved 88% packet delivery). In contrast, AODV achieved >90% reliability in one scenario of RWM, but failed to do so for one scenario in RPGM and GM. Given the random nature of RWM it is not surprising that GQPR performed poorly, as did all of the other protocols except AODV in the 10 node scenario. Real-life human mobility is seldom purely random, and while the RWM should not be discounted as a mobility model, it is also not representative of the way humans are liable to move in a disaster-recovery scenario. Even in a dynamic environment potentially containing various obstacles and hazards, humans are still likely to move in an organised fashion. Thus the RPGM and GM models should be seen as more representative of human mobility. While the GM contains some elements of random behaviour, it also includes memory-based mobility, therefore allow it to model for the possibility of random behaviour that can exist in human mobility. Except for the 30 node scenario, GQPR performs well in the GM simulations. As the RPGM is based on group (as well as individual) mobility, the results provided by it are interesting as GQPR not only consistently achieves its best packet delivery rates, but also comes close to 100% delivery in the 10 and 50 node scenarios. The results from the RPGM scenarios are particularly positive when considered alongside the delay results, with GQPR achieving its lowest level of delay in the 10 node scenario, and never rising above 5 ms of delay. While high packet reception may be expected to lead to higher levels of delay, GQPR is able to achieve delivery rates close to 100% and very small levels of delay. The mobility model may be a factor here, with GQPR being able to better predict neighbour locations and use this information for QoS predictions, and general geographic routing. That GQPR is able to achieve this balance also suggests that the trade-off it makes between the competing demands of reliability and delay are made successfully so as to allow the right balance that does not sacrifice low delay for high reception, or vice versa. As the other protocols all obtain high levels of delay and do not perform as well as GQPR in reliability for the RPGM scenarios, this further strengthens the case for GQPR's routing logic.

The delay results for GM and RWM are also favourable, although RWM's results need to be considered in the context of GQPR attaining low reliability. While GQPR is able to achieve a strong balance in the RPGM scenarios, the low delay in the RWM scenarios may simply be a result of the low packet delivery rates. However, as the other protocols attain higher levels of delay than GQPR even with similar or worse packet delivery rates, the low packet delivery is unlikely to be the main explanation for this. A possible explanation is that instead of low reliability causing low delay, GQPR has prioritised low delay over high reliability leading to packets being dropped either because they do not meet the QoS requirements or because they reach the local maximum. This is possibly a result of the unpredictable mobility causing miscalculations of QoS or rendering information on neighbour's out of date, even with steps taken to prevent this. As GQPR performs strongly in both reliability and delay for the GM scenarios, this suggests that GQPR may struggle with purely random motion but is able to adapt to some degree of randomness. The delay results for the GM scenarios are interesting as they show a continued level of decrease as the number of nodes increases, and also the highest level of reliability leads to the lowest level of delay. This would seem to validate the assertion that GQPR is able to handle some degree of randomness in mobility, and adapt its behaviour suitable, but will struggle when faced with pure random motion. Although GQPR does not explicitly predict delay variation, it still performs well in all scenarios. While there can sometimes be a tendency to focus on reliability and delay, delay variation is an important metric in streaming as large levels of delay variation can lead to difficulties in playback. Even if a buffering mechanism is utilised, frequent high variations in packet arrival times can make buffering visible to the user and in the context of a video-call create an awkward experience. As discussed earlier, there is no universal level of acceptable delay variation, although obviously the lower the better, but an informal level of between 10–50ms seems to be the limits of acceptability. GQPR manages to remain under the 50 ms in all scenarios, and under 10 ms in four. In the 30 and 50 node scenarios of RWM, GQPR obtains around 20 ms of jitter. This is comparatively high, but not as high as the 29 ms obtained in the 30 node scenario of RPGM. In fact, GQPR may have been expected to obtain an even higher level of jitter for the RWM scenarios given the nature of mobility. The low delay variation for the RWM scenarios may be a consequence of the overall low delay. Similarly, as low delay is exhibited in all mobility models, the generally low levels of delay variance are unsurprising. While low average delay does not always translate to low delay variance, these results would appear to indicate that GQPR contains few or no large spikes in delay. This is interesting given that GQPR is not only an ad-hoc routing protocol, but also uses per-hop forwarding mechanism that does not attempt to memorise routes or paths. In contrast, the other protocols struggle with high levels of delay variation, sometimes several seconds. Therefore

it appears that GQPR's low levels are due to its consistently low delay levels. These results therefore prove that GQPR is a viable protocol for distributing streaming multimedia in ad-hoc networks. While GQPR may have struggled to attain suitable packet delivery in the RWM scenarios, this is most likely due to the random motion which is not representative of real-human mobility. In the two other scenarios, GQPR performed as the best overall for reliability, coming close to 100% delivery in several scenarios, and only once failing to meet 90% delivery. In terms of delay, GQPR is clearly the best performer overall, being the only protocol to attain levels of delay under not only 300 ms, but also 100 ms in all scenarios. Similarly, GQPR is also the best performer for delay variation coming first for every scenario. Therefore while AODV may achieve similar (and sometimes better) results in terms of reliability, GQPR consistently outperforms it in terms of delay and delay variation, with AODV regularly exceeding the 300 ms delay barrier (in some cases going over 2 s). GQPR's strong performance in the RPGM and GM scenarios validates its use of location-predictions alongside other context factors to make routing decisions. Overall, despite operating over an ad-hoc network GQPR is often able to achieve levels of QoS required of infrastructure networks. This suggests that GQPR is well-suited for use in streaming video in telemedicine scenarios.

## 6. Conclusion

Telemedicine has proven to be useful in both replacing or augmenting existing medical services and developing new ones. By allowing medical professionals to communicate remotely either with each other or patients, telemedicine harnesses the potential of communication networks to help save and improve lives. There has been much debate about the potential applications of telemedicine to disaster recover scenarios such as earthquakes. In these situations, injured persons are often stranded without access to medical professionals. If first responders on the ground were able to communicate with medical personnel then they could be provided with instructions on how to handle the patient, and the patient could be monitored remotely. However communications infrastructure is a key component of any telemedicine system, and in such instances there is a significant chance that it will be non-existent or severely damaged. Ad-hoc networks provide the possibility of forming a network consisting only of the end-user devices with no infrastructure. Such a network can be created spontaneously and managed in a distributed manner. Almost any device equipped with a WiFi radio can take part in an ad-hoc network if it has the correct software. Ad-hoc networks have however largely been confined to novel research problems and most real-world deployments are of a military nature. In order to run a successful telemedicine service stringent QoS demands must be met by the network so as to achieve a suitable level of video/audio quality. Existing ad-hoc routing protocols typically prioritise packet delivery over QoS management and may therefore be unsuitable for handling QoS-sensitive traffic. The work here has explored the possibility of designing and developing a framework that is able to provide streaming multimedia over ad-hoc networks. GQP2PS aims to leverage location and mobility information, along with other context information to make QoS predictions that will allow for a suitable streaming quality to be achieved.

## References


[1] S. Armenia, et al., Transmission of VoIP traffic in multihop ad hoc IEEE 802.11 b networks: experimental results, in: Proceedings of the First International Conference on Wireless Internet, 2005, 2005, pp. 148–155.

[2] H.N. Chan, K.N. Van, G.N. Hoang, Characterizing Chord, Kelips and Tapestry algorithms in P2P streaming applications over wireless network, in: Proceedings of the Second International Conference on Communications and Electronics, 2008, 2008, pp. 126–131.

[3] Q. Chen, S. Kanhere, M. Hassan, K.-C. Lan, Adaptive position update in geographic routing, in: In 2006 IEEE international conference on communications, vol. 0, no. c, 2006, pp. 4046–4051.

[4] Y.K. Choong, Anonymizing Geographic Ad Hoc Routing for Preserving Location Privacy, in: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops, 2005, pp. 646–651. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1437239.

[5] Cisco Systems, 2006. Understanding Delay in Packet Voice Networks. Available at: http://www.cisco.com/en/US/tech/tk652/tk698/technologies_white_paper09186a00800a8993.shtml (accessed 30.01.14).

[6] Datavox, 2014. Frequently asked questions about voice over Internet protocol and associated terms. Available at: http://www.datavox.net/Products&Services/faqs.htm#packetloss-level (accessed 30.01.14).

[7] W. Feng, L. Zhang, J.M.H. Elmirghani, Energy saving geographic routing in ad hoc wireless networks, Communications, IET 6 (1) (2012) 116–124.

[8] G.G. Finn, Routing and Addressing Problems in Large Metropolitan-Scale Internetworks, University of Southern California ISI/RR-87-180, March 1987 http://www.isi.edu/div7/people/finn.home/routing_and_addressing_problems_in_large_metropolitan-scale_internetworks.BW.pdf (accessed 02.09.15)..

[9] A. Jamal, S. Hussain, A. Zafar, A.Z. Malik, Role of telemedicine during disaster: a case study, in: Proceedings of 9th International Conference on e-Health Networking, Application and Services, 2007, IEEE, 2007, June, pp. 261–263.

[10] B. Karp, H. Kung, GPSR: greedy perimeter stateless routing for wireless networks, in: Proceedings of the 6th annual international conference on Mobile computing and networking, ACM MOBICOM, ACM, 2000, pp. 243–254.

[11] E. Kranakis, H. Singh, J. Urrutia, Compass Routing on Geometric Networks, IN PROC, in: 11th Canadian Conference On Computational Geometry, BC, Canada, 1999, pp. 51–54.

[12] F. Kuhn, R. Wattenhofer, A. Zollinger, Worst-Case optimal and average-case efficient geometric ad-hoc routing, in: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing, New York, NY, USA, ACM, 2003, pp. 267–278.

[13] F. Kuhn, R. Wattenhofer, A. Zollinger, Worst-case optimal and average-case efficient geometric ad hoc routing, in: Proc. ACM MobiHoc, 2008, pp. 24–30.

[14] T. Leinmuller, et al., Influence of falsified position data on geographic ad-hoc routing, Secur. Priv. ad-hoc and Sens. Networks (2005) 102–112.

[15] J. Li, S.M. Shatz, Toward using node mobility to enhance Greedy-forwarding in geographic routing for mobile ad hoc networks, in: The international workshop on mobile device and urban sensing (MODUS 2008), St. Louis, MO, 2008, pp. 1–8.

[16] X. Liao, H. Jin, Y. Liu, L.M. Ni, D. Deng, AnySee: peer-to-peer live streaming, INFOCOM 25 (2006) 1–10.

[17] H.-I. Liu, I.-F. Wu, MeTree: a contribution and locality-aware P2P live streaming architecture, in: Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications, 2010, 2010, pp. 1136–1143.

[18] P.S. Prasad, P. Agrawal, Movement prediction in wireless networks using mobility traces, in: Proceedings of Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE, 2010, pp. 1–5.

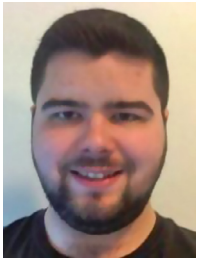
[19] M. Qin, R. Zimmermann, L.S. Liu, Supporting multimedia streaming between mobile peers with link availability prediction, in: Proceedings of the 13th annual ACM international conference on Multimedia - MULTIMEDIA '05, 2005, p. 956.

[20] R. Roine, A. Ohinmaa, D. Hailey, Assessing telemedicine: a systematic review of the literature, CMAJ: Canadian Medical Association Journal 165 (6) (2001) 765–771.

[21] S.H. Shah, K. Nahrstedt, Predictive location-based QoS routing in mobile ad hoc networks, in: Proceedings of IEEE International Conference on Communications, 2002. ICC 2002., 2002.

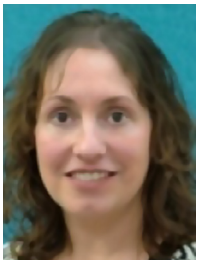
[22] I. Stojmenovic, M. Russell, B. Vukojevic, Depth first search and location based localized routing and QoS routing in wireless networks, in: Proceedings of International Conference on Parallel Processing, 2000, 2000.
[23] Stuedi, P., Alonso, G. (2012) VoIP for isolated and Internet-Connected mobile Ad hoc networks. Tech paper: http://www.mics.org/getDocum.pdf?docid=2332&docnum=1 (accessed 02.09.15).
[24] X. Tu, et al., Nearcast: A locality-aware P2P live streaming approach for distance education, ACM Trans. Internet Technol. 8 (2) (2008) p.2:1–2:23.
[25] A.H. Vo, G.B. Brooks, M. Bourdeau, R. Farr, B.G. Raimer, University of Texas Medical Branch telemedicine disaster response and recovery: lessons learned from hurricane Ike, TELEMEDICINE and e-HEALTH 16 (5) (2010) 627–633.

**Fraser Cadger** BSc (Hons), PhD is currently employed as a System Test Engineer at Hitachi Data Systems. Fraser's PhD project focussed on telemedicine in disaster recovery scenarios, and led to the development of routing and streaming protocols suitable for ad-hoc networks. In addition, Fraser also taught part-time as a technical instructor in subjects such as networking, operating systems, and object oriented programming.



**Kevin Curran** is a Reader in Computer Science and group leader for the Ambient Intelligence Research Group. Dr Curran has made significant contributions to advancing the knowledge of computer networking evidenced by over 600 published papers. He is a regular contributor to BBC radio & TV news in the UK and quoted in trade and consumer IT magazines on a regular basis. He is an IEEE Technical Expert for Security and a member of the EPSRC Peer Review College.

**Jose A. Santos** was born in Maracay, Venezuela in 1973. He received his Electronic Engineering Degree from the Electronics Department, Universidad Simon Bolivar, Caracas, Venezuela in 1998, and his PhD in Electronic Engineering from the School of Electrical and Mechanical Engineering, University of Ulster, Northern Ireland, UK in 2003. He has been a Lecturer in Computer Science at the School of Computing and Intelligent Systems of the University of Ulster since 2002, he is also a member of the Intelligent Systems Research Centre at the University of Ulster where he is part of the Ambient Intelligence Research Group. His current research interests lie in the fields of Ambient Intelligence & Mobile Computing, Wireless Communication Systems, Computational Intelligence, Biomedical Engineering, Sensor Technology and Robotics.



**Sandra Moffett** holds a BA (Hons) Business Studies, a MSc Computers in Education, a Post Graduate Certificate in Education and a PhD in Knowledge Management. Sandra is currently employed as a Lecturer of Computer Science with the University of Ulster's School of Computing and Intelligent Systems, Magee Campus. As a research active member of staff she is also a member of the Ulster Business School Research Institute and the Intelligent Systems Research Centre (Ambient Intelligence Group). Her expertise on Knowledge Management contributes to her being one of the UK leading authors in this field with over 50 publications in peer reviewed Journals and International Conferences. Sandra is founder of the MeCTIP model and associated 'Benchmarking KM' tool, and has received both internal and external funding to support her research investigating KM application within UK organisations.