General Title:

SmartSpace: Facilitating Resource Sharing In Urban Areas Through Biased Auctions

by

Paul Healy, B.Sc.

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Master of Science in Computer Science

University of Dublin, Trinity College

September 2011

Declaration

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

Paul Healy

August 31, 2011

Permission to Lend and/or Copy

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

Paul Healy

August 31, 2011

Acknowledgments

I would like to thank my supervisors Vinny Cahill and Mélanie Bouroche for their guidance and encouragement during this project.

Thank you to my family for their support and help during the project, especially to my parents and uncle Ger.

Paul Healy

University of Dublin, Trinity College

September 2011

General Title:

SmartSpace: Facilitating Resource Sharing In Urban Areas Through Biased Auctions

Paul Healy, M.Sc.

University of Dublin, Trinity College, 2011

Supervisor: Vinny Cahill/Mélanie Bouroche

The population of an area often shares some common interests due to their shared location. In some cases, people can help each other to perform tasks or achieve goals. This research facilitates resource sharing in urban areas.

This thesis presents a protocol suitable for conducting biased auctions over ad hoc networks for the purposes of resource sharing. The threats and challenges faced by such a protocol are presented as well as how they influenced its design. A real world example of an urban dweller making their parking space available to nearly drivers is employed. The protocol is implemented in a sample system called *SmartSpace*, which allows drivers to bid for the use of a parking space near their location.

The protocol allows for the auction to be biased using social networking data. Auctions conducted over the SmartSpace system are biased in favour of friends of the owner of the parking space. Yarta, a new middlware presented in Toninelli et al. (2011), is used to determine the parking space owner's friends by facilitating data sharing between social networking applications and the SmartSpace auctioneer application.

An evaluation of the protocol is presented by giving details of how identified threats are prevented or their effects mitigated.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

In many cases the population of an area share some common interests. For example, a crowd at a train station may share an interest in hearing about changes to the train timetable, or residents of an apartment complex may share an interest in discussing issues related to the building. In some cases, people can help each other to perform tasks or achieve goals. In the absence of other relationship ties, location can act as a motivator for social interaction. In the apartment complex scenario, fellow residents may be the only source of information which is specific to that apartment complex. The residents can help each other to achieve goals by information sharing and co-operation.

This collection of people is, therefore, an ad hoc location-based social network. Traditionally, we think of a social network as a virtual space where communication takes place irrespective of the physical location of the users. Often, a personal or professional relationship exists outside of the social network and this forms a large part of the motivation for communication. In a location-based social network, no assumption is made about pre-existing relationships between users. Users are only connected whilst they share the

same space.

The high level of ownership of smartphones (such as iPhone [1] and Android [2] devices) provides new opportunities for facilitating these location-based social interactions. Generally smartphones have the capability to form ad hoc networks using Bluetooth or other wireless technologies. Many provide access to services such as GPS and are Internet enabled. This allows for richer communication over ad hoc networks than has been previously possible. Users can take advantage of the ability of these devices to spontaneously form ad hoc networks as well as the access to services that they provide. Thus, users can co-operate to achieve tasks that would otherwise prove impossible. This allows for the development of a new generation of mobile social networking applications. The focus of these new applications is providing communication services over ad hoc networks with a view towards fostering sharing and co-operation amongst users.

Currently most social networking applications running on mobile devices manage their own data in a custom format and do not permit other applications to access and re-use the data. New mobile middleware such as Yarta proposed by Toninelli et al. (2011) provides methods of sharing information between these applications. For example, although no prior relationship is assumed in the location-based social networks described here, if relationships between participants do exist on other social networks it may be advantageous to use that information. For example, users may wish to discover if their friends are close by in order to arrange a meeting. An overview of recent development in mobile middleware is presented in Chapter 2.

Urban areas provide exciting possibilities for these location-based social networks. The high population densities in cities allow for dense, reliable ad hoc networks and also suggests a high demand for city resources. Many people own resources that they could make available to their neighbours when they are not in use. Auctions provide a suitable

[1] http://www.iphone.com
[2] http://developer.android.com

system for urban dwellers to share their resources with their neighbours. If the resource is a parking space, for example, it is difficult for the seller to determine its value. It may change from hour to hour or day to day depending on events occurring in the area. Auctions allow the bidders to determine the value of the resource which will increase whenever demand increases. This benefits the auctioneer as they always receive the maximum price a bidder is willing to pay for the resource. It is also beneficial for the bidders because it allows them to place their own personal value on the space. For example, the space may have a high value for a bidder running late for a meeting. They may be prepared to pay more for the space than the other bidders meaning they can maximise their chance of being allowed to park.

This research is motivated by the common desire for communication and co-operation and technological advances that facilitate these desires through the spontaneous creation of ad hoc networks using mobile devices.

1.2 Project Goals

The main goal of this research is to develop a system that allows people in urban areas to make resources available to others in their locality through auctions over ad hoc networks. The steps carried out to achieve this goal were:

1. The identification of the threats and challenges that such a system would face. These include threats from malicious users of the system and the challenges associated with communication over ad hoc networks.
2. The development of a protocol to address these threats and challenges.
3. The implementation of the the protocol in a sample application.
4. The development and implementation of an extension to the protocol which allowed for the use of social network data to bias the auction.

5. The evaluation of the final protocol in relation to the identified threats.

1.3 Contribution

This work presents a novel solution to resource sharing in urban communities. The final auction protocol presented is suitable for conducting auctions over ad hoc networks even where node mobility is high. The threats and challenges identified here may be of use to developers of other location-based social networking applications. This work is one of the first applications to be developed on the Yarta middleware and highlights a potential use for information sharing between applications, i.e. biased auctions.

1.4 Auctions Overview

This section presents a brief overview of the different auction formats relevant to this project, namely English auctions and Sealed-Bid auctions. Whilst many different variations of both exist each variation shares the same basic principles. An overview of auctions from a computer science perspective is presented by Parsons et al. (2011). The focus of this project is on one-sided, first-price auctions. One-sided refers to those bidding being either buyers or sellers. The job of the auctioneer is to determine a winner. An alternative to this is a two-sided auction in which buyers and sellers bid and the job of the auctioneer is to match one to the other. The first price aspect refers to the fact that the winning bidder pays the value of the highest bid.

1.4.1 English Auctions

The English auction is the type of auction with which people may be most familiar. A seller may set a minimum price that they will accept for the item being auctioned. Bids

are sent in an open-cry fashion. This means that all participants hear bids as they are being made. Each participant is therefore aware of the current winning bid. Bids are collected by an auctioneer. As each bidder is aware of the current winning bid and must bid higher in order to beat it bids are submitted in ascending order. As Parsons et al. (2011) notes, this is not necessarily a rule of the auction but an emergent property of the open-cry bidding process.

1.4.2 Sealed-Bid Auctions

Open-cry auctions are particularly susceptible to collusion. Such threats are discussed in Chapter 3. In order to prevent some types of collusion, bidders in sealed-bid auctions submit a secret bid to the auctioneer. During the bidding period the auctioneer should not know the value of the bids. After bidding has been completed, the auctioneer examines the bids and determines the winner. Sometimes sealed-bid auctions may be used to protect the privacy of bidders. The identity of the winning bidder or the value of the bid does not always have to be revealed to the bidders.

1.5 Document Structure

Chapter 2 presents the background to some technologies and concepts discussed in this thesis. The current state-of-the-art is presented in relation to auction protocols and middleware for mobile devices.

Chapter 3 discusses the design of the auction protocol in relation to the threats the protocol faces and the design of the sample application in which it is implemented.

Chapter 4 describes the implementation of the protocol in a sample application.

Chapter 5 presents an evaluation of the protocol by discussing how the identified threats are dealt with.

Conclusions drawn from this research are discussed in Chapter 6 with some suggested future work that could be carried out.

Chapter 2

State Of The Art

2.1 Introduction

2.2 Background

This section presents a brief overview of the cryptographic principles used in this research and opinions on location-based social networking.

2.2.1 Cryptography

Privacy is one of the main goals of cryptography. Diffie & Hellman (1976) defines privacy as the ability of parties to communicate over a public channel without the risk of others eavesdropping on their communication. Cryptography can also offer authentication, which allows participants to prove and confirm their identity. Symmetric cryptography involves encrypting some *plaintext* with a key to produce an encrypted *ciphertext*. The same key is used to encrypt and decrypt the data. If two parties wish to communicate securely over a network, agreement on a common key is required before such communication can take place.

Diffie & Hellman (1976) presents a *public key cryptosystem* to remove the need for agreement on a symmetric key prior to encrypted communication. To perform public key encryption a user generates two keys, a public key and a private key. The public key can be used by other users to encrypt data sent to that user. The user can decrypt the data with their private key. It is computationally infeasible to compute the private key from the public key so the user can make their public key available to all other users. Diffie & Hellman (1976) also shows how this public key cryptosystem can provide authentication of users. If a user wants to prove that they sent a message they can first encrypt it with their private key. Anybody who receives the message can decrypt it with the sender's public key. Provided the sender has not shared their private key with anybody else, only they could have encrypted the message. This is an example of what Diffie & Hellman (1976) refers to as "one-way authentication".

2.2.2 Location-Based Social Networks

Foth (2006) presents a study of location-based social networking in inner-city neighbourhoods. The residents of three different Australian apartment blocks were surveyed regarding their opinions and concerns relating to location-based social networking. The research identifies some interesting uses for, and attitudes towards, location-based social networking. One of the uses for such networks identified was discussion about collective issues, i.e. that affect all apartment residents. Residents surveyed in the study said they would use location-based social networks for discussion about changes occurring in the area. The research gives the example of residents discussing how the local council's proposal to build a new road nearby would affect them. The residents could organise a physical meeting to discuss the issue. However, conducting the discussion over a location-based social network may may encourage input from residents who would not normally be motivated to attend such meetings or may be unable to do so.

The location-based network also allows residents to find out information about their

local community. Parallels are drawn between this type of communication and posting a message on a noticeboard in the locality. The example is given of an apartment resident attempting to discover if Spanish lessons are available in the area. This information is likely to be known in the community and may not be known elsewhere or available on the Internet.

The study found that many residents are open to using location-based networks for making friends. Many of the interactions over such networks may take place by chance. However, these chance interactions can lead to bonds being formed. Residents may be more open to forming social ties with other network users as the fact that they live in the same location may mean that they are trusted more. If users did not live in the same apartment complex they may not necessarily trust the other network participants.

2.3 Ad Hoc Auctions

This section presents prior work that has been carried out in the area of auctions over ad hoc networks.

Fourati & Agha (2006) presents a protocol suitable for deploying auctions over ad hoc networks. Their research focuses on the challenges presented by the ad hoc network, namely ensuring each bidder has a fair chance to bid and preserving the integrity of bids. Their protocol influenced the development of the protocol presented in this thesis. It implements an English auction. However, it removes the auctioneer from the bidding process completely and distributes its duties amongst the bidders. Each bidder is responsible for collecting bids and determining the auction winner. This is primarily done to prevent security vulnerabilities caused by the seller acting dishonestly during the auction. The auction progresses as a series of rounds with the highest bidder during the last round being declared the winner of the auction. To begin the auction a seller broadcasts a message to the network containing details of the item for auction and details of the auction

such as the number of rounds and the minimum bid value that will be accepted. During each round the bidders both broadcast their bid to the network and collect bids sent by their fellow bidders. At the end of each round each bidder independently calculates the round winner and decides whether or not they will bid higher in the next round. After the last round a settlement takes place between the winner and the seller via unicast messages.

The methods used to ensure bid integrity and fairness during the auction mean that the protocol is not suitable for implementation in the type of urban auction that is the focus of this thesis. Bid integrity is achieved using public key cryptography and message signing. However, to prevent a compromised node revealing its private key the key is split up and shared amongst the node's trusted 1-hop neighbours before being destroyed. To bid a participant sends their bid to their trusted 1-hop neighbours encrypted with their respective public keys. Each neighbour decrypts the message, signs the contents with their portion of the private key and broadcasts the message to the network. Using the principle of (k, n) threshold cryptography, presented in Shamir (1979), the private key is split up and shared with n trusted 1-hop neighbours. Each neighbour uses their share to sign the message and broadcasts it to the other bidders. Bidders only have to receive the message k times in order to verify the signature of the message where each of the k messages received is signed with a different share of the key and k is less than n. This makes the assumption that a node can trust their 1-hop neighbours not to modify their bid. As outlined in Chapter 1, this thesis assumes no prior relationship between participants in location-based social networks and therefore, as bidders are always in competition, assumes that they do not trust each other and may act dishonestly given an opportunity to do so.

To implement fairness, the auction rounds for each bidder are shifted in time, depending on proximity to the seller and rounds are the same length for each bidder. The first round starts for each node as soon as the initial message containing the auction details

is received. Therefore rounds will begin for participants closer to the auctioneer sooner than for those further away. If rounds were not shifted in time and all rounds finished at the same time for each bidder the farthest away bidders may not have time to send their bids. This system provides each bidder with the same fair chance to bid but the system breaks down in auctions with high node mobility where the movement may lead to bids being received and processed during different rounds by bidders who have moved closer or farther away from the initiator.

Doghri (2008) also addresses the issue of auctions over ad hoc networks. This research also highlights the suitability of ad hoc networks, especially Mobile Ad Hoc Networks (MANETs), for conducting auctions due to the spontaneous manner in which these networks are established. As in this thesis the assumptions of Doghri (2008) are that bidders will be present at the start of the auction and that the auctions are limited in duration. However, in contrast to this thesis, node mobility is assumed to be low. This research again removes the auctioneer from the bidding process, allowing bidding to continue should the auctioneer be disconnected from the network. While this may be a desirable feature for auction in which the auctioneer is temporally disconnected, in the case where the auctioneer does not reconnect to the network the continuation of bidding is futile. This research does not provide security measures to prevent malicious behaviour by bidders, although this is identified as an area that warrants future research.

The main contribution of this research to the area of ad hoc mobile auctions is the development of a communication architecture, WHAS (Wireless Auction Handling System). The goal of this architecture is to separate communication services from the processing of messages. This allows auction applications implementing different types of auctions, potentially using different technologies, to use the same communication layer. This is accomplished by inserting a new layer called the "P2P-Auction" layer into the TCP stack. This layer provides communication services to the application. The research presents a protocol that uses the "P2P-Auction" layers running on each node to communicate.

The protocol allows the auction type to be specified during the setup phase providing a communication protocol framework for developers of entirely decentralised auctions (i.e. auctions without any central auctioneer) of various types.

Fourati & Agha (2006), discussed previously, attempts to calculate a fair round duration that would afford every bidder the same chance to bid irrespective of the distance bids have to travel. This value is calculated as follows:

$$T_{Fairness} = 0.982656 * MaximalHopsNumber \qquad \text{Fourati \& Agha (2006)}$$

Doghri (2008) find this period to be too large during their evaluation, with nodes being idle for much of each round. The research also presents evidence that the round duration time does not need to increase linearly in networks with more hops. Establishing a fair bidding period for auctions over ad hoc networks is obviously a complex problem given the unpredictable nature of the density and size of the network. Research presented by Fourati & Agha (2006) and Doghri (2008) suggest a move towards a dynamic period.

2.4 Recent Auction Protocols

Recently developed auction protocols have mainly focused on bidder privacy. Zheng et al. (2007) proposes two protocols for first price sealed bid auctions. One protocol provides complete privacy for the winning bidder, at the end of the auction the identity of the winner and the value of the winning bid are known only to the auctioneer. The second protocol allows the winner to be verified and so details of the winner are made available publicly. This paper clearly shows the trade-off between privacy in auctions and the ability to verify that the auctioneer has conducted the auction correctly. The main phases of the protocols are registration, bidding, opening and announcing. The most interesting aspect of the protocols is that the auctioneer publishes a list of potential bids that the bidders may make. The secrecy is provided by one way hash functions. Bidders make

a commitment to the seller to pay one particular bid value by hashing the value of the bid with a random number and sending it to the auctioneer signed with their private key. The bidder creates a new vector by running hash functions on the contents of the old one, splitting it into several parts and distributing the parts amongst the other bidders, keeping one part for themselves. Each bidder combines their part with those received from other bidders and sends the resulting vector to the auctioneer. The auctioneer can compute the highest bid from the vector and then find the commitment that was made by a bidder for that bid value. The auctioneer publishes the commitment so that each bidder can check if it is theirs. If full privacy is not desired the bidder is asked to send out the random number that they used to compute the original hash so that the other bidders can use it to determine the winner.

Parkes et al. (2006) presents a system for conducting sealed-bid auctions. Their protocol preserves bid secrecy and is used to conduct trustworthy auctions. The protocol ensures that no bidder receives any information relating to the bids that would present them with an advantage and also ensures non-repudiation of bids. The auction requires a space such as a website which can be updated by the auctioneer. The auctioneer uses the space to make public announcements and display information that can be used by other parties to verify the auction outcome. The auction also requires the participation of Notaries. Notaries are trusted parties such as accountants or law firms who act as witnesses to the auction. Encrypted bids are submitted to the auctioneer and the Notaries. After the bidding period has ended bidders make their encrypted bids known to the other bidders who verify that the bids are consistent with those posted by the auctioneer in the public space. The auctioneer can then decrypt the bids in private and choose a winner. The auctioneer can prove to the bidders that the winning bid was chosen correctly without revealing details of the bids. This proof is achieved using Paillier encryption Paillier (1999). Using Pallier encryption the cyphertext resulting from an operation on two pieces of ciphertext is a valid cyphertext that may have resulted from performing an operation

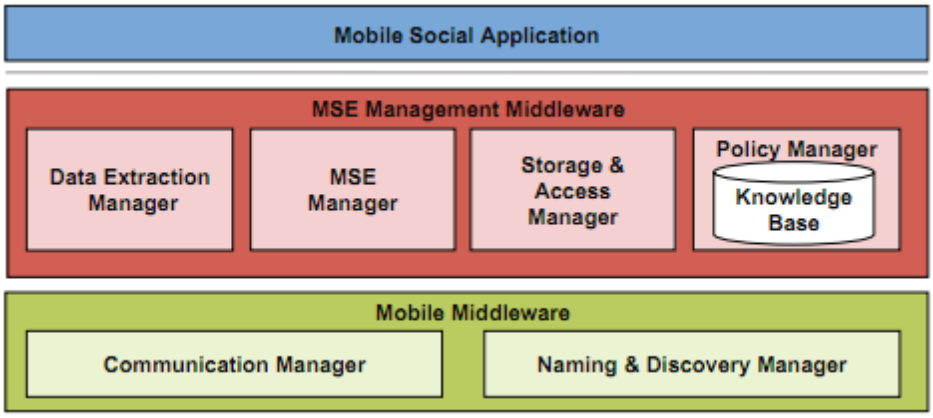

Figure 2.1: Yarta Architecture Toninelli et al. (2011)

on their corresponding plaintexts. This auction protocol required infrastructure such as the seller's website to be set up and the participation of trusted parties such as law firms or accountants and so is not suitable to conduct spontaneous auctions.

2.5 Middleware For Mobile Applications

Toninelli et al. (2011) aims to aid the development of mobile applications as described in Chapter 1 using Yarta. In this research such applications are referred to as *Mobile Social Applications* and the set of interactions that occurs between network participants as the *Mobile Social Ecosystem*. Figure 2.1 shows the architecture of the Yarta middleware. Data is stored in Yarta according to a representational model to enable sharing and re-use. The model itself re-uses existing ontologies such as Freind-of-a-Friend (FOAF) [1] and the Dublin Core [2] and is represented using the Resource Definition Framework (RDF). A key feature of the model is that it can be extended by application developers. Because all applications built on Yarta share a common data model, and any extensions to that model by individual applications can be shared and merged with with the recipient's model, applications can reason about the *meaning* of shared data. The *Knowledge Base*

[1] http://www.foaf-project.org
[2] http://www.dublincore.org/documents/dces

shown in Figure 2.1 exposes APIs to access and change data stored in the model according to the rules of the *Policy Manager*. Policies are represented as SPARQL queries and RDF statements. Yarta therefore hides the low level RDF details from developers and exposes a high level API to facilitate access to stored data. That API is automatically updated if changes are made to the underlying model. As Yarta is developed in Java it is highly portable with the ability to run on Android phones as well as personal computers.

Another middleware, *MobiClique*, is proposed by Pietiläinen et al. (2009). MobiClique allows communication over ad hoc enabled mobile devices. One of the features of MobiClique is that it leverages data from online social networks to allow communication in the real world, adding a new dimension to online friendships. When two users form an ad hoc network and their social network data matches (i.e. each user has the other listed as a friend) MobiClique allows communication to take place. At present MobiClique uses data from Facebook to determine details about the user such as who they know and their interests. MobiClique provides mobile device discovery. When a new node is discovered the devices exchange user identity information. During this exchange the users' social profiles are exchanged. The current design does not provide security. MobiClique provides the functionality for single users to communicate or for messages to be sent to a group of users who share a common interest. Similarly to Yarta an API is provided for application development using MobiClique. However, unlike Yarta the API does not allow the user to modify the data model only to change personal information and add or remove friends. MobiClique is developed using C++ and C# and is designed to run on the Windows Mobile platform.

Borcea et al. (2007) presents another middleware for developing social applications on mobile platforms. One of the differences between MobiSoC and the other middleware discussed here is that applications written to use MobiSoC must communicate with a server running the MobiSoC middleware over the Internet. Each mobile device uses a client to communicate with the server in order to preserve device battery. Similarly to

Yarta, MobiSoC has been developed in Java. MobiSoC attempts to learn information about users by monitoring the places that they go or the people that they meet. It learns information about places based on how many people are in the area and it can discover social groups by monitoring the common interests, friends or areas that the user attends. As this information is potentially private MobiSoC provides access control mechanisms when attempts are made to access social data.

Chapter 3

Design

3.1 Introduction

This chapter presents the steps taken during the development of a protocol which is suitable for auctions of smart city resources over ad hoc networks. The challenges of such a protocol are outlined and then the threats, faced by the protocol, are examined. Two initial designs, followed by the final design, are presented. Finally, a method of introducing bias into the auction is discussed.

3.2 Challenges

One of the main challenges in the development of an auction protocol for ad hoc networks is the issue of communication over the network. The dynamic topology means that it is highly likely that network messages will become lost and that nodes will not be in the same state. Therefore, bidders may disagree with decisions made by the auctioneer. Obviously, there will always competition between bidders in an auction. However, as the participants do not know each other, and may not even be able to see each other, the

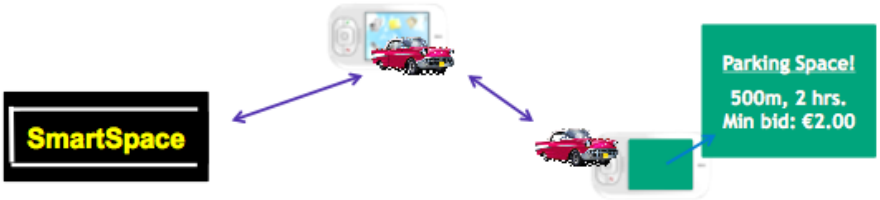

Figure 3.1: Space Availability Announcement

risk of malicious behavior on the part of auctioneers and bidders is high. This thesis focuses on one particular application of the protocol, so that the challenges and threats facing such systems may be clearly addressed as well as any application-specific issues. The protocol is implemented in a system called *SmartSpace*. It allows urban dwellers to make their parking spaces available to nearby road users. Such an application introduces the challenge of high node mobility which may exacerbate communication issues. Bidders participating in the auction have an immediate need for the space. Therefore, both an efficient auction and swift settlement are desirable.

3.3 SmartSpace Overview

This research envisages a situation whereby a city resident may own a network enabled smart parking space near his/her home. It is envisaged that drivers in the area, who are in need of parking, have the SmartSpace bidder application running on their phones. The parking spaces and mobile devices form an ad hoc network. As soon as the owner of the space pulls out, the parking space announces the availability of the space to the network, as shown in figure 3.1. This announcement contains details of the space such, as its latitude and longitude. Interested parties then bid for the use of the space. The value of their bid indicates how much they are willing to pay for the space per hour. The winning bidder may park in the parking space. As soon as they vacate the space and provided there is time before the expected arrival of the space's owner, the smart parking

space announces the availability of the space to the ad hoc network. The focus of this work is to develop a protocol suitable to conduct the auction.

3.4 Threat Analysis

The threats faced by a system such as this are varied. They can be classified as general threats, malicious attacks and advantage-motivated attacks. The ad hoc nature of the network presents threats to the auction, as discussed below. The smart parking space conduction the auction is referred to at the auctioneer.

3.4.1 General Threats

General threats are threats that exist outside of the technical field. For example, a driver may steal the space by parking in it without paying for it. A driver may not vacate the space when the period for which the owner has made it available for use by others has ended. Another example of a general threat would be a malicious party auctioning a space that is not theirs to earn revenue from it.

3.4.2 Malicious Attacks

Malicious attacks are attacks that are carried out in order to disrupt the running of the auction. These attacks generally do not provide any gain for the attacker, except in the case that a malicious owner may attempt to disrupt the running of another auction. Such attacks identified in this research are replay attacks and denial of service attacks.

Replay Attack 1

A malicious party listening to traffic on the ad hoc network may store a copy of a bid message. During a later auction for that space the malicious party may re-submit the bid by re-sending the bid message to the auctioneer. Should the bid win the auction the auctioneer will choose a bidder not even participating in the auction as the winner.

Replay Attack 2

This time a malicious party may store a copy of an auction announcement message send by an auctioneer. Once the auction has been conducted and the space has been taken the malicious party may re-send the message to the network causing bidders to attempt to participate in an auction for a space that is not available.

Denial Of Service Attack

A malicious bidder or auctioneer may attempt to disrupt the progression of an auction by generating large amounts of network traffic and attempting to take down the network by overloading nodes.

3.4.3 Advantage motivated Attacks

Advantage motivated attacks are attacks that take place during an auction by a malicious auctioneers or bidders in order to interfere with the fairness of the auction. A space owner's motivation to carry out such attacks is an attempt to increase the amount received for the space and a bidder's motivation is to win the auction, generally with the lowest value bid possible.

Bid Modification

As the auction is to be carried out over an ad hoc network, nodes will be required to forward bid messages from the other bidders. In a scenario whereby a bidder must send their bid to the auctioneer, via another bidder, that bidder may change the value of the bid contained in the message to a value lower than their own bid. Such an attack may results in auctioneer incorrectly choosing the malicious bidder as the winner.

Routing Attacks

Using routing attacks, a participant can use the fact that they are required to forward auction related traffic to gain an advantage. For example, a participant may refuse to forward a message announcing the auction in order to reduce the number of bidders participating in the auction. They may not forward bid messages leading the auctioneer to think that bidders have not sent bids. They may examine the contents of bid messages and refuse to forward any bids with a higher value than their own.

Bidder Impersonation Attack

In a situation where a SmartSpace only accepts the first or last bid from each bidder, a malicious bidder may impersonate other bidders and send in a series of low value bids at the start or towards the end of the auction. The attacker could then submit a slightly higher bid for themselves and win the auction.

Eavesdropping Attack

As bidders forward bid messages from other bidders they may examine the contents of the message and submit a bid higher than the bid value. Such a situation would give participants close to the auctioneer an advantage as they would potentially handle most

traffic from other bidders.

Collusion Attack 1

An auctioneer may plant a bidder or *stooge* amongst the bidders.The purpose of the stooge is to drive up the value of the parking space and increase the price other bidders have to pay to use it. This attack is potentially another impersonation attack in the event that the auctioneer pretends to be a bidder. Collusion is a problem in English auctions as the current winning bid is known by all bidders. Bidders may attempt to outbid a bidder submitting false bids.

Collusion Attack 2

In order to conduct a fair auction all bidders should know the same information and no information should be made available to a bidder that would give them an advantage during the bidding process. In a sealed-bid auction the value of the winning bid should not be determined until all bids have been submitted and the bidding process has ended. However, if an auctioneer was to examine the bids and inform a bidder of the highest bid they would have an opportunity to out-bid it. Such an attack benefits the auctioneer and the bidders.

Misinformation Attack 1

During an auction, for example an English auction, in which bidders send their bids to the auctioneer via unicast messages a auctioneer may announce that the current leading bid is higher than it really is, thus encouraging other bidders to bid higher to win the auction.

Misinformation Attack 2

An auctioneer may claim that there is more interest in the resource for auction than is actually the case. This may be achieved by making a false statement about the number of bidders participating in the auction. Bidders may be influenced to bid higher thinking that high demand for the resource will mean a higher bid is necessary to win the auction.

3.4.4 Design Focus

The design of this protocol focuses both on the advantage-motivated attacks and the malicious replay attacks. These are the attacks that affect the fair running of the auction. The general threats to the system are not covered as these can be dealt with through the administration of such a parking system. The denial of service attack is not addressed as it is an attack on the ad hoc network structure itself rather than a direct attack on the auction protocol. Methods to deal with denial of service attacks are discussed by Chen et al. (2006). Attacks involving participants, who refuse to forward auction traffic, are not addressed as they are unlikely have have much effect. Messages in dense ad hoc networks can generally take several routes to their destination.

Other issues which are taken into account during the design of the auction protocol include ensuring that close proximity to the auctioneer does not present an unfair advantage to bidders. The practice of *sniping* is discussed by Parsons et al. (2011). In an English auction there is little incentive for a bidder to bid early, as submitting a bid indicates interest to other bidders. Bids can be more effective if they are sent close to the end of the auction and are only slightly higher than the winning bid at the time the bid was sent. In an ad hoc network messages sent by bidders close to the auctioneer will reach their destination before bids sent by those further away. Bidders may attempt to take advantage of this situation and close bidders may send a bid just before the close of the bidding period at a time when a bid sent by a further away bidder would not reach the

auctioneer.

3.5 Protocol Design

This section describes the protocols that were developed over the course of this research. Lessons learned during the development of these protocols influenced the final protocol described in the next section.

3.5.1 Protocol A

This protocol is based on the protocol described by Fourati & Agha (2006), previously discussed in Chapter 2. That protocol achieved bid integrity by sharing a bidder's private key amongst their neighbours and sending bids directly to them encrypted with their respective public keys. Each neighbour decrypts the bid, encrypts it with their share of the bidder's private key and broadcasts the bid to all other nodes. Sharing the private key amongst a bidder's neighbours means that the bidders can destroy the private key to avoid revealing it in the case that it is compromised. That protocol is unsuitable for use in the SmartSpace system as it leaves bids open to modification by malicious neighbours who may change the value of the bid once they have decrypted it.

The main goal in the design of Protocol A was the prevention of bid modification. As identified in the threat analysis, the ad hoc nature of the network means that nodes must act as routers and forward auction traffic from other bidders. It is trivial for bids sent without any form of encryption to be modified. While there is a risk that a node may be compromised and have their key revealed, it is likely that bidders may be uncomfortable with sharing portions of their private key. Collusion and co-operation between a bidder's neighbours would allow the recovery of the key. Fourati & Agha (2006) assume low mode mobility. However, the high node mobility of the SmartSpace system means that it is

unlikely a node will share the same neighbours for the duration of the auction. For these reasons, the main difference between the original protocol and Protocol A is that bidders sign their own bids.

As with the original protocol, the auction consists of a number of rounds. During each round, each bidder both makes bids and collects bids from other bidders. There is no auctioneer. Bid authenticity and winner determination are carried out by each individual bidder. At the end of each round each participant calculates the winner of the round and may decide to bid higher in the next round. In order to give each bidder an equal chance of bidding, rounds begin for each bidder when they receive a message signaling the start of the auction. Therefore, rounds begin for bidders farthest away from the auctioneer last as they receive this message last. In order to deal with this inequality all rounds are of the same duration but shifted in time. A minimum round duration is required to ensure that the two farthest away bidders in the network can receive each others bid.

Protocol A proceeds as follows:

1. Auctioneer broadcasts an auction notification message to the ad hoc network.
2. Interested bidders reply with their public keys
3. Auctioner sends out a message to start the auction. The message contains the list of participant's public keys, the number of rounds, the duration of each round and the minimum bid that will be accepted.
4. Rounds start for each node when it received the start message. During rounds each bidder collects bids and may decide to bid themselves. Bid messages contain a public key to identify the bidder and each bid message is signed with the bidder's private key so that its authenticity can be verified.
5. At the end of each round each bidder determines the round winner and may decide to bid higher in the next round.

6. At the end of the auction the winning bidder contacts the auctioneer to initiate a settlement.

Whilst Protocol A does remove the requirement for bidders' to share their private keys it is not suitable for the SmartSpace system. High node mobility means that bids may frequently be received outside of the rounds during which they were sent.

Messages may get lost during the bidding rounds leading to nodes making incorrect decisions on who won. A node that does not receive the highest bid during the second last round will make an incorrect decision on the current highest bid and may bid lower in the next round than they may otherwise have done.

As the winning bidder contacts the auctioneer once the auction is over to initiate the settlement, a bidder may claim that they sent a bid higher than the real winning bid but that the auctioneer did not receive the their bid. The auctioneer, in order to maximise profits, may accept the higher bid of the malicious party.

3.5.2 Protocol B

Protocol A therefore is still affected by participant dishonesty and communication challenges. These challenges make it difficult for the participants to reach any agreement on which bids are valid. Each is motivated to claim that their bid is the winning bid. Protocol A removes any auctioneer from the selling process as does the Fourati & Agha (2006) protocol as they cannot be trusted to conduct the auction fairly. Protocol B puts the auctioneer back into the bidding process with a design that aims to prevent dishonest behaviour by implementing a sealed-bid auction. The role of the auctioneer is to aid agreement between nodes as to which bids are valid.

To begin the auction, the auctioneer broadcasts the announcement of the auction to the network. Interested parties reply with their public keys. To begin the auction the auctioneer broadcasts a message containing a list of the participants' public keys. Bidding

is allowed to take place until the auctioneer informs bidders to stop bidding. As this is a sealed-bid auction, the concept of fairness changes slightly. It is no longer imperative that each bidder has exactly the same time to submit bids. It is only necessary that each bidder has enough time to submit one single secret bid. The order in which bids are received is no longer important. The temptation for a bidder to wait to bid until just before the close of bidding to submit their bid is removed as it does not provide them with any advantage. During the bidding period each bidder encrypts their bid, signs it and sends it to all of the participants. Each bidder still collects all bids as does the auctioneer. At the end of the bidding period the auctioneer sends out a message to bidders to stop bidding.

After the bidding phase, the auctioneer attempts to reach an agreement between themselves and the bidders as to which bids are valid bids in a synchronisation phase. All bids remain encrypted. At the end of the synchronisation phase, all bids that are known by the auctioneer will be considered valid. The goal is to allow bidders to discover any bid messages they did not receive and to allow the auctioneer to do the same. To begin the synchronisation period, the auctioneer broadcasts a message to the bidders containing a list of the bids that they have received so far. These are the bids that the auctioneer currently believes to be valid. If a bidder agrees with the list of bids they take no action. A bidder may have stored a bid from a fellow bidder that was not received by the auctioneer and therefore is not on the list. In this case the bidder may decide to forward that bid to the auctioneer. If several bidders forward that bid to the auctioneer the auctioneer may decide to accept the bid. To end the synchronisation period the auctioneer sends out a second synchronisation message containing the definitive list of bids. This list may include bids newly accepted by the auctioneer. Any bidder who does not have a bid contained in the list should store a copy as it is a bid the auctioneer considers to be valid. The synchronisation period allows for co-operation between bidders. A bidder may forward another bidder's bid to the auctioneer in the hopes that another bidder will forward their

bid if it is not known by the auctioneer. It also allows a group of bidders disconnected from the main network to share their bids amongst themselves and for each to petition the auctioneer to accept the others bids upon reconnection.

The second synchronisation message triggers bidders to share their bid decryption keys with the auction participants. Each participant can then make an independent decision on the winner. The winning bidder contacts the auctioneer to perform the settlement. Once the auctioneer receives the settlement message it broadcasts an accept message to all of the bidders. Each bidder can then verify that the auctioneer has accepted as a winner a participant who submitted their bid during the bidding phase and can confirm the value of the winning bid.

Protocol B minimises the threats posed by malicious behavior on the part of the auctioneer. As bids are encrypted when they are received by the auctioneer, there is no opportunity for them to perform any attacks based on the value of bids. There is no incentive for them to attempt to influence bidder behaviour as the auctioneer has less information about the bids than the bidders themselves. For example, if a auctioneer posed as a bidder or planted a stooge to submit high bids the value of any bids made would only be revealed after the bidding process had been completed.

However, the synchronisation portion of the protocol is unrealistic and would not work in practice. Firstly, communication issues mean that any attempt at a distributed agreement between nodes is very difficult. Secondly, although they may benefit from the system themevles it is unlikely that nodes will forward bids that are not their own to the auctioneer. The auction requires a protocol that will allow nodes to make best effort decisions in the case that they have not received all of the intended auction messages.

3.5.3 Final Protocol

The final protocol presented combines the security mechanism of Protocol B coupled with a new network architecture to address the communication challenges faced by Protocol B. The final protocol uses aspects of centralised and de-centralised architectures to achieve this goal. Once again this protocol implements a sealed-bid auction.

The auction begins with the *announcement phase*. The auctioneer announces the auction and receives public key of interested bidders. To start the auction the auctioneer sends out a message containing a list of bidders' public keys. This begins the *bidding phase*. This time only the auctioneer collects bids during the bidding phase in order to dispense with a need for agreement amongst several nodes. Centralising the bidding process raises the issue of proximity to the auctioneer presenting an advantage. In a centralised English auction over an ad hoc network it may be possible for the nodes closest to the auctioneer to snipe bids (as described above) by taking advantage of the short time it takes for their messages to reach the auctioneer. However, this thesis regards the only requirement for fairness in sealed-bid auctions to be a chance to submit one secret bid. All bidders know the level of interest in the auction as the list of bidders public keys is sent at the start of the auction. Bidders therefore close to the auctioneer gain nothing from counting the number of bids as they route them and making their decision on the value to bid based on the level of interest in the auction as those bids were made under the assumption that all bidders on the bidder list would bid.

As soon as the auctioneer receives bid messages from all registered bidders the bidding process is ended in order to conduct the auction swiftly. The issue of determining a fair time for bidding was addressed by Fourati & Agha (2006) and is discussed in Chapter 2. As well as being a network with high node mobility it is likely that the density of the ad hoc network will change during the auction as nodes join or leave the network. This makes determining a duration for the bidding period difficult and a dynamic period would

appear to be the answer to this issue. Whatever the duration of the bidding period, the auctioneer must continue with the next phase of the auction at its termination even if all expected bids have not been received. The bidders are in need of a parking space and so expect an efficient auction.

To finish the bidding phase the auctioneer sends out the list of encrypted bids received and moves the auction into the *decryption and settlement phase*. The bids contained in this list are the only valid bids. Receiving this message causes bidders to broadcast their bid decryption keys to the network. Bidders use these keys to decrypt the bids and determine the position of their bid. Any bidder who determines that their bid is a high bid can send a message to the auctioneer claiming the space. The auctioneer listens for these claim messages and accepts the highest claimant. The decryption and settlement phase is designed to be robust and practical. In the event that a bidder does not receive all of the keys during the decryption period, they make a best effort decision on the position of their bid and decide whether or not to attempt to claim the space accordingly.

It is in the interests of any bidder who believes that their bid has ranked highly to claim the space as the bidder who submitted higher bids may have become disconnected and may be unable to claim the space. If the auctioneer does not receive all decryption keys they may be tempted to postpone choosing the winning claimant in the hopes that they receive the missing key and that it decrypts a higher bid. However this is not in the best interests of the bidders who favour a swift auction. Pressure is put on the auctioneer to choose a winning claimant quickly before bidders leave the auction or area in search of another space.

After choosing a winner the auctioneer broadcasts details of the winning claimant to all bidders. In the event that the auctioneer has made a best effort decision on the winner of the auction (because they have not received all decryption keys or did not receive all of the claim messages they were expecting) it is possible that a bidder will disagree with the auctioneer's decision as they have sent a higher bid. In this situation the bidder has lost

out on the space, possibly due to becoming disconnected from the network for a period. However the protocol has proceeded with best effort decisions. Should the bidder who submitted a higher bid claim the space after the auctioneer has chosen a winner, the auctioneer is unable to accept the claim as they have already sent a signed commitment to all bidders to give the space to another bidder.

3.5.4 Final Protocol Messages

The protocol requires 8 main messages. All messages carry the auction identifier distributed by the auctioneer. These are:

- *Announce messages.* These messages are used to inform potential bidders than an auction is about to start. Announce messages carry an auction identifier, the current time, a time to live, the latitude and longitude of the space.
- *Join messages.* These messages are sent by bidders who wish to join the auction. These messages contain the public key of the bidder for identification purposes.
- *Start messages.* These messages contain a list of the public keys of registered bidders. They signal the start of the bidding phase.
- *Bid messages.* These messages contain the public key of the bidder for identification purposes and the bidder's encrypted bid.
- *Bids Received messages.*These messages signal the end of the bidding phase. They contain a list of the encrypted bids that the auctioneer received together with the public keys of each bidder for identification purposes.
- *Share Key messages.* These messages contain the public key of the sender for identification purposes and the symmetric key that can be used to decrypt the bidder's encrypted bid.

- *Claim messages.*These messages are send by bidders to the auctioneer in an attempt to be selected as the auction winner. They contain the bidder's public key for identification, the symmetric key that can be used to decrypt the bidder's bid in case it was not received by the auctioneer and the value the bidder's bid.

- *Accept messages.* These messages are sent from the auctioneer to all bidders when the auctioneer determines which bid should be accepted. They contain the value of the winning bid, the key that can be used to decrypt the winning bid and the public key of the winner. Bidders can use this information to check the decision making of the auctioneer.

All of the above messages are sent with a signed cryptographic hash of the message. As vehicular ad hoc networks may potentially span a large geographical area, measures are in place to prevent broadcast messages related to the auction from travelling beyond the area where the auction is taking place. Limiting the distance that broadcast messages may travel by the number of hops that they can make in the network is unsatisfactory as when more nodes become part of the network, causing the network to become denser, messages may not reach those in outlying areas. Instead broadcast messages are limited by the physical distance they may travel. Every sender of a broadcast message sets the distance that the message my travel in kilometers and their current latitudinal and longitudinal position when the message is sent. Nodes forwarding the broadcast message first check their distance from the sender. If they are within the range specified by the sender they continue the broadcast by forwarding the node to their neighbours. Announcement broadcast messages also contain the time at which the message was sent, although the original idea of including this time was to prevent the identified replay attack. Nodes can also use this time to stop the flooding of a message if it is no longer within its time to live.

As stated, the auction protocol has been designed to be robust in the face of disruptions to network communication. For example, if an auctioneer does not receive all of the keys

it was expecting during the decryption phase it will still make a best effort decision on a winner. For this reason broadcast messages are sent several times by each sender and the main unicast messages are acknowledged.

3.6 Including Bias

3.6.1 Motivation

In auctions of resources conducted by city dwellers, there may be advantages in biasing the bidding towards a group of people, friends of the auctioneer for example. This has obvious advantages for the friends as they have a greater chance of winning the auction but there are advantages for the auctioneer too:

- The bias may encourage friends to participate in the auction.
- The friends may conduct auctions of resources themselves and a reciprocal arrangement may be in place.
- The auctioneer may trust their friends more to pay or vacate the space on time.

3.6.2 Overview

In order for the auction to be biased in favour of a bidder, the auctioneer must firstly accept that the bidder is a friend and then ensure that the bidder is who they claim to be and not someone attempting to impersonate the friend. For the purposes of this extension to the protocol we assume that the auctioneer knows the public keys of their friends. In order to authenticate themselves the friends must therefore prove to the auctioneer that they know the corresponding private key. Biasing the auction towards a bidder is done by increasing the value of their bid without increasing the amount the bidder has to pay if he or she wins. Auctioneers offer bias as a percentage of the bidder's bid. For example,

if an auctioneer offers a 10% bias towards friends a bid of 2.00 euro is treated by the auctioneer as a bid of 2.20 euro although the price the bidder pays should he or she win the auction remains at 2.00 euro.

3.6.3 Updated Protocol

In order to establish that a bidder knows a particular private key the auctioneer issues them with a challenge. The bidder signs the challenge with their private key, and sends it back to the auctioneer. The auctioneer can confirm that the challenge was signed by the correct private key by decrypting it with the bidder's public key and comparing the result to original challenge.

The challenge/response exchange takes place directly after the auction has been announced and before the bidder has joined the auction. This informs the bidder, when choosing which auction to join, which auctioneers regard them as a friend and therefore which auctions are biased in their favour.

Attacks on the challenge response that affect the running of the protocol are difficult given that the period of time between the announcement of an auction and the beginning of the auction is quite short. It is good practice for the challenger to issue random challenges.

Including bias in the auction requires changes to the announcement phase of the protocol. The new announcement phase proceeds as follows:

1. The auctioneer sends an Announce message to the bidders.
2. A bidder replies with a request for a challenge.
3. The auctioneer sends a signed challenge message to the bidder.
4. The bidder signs the challenge contained in the message and sends it back to the auctioneer.

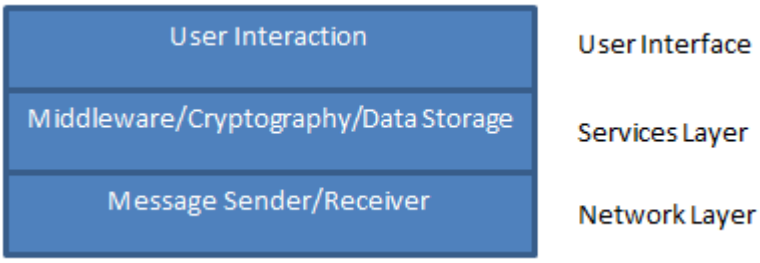

Figure 3.2: Application Architecture

5. The auctioneer accepts or rejects the response from the bidder.

3.7 SmartSpace System Design

The SmartSpace system is currently composed of two mobile applications, a bidder application and an auctioneer application for use by bidders. This section gives a brief overview of the software design of both components. The auctioneer mobile application provides the functionality that the smart parking space would provide in a real world deployment of the system.

3.7.1 Overview

The bidder application and auctioneer application share the same basic architecture. This architecture is presented in figure 3.2. At present, both applications are designed as mobile applications to show clearly how both the bidder and the auctioneer can utilise the Yarta middleware for including bias in the auction. The Yarta data model allows us to store information about the relationship between people and things. For example, a social networking application that allows a user to communicate with their friends may store friendship information in Yarta. The Yarta representation of the auctioneer is linked to the Yarta representation of each friend by the *knows* relationship in the Yarta data model. The SmartSpace auctioneer application accesses the Yarta knowledge base to

establish who is known by the space owner and uses this information to bias the auction in favour of those people. The bidders access the Yarta knowledge base to extract the Yarta representation of themselves and make it available to the auctioneer.

The Network Layer handles all network communication for both applications. It sends and receives the broadcast and unicast auction messages that are defined above. Messages, sent between participants of the auction, are contained in *envelopes*. An envelope consists of an identifier describing the type of message it contains, the auction message itself and also the signed cryptographic hash of the message, if required.

The next layer consists of the application middleware, cryptography service and application temporary data storage. The cryptography service stores the public/private key pair and also generates a new symmetric key for use during each auction. It also provides encryption, decryption, hashing and signing services to the application. For demonstration purposes, the auctioneer application currently contains a user interface showing the state of the auction and allows a space owner to start auctions and accept claim messages from bidders. A real world deployment of the SmartSpace system would require the parking space to conduct the auction itself, without user input, as it will be run in the absence of the space owner. In this case the SmartSpace auction application could synchronise friendship or other bias related information when the space owner's smartphone is in range. The protocol rules are defined in the middleware. These rules define which actions should be taken when an application message is received or actions are performed by the user. The data storage area stores auction and bidder information received and personal user information such as friend details, bid value and current user location.

The user interface allows input from the users and displays information about the current state of the auction. The user interface of the auctioneer application allows the auctioneer to set a minimum price for the space and start the auction. It updates as bidders register, bids are received and decrypted and claim messages are received. It allows the auctioneer to accept a claim. The bidder application displays auction announcement details as they

are received and allows the bidder to join an auction. It then allows the bidder to enter and submit their bid and displays information about other bidders' bids when they are received. It also notifies the bidder of the position of their bid as bids are decrypted. Finally, it allows the bidder to submit Claim messages and displays the result of the auction.

Chapter 4

Implementation

At present both the SmartSpace auctioneer application and bidder application are implemented as Android applications. This chapter presents an overview of Android applications and then specifically an overview of the SmartSpace applications. Implementation details are then presented in more detail.

4.1 Overview of Android Applications

Android is a popular operating system for smart phone devices. Android is a Java based operating system and so applications written for it are written in Java. The main components of an Android application used in this research are *Activities* and *Services*.

An Activity consists of a single user interface that presents some functionality to the user. For example in the case of the SmartSpace bidder application one Activity provides the user with the list of available auctions and allows them to click on one to join the auction. A collection of Activities can interact the provide all of the user interface related functionality in an application however each Activity runs independently of any others. For example, when a bidder selects which auction they wish to join another Activity starts

to display the location of the space in relation to their current position.

In order for the application to know the current position of the bidder a *Service* is used. A service performs long running tasks and runs in the background. A service does not provide a user interface to the user and so a user is often un-aware of their presence. Activities start other Activities or Services using using *Intent* objects. Examples of this are given below.

Android applications contain a *manifest file*. The purpose of this is to inform the operating system of the application's components. The manifest file contains a list of all of the activities, services and other components in the application. It also specifies which Activity should be launched when the application is launched, identifies any permissions required by the application as well as other application details such as the hardware and software features needed by the application.

Finally, the user interface layout for each Activity is defined in a individual XML file. This separates the application code from the design of the application. Below details of the Activities, Services, XML layouts and other components are presented in relation to the SmartSpace applications.

4.2 Overview of Implementation

This section presents a brief overview of the SmartSpace application. The project was developed in Eclipse [1]. The project consists of the Android components described above as well as standard Java classes. The Android SDK [2] provides an emulator to aid in the development of applications. This emulator simulates the application running on an actual mobile device and was used to develop and test the applications. Several different emulators running on the same computer were used to simulate a sample ad

[1] http://www.eclipse.org
[2] http://developer.android.com/sdk/

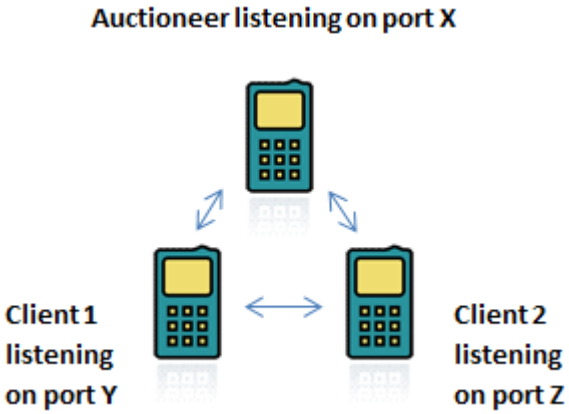

Figure 4.1: Sample Network Architecture

hoc network. An example of the network architecture is shown in figure 4.1. Due to limitations of the emulators, each instance of the emulator that requires the capability to receive network packets from any other must listen for messages on a unique port number. For this reason in the current implementation of the SmartSpace system each participant listens for messages on a different port and the network layer has been designed and implemented according to these restrictions. As the network is not a true representation it has not been possible to implement the geographic restriction on broadcasts messages however the groundwork for such an implementation has been put in place and is discussed below.

4.3 Implementation Description

4.3.1 Protocol Messages

We firstly examine the implementation of the auction messages defined in the protocol design. Each protocol messages is implemented as a Java class. Each messages class extends the *AuctionMessage* class in order to re-use core message functionality. An example of message class inheritance is presented in figure 4.2 and the functionality provided by AuctionMessage.java is shown in figure 4.3. Notice two constructors are shown, one for the

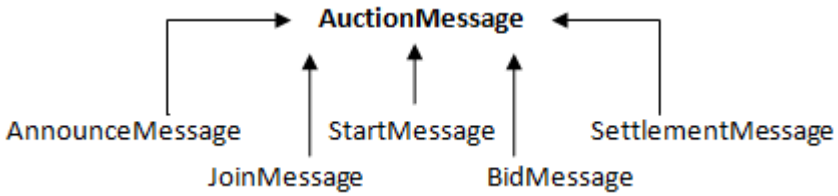

Figure 4.2: Example of message inheritance

```
//CONSTRUCTORS
public AuctionMessage(int type, long id){
public AuctionMessage(int type, double lat, double lon, double distance, long id){
//PUBLIC METHODS
public int getType(){
public double getLatitude(){
public double getLongitude(){
public double getDistance(){
public long getID(){
```

Figure 4.3: Core Auction Message Functionality

creation of broadcasts messages containing information about the latitude and longitude of the sender and the other for unicast messages without that information. All message classes implement the Java Serializable class as they must be serialized and stored in a byte array before being sent over the network. The AuctionMessage class provides functions to extract information common to all messages, the message type, and also functions to support basic operations on broadcast auction messages such as to extract the latitude and longitude of the message sender.

Some messages carry auction specific Java objects across the network. For example the *Auction* and *Bidder* classes describe auctions and bidders respectively. When a bidder sends a JoinMesage to join the auction that message contains a Bidder object that represents themselves. It contains fields for the bidder's name, public key, encrypted bid, decrypted bid and a flag indicating their bid status (e.g. no bid, bid, decrypted bid). Before sending the message they set their name, public key and initialise their bid value to be 0.00 euro. When the auctioneer application accepts the join message they store a copy of the bidder object contained in it in their list of bidders. As encrypted bids and keys are received they are stored in the appropriate bidder object. When bids are decrypted the

decrypted value is stored in the bidder object too to facilitate ordering bidders according bid value.

Messages are stored in Envelope objects before they are send across the network. The Envelope class stores the AuctionMessage object being sent across the network, the signed cryptographic hash of the message as a byte array and an identifier indicating the type of message the envelope contains. AuctionMessage objects are placed in Envelope objects at the middleware layer and sent to the network layer. When they are received they are passed from the network layer to the middleware layer where the message is verified. The Constants class stores integers that are used to identify different types of message.

4.3.2 Network Layer

The network layer of both the auctioneer and bidder applications is comprised of three classes, the *Sender*, *Listener*, and *ListenerThread*. The Listener class runs in its own thread. It listens for User Datagram Protocol (UDP) packets and upon receipt of a packet starts a new thread (an instance of ListenerThread) to process the packet contents as in figure 4.4. The new thread extracts the contents of the packet and reforms the Envelope object. The thread then passes the the Envelope object to the services layer.

The function of the Sender class is to send UDP packets. Envelope objects are passed to the Sender class from Middleware. The Sender class writes the Envelope object to a byte array which it places in an UDP packet and sends. Due to the limitations of the Android emulator mentioned earlier sending a broadcast message in the current implementation involves sending several unicast messages to receivers listening on different ports. The network layer is created by Services Layer when the application is started.

```
while(listening){
    DatagramPacket packet = new DatagramPacket(buffer,buffer.length);
    try {
        socket.receive(packet);
        Runnable r = new ListenerThread(middleware, packet);
        new Thread(r).start();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Figure 4.4: Receiving a UDP Packet

4.3.3 Services Layer

The services layer of both applications consists of middleware, cryptography and data storage. The *Cryptography* class provides the symmetric and asymmetric cryptography functionality for the application. In both applications it holds the user's public and private keys and in the case of the bidder application it generates and stores a new symmetric key when the bidder joins a new auction. The *Middleware* class provides a link between the Network Layer, data storage class and user interface Activities. It utilises the functionality of the Cryptography class. When a signed message is received the Middleware class uses the Cryptography class to create a SHA-1 hash of the message using the method *getHashOfObject*. It then uses the Cryptography class to decrypt the signed hash sent with the message with the public key of the sender using the method *decryptHashWithPublicKey*. It then compares these two hash values and if they are equal processes the message. In order to decrypt the hash of the message with a public key the firstly gets an instance of the cipher to be used for encryption and initialses it to be used in decrypt mode with the appropriate public key. The *doFinal* method performs the decryption and returns the decrypted byte array. In order to compute the hash of the message to compare it to the resulting decryption the getHashOfObject method firstly converts the message to a byte array, and then initialises a *MessageDigest* object to use the *SHA-1* hash algorithm and computes a hash of the object. In order to sign a hash with application's private key similar steps to the decryptHashWithPublicKey method

are followed however the cipher is initialised in encrypt mode with the sender private key.

As the Network Layer is multi-threaded and passes Envelope objects to the Middleware each method in the Middleware is synchronised to prevent multi-threading issues such as race conditions. The auctioneer application uses threads to end each phase of the auction and begin the next. For example once the auctioneer begins to listen for Join messages the *AuctionJoinPeriodThread* is started which will wait the required join duration and then trigger the auctioneer application to stop listening for Join messages and start listening for bid messages using the method call *setListeningForJoinMessages(false)*. This method itself starts the *AuctionBidPeriodThread* which runs in a similar fashion. The data storage class stores personal information in the case of both the bidders and the auctioneer. In both cases it stores a Java object representing the current auction. In the case of the bidders it contains lists of Auction objects that have been received in Announce messages and also a list of Bidder objects representing other bidder in an auction. The Middleware implements the rules of the auction protocol.

For example, when the bidder application receives an announce message the Network Layer invokes the *receiveAnnounceMessage* method of the Middleware class. In an effort to enforce the rules regarding the geographical distance over which a broadcast message is valid the method establishes the distance from its current location to the location of the origin of the message. If the distance is too large the message is ignored. If the announce message is accepted as valid the Auction object is extracted to from the message and added to the list of objects in the data storage area. Adding to the list of auctions causes the user interface to be updated via message handler.

Receiving an announcement message automatically causes the bidder to send a request for a challenge. If the public key of the bidder requesting the challenge is known by the auctioneer application a challenge will be sent otherwise the request for a challenge will be ignored. If the public key is known to the auctioneer the bidder is regarded as a friend.

```
public void setNewSyncKey(){
    KeyGenerator keyGen = null;
    try {
        keyGen = KeyGenerator.getInstance("AES");
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    keyGen.init(128);
    bidEncriptionKey = keyGen.generateKey();
}
```

Figure 4.5: Generating a New Symmetric Key

The list of public keys of those known to the auctioneer is populated by the calls to the Yarta middleware (described later). If a challenge is received by the bidder application the Middleware class signs it and sends back a signed response message. If the response is accepted the auctioneer sends a signed *Friend* message. When the bidder receives the friend message they locate the auction in the list of auctions for which they have received auctions. This causes the user interface to update and display that the auction includes a discount. When the bidder selects an auction to join through the user interface the bidder application sends a Join message to the auctioneer. The message includes a Bidder object containing the bidder's details.

When the auctioneer application receives a Join message it first confirms that the message refers to the current auction. If the message is valid the Bidder object is removed and stored in the list of bidders in the data storage area. The bid status of the bidder is set to *no bid* meaning that the string representation of the object at this time is just the bidder's name. The auctioneer replies to the Join message with an acknowledgement. When the bidder receives this acknowledgement the Cryptography class generates a new symmetric key to encrypt the bidder's bid using the method *setNewSyncKey* shown in figure 4.5.

In order to make a bid the user enters their bid value though the user interface of the bidder application. The bid is then encrypted using the Cryptography class. If the bidder has been identified as a friend the friend increase percentage is added to the bid before

encryption. The Middleware class then sends the bid in a signed Bid message to the auctioneer. The auctioneer stores the encrypted bid in the appropriate Bidder object and acknowledges that the bid was received. The auctioneer also updates the bid status of the bidder to *bid*. When the BidPeriodThread has ended the auctioneer iterates thorough the list of bidders and creates *KeyBidPair* objects which contain the bidders public key for identification purposes and their associated encrypted bid. This list is then placed in a *BidsReceived* message and broadcast to the bidders.

The bidders iterate through this list amending their individual lists of Bidder objects with the newly received encrypted bids. They also broadcast the keys used to encrypt their bids. As keys are received by the auctioneer and bidder applications the encrypted bids are decrypted, the plaintext bid values are stored in each Bidder object and the user interface is updated to display bid value.

4.3.4 User Interface

The user interface for both applications is comprised of a set of Activities. The Activities that make up the bidder application user interface are *BidderApplicationActivity*, *AuctionRunActivity*, *LocationDisplayActivity* and *ResultActivity*.

Each activity runs in its own thread but provides a handler so that its user interface component can be updated by another thread. For example when the bidder application Middleware receives an Announce message and adds the Auction object to the bidder's list of auctions it updates the BidderApplicationActivity using a handler. The BidderApplicationActivity is a special type of activity called a ListActivity. When it is created is is bound to the list of Auction objects in the data storage class. The string representation of each Auction is displayed in the list. However, when an item is added to the list the BidderApplicationActivity be informed to refresh the contents of the list. This action is performed through the hander.

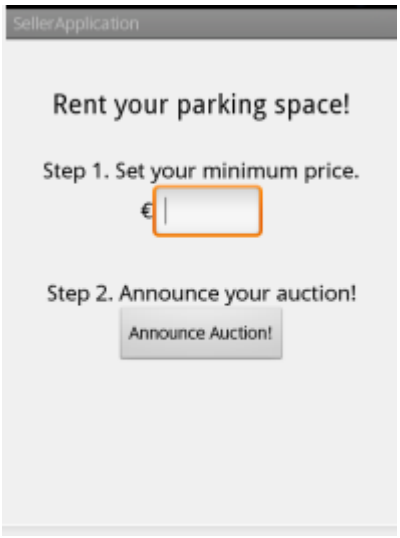

Figure 4.6: Sample Auctioneer Application Activity

The user interface of the auctioneer application is comprised of the *AuctioneerApplicationActivity*, *AnnounceActivity* and the *AuctionRun* Activities.

As mentioned previously the layout of the user interface is separated from the application code such as the handlers to perform actions on button presses, etc. Each Activity must implement the *onCreate* method where the layout of the activity is set using the method *setContentView*. For example the XML layout file for the *AnnounceActivity* Activity is defined in *auction.xml*. That XML file specifies that the Activity should contain TextView and Button objects in a vertical linear layout as seen in figure 4.6. These are form elements provided by Android. Each element in the XML layout contains an identifier which can be user by the Activity to interact with the element, for example to react to a button click or set the text in a TextView.

4.3.5 Yarta Layer

The Yarta application runs in the background on the auctioneer and bidder Android devices. As the seller application is loading it binds to the Yarta service and requests friendship information. Calls to extract information from the Yarta knowledge base take

```
void doBindService() {
    bindService(
            new Intent(CommunicationConstants.YARTAKBSERVICE_INTENT),
            mConnection, Context.BIND_AUTO_CREATE);
    mIsBound = true;
    mCallbackText.setText("Binding.");
}
```

Figure 4.7: Binding to the Yarta Service

```
List<Agent> knownPersons;
try {
    knownPersons = mSAM.getMe().getKnows();
    for(Agent a: knownPersons){
        if (a instanceof Person){
            UserStorage.addFriend(((Person)a).getPublicKey());
        }
    }
} catch (KBException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

Figure 4.8: Extracting Public Keys of Friends From Yarta

time as they involve inter-process communication. For this reason all friendship information is retrieved and stored in the data storage class. In order to bind to the Yarta service the application invokes the the *doBindService* as shown in figure 4.7. Yarta represents a person or group of people as *Agents*. In order to establish who the auctioneer's friends are it firstly retrieves the Yarta Agent object representing the auctioneer. It then retrieves the list of Agents who are linked to the that agent through the *knows* relationship in the knowledge base. Figure 4.8 shows the public keys of the Agents being extracted and stored in the data storage class.

Chapter 5

Evaluation

5.1 Introduction

In order to evaluate threats that this protocol aims to address (identified earlier) a brief overview of each threat is presented in turn, along with the defense that the protocol provides again that threat.

5.2 Analysis of Protocol

5.2.1 Malicious Attacks

Malicious threats were identified as attacks carried out in order to disrupt the running of the auction.

Replay Attack 1

A malicious party re-sends a bid message from a previous auction in order to confuse the auctioneer into choosing a bidder who is not even participating in the auction as the

winner.

The main defense against this attack is that all traffic related to the auction carries an auction identifier. This allows several auctions to be conducted over the same ad hoc network by allowing participants to filter out traffic relating only to their auction. As all bid messages carry a signed cryptographic hash of the message contents any attempt by the malicious party to alter the auction identifier in the message would render the message invalid. The auctioneer ignores any messages that do not carry a valid signature.

Replay Attack 2

A malicious member of the network re-broadcasts the auction announcement message of a previous auction when the space is not available.

Each auction announcement message contains the time at which the message was sent. Auction announcement messages become invalid after a period of time. If the message is re-broadcast after this time it will be ignored by any recipients. Recipients compare the time in the message against the current time determined from from GPS satellites to ensure clock synchronisation.

5.2.2 Advantage-Motivated Attacks

Advantage motivated attacks were defined at attacks that allow a bidder or seller to interfere with the fair running of the auction.

Bid Modification

A malicious party modifies the bid of other bidders to a value lower than its own bid as it routes bid messages.

Although bidders are required to act as routers and route auction traffic in the ad hoc network the protocol protects the integrity of bids. Firstly, as the value of the bid in bid messages is encrypted with a symmetric key no node routing the message can determine the value of the bid. Bid messages also carry a signed cryptographic hash of the message so any modifications to the message are detected by the auctioneer and invalidate the message.

Bidder Impersonation Attack

In order to increase their chances of winning the auction a malicious bidder impersonates their fellow bidders and submits low bids on their behalf.

This threat is prevented by the use of public key cryptography. An auctioneer only accepts bid messages that contain the correct signed hash of the message. Any bid message received that is not signed correctly is regarded as a forgery and ignored.

Eavesdropping Attack

Bidders examine the contents of a bid message and submit a higher bid based on what they find, giving them an advantage in bidding.

As with the bid modification attack, the symmetric encryption employed during bidding prevents nodes from determining the contents of bid messages. An attacker may store a bidder's decryption key and attempt to use it to decrypt future bids by that bidder. This attack is not effective as bidders generate a new bid encryption key for every auction in which they participate.

Collusion Attack 1

In this attack, a bidder plants a *stooge* in the auction or participates themselves by pretending to be a bidder to influence other bidders to bid higher.

This attack is mitigated by sealed-bid nature of the auction. Although a stooge bidder may participate in the auction, they cannot be effective as the other bidders will not find out the value of their bid until after bidding has been completed.

Collusion Attack 2

To perform this attack a malicious seller examines the sealed bids after the close of bidding and informs a malicious bidder of the winning bid. The malicious bidder then submits a higher bid to win the auction.

The auctioneer is unable to examine the contents of bids until they have been provided with the private keys. In order to receive the decryption keys they must first send out a list of all the encrypted bids that they have received. The bidders and the auctioneer decrypt the bids at the same time and bidders who think they have submitted a top bid send claim messages. Two things deter the auctioneer from acting dishonestly at this stage of the auction. The first is their desire to settle with a claimant as quickly as possible in order to finalise the auction before the claimant is disconnected from the network or decided to take another parking space. The second is that all bidders, whether they have won or not listen for settlement messages. This provides a check on the conduct of the auctioneer. If the auctioneer does not broadcast a settlement message or broadcasts one for a winner who is not recognised by the other bidders, they are unlikely to participate in an auction with that auctioneer again.

Misinformation Attacks

The first misinformation attack identified in Chapter 3 occurs when auctioneers lie about the current winning bid received. This attack is not applicable to this auction protocol as the auctioneer has no knowledge about the contents of the encrypted bids it holds. The second attack occurs when auctioneers provide false information to bidders about the level of interest in the auction.

The protocol is designed in such a way that after the announcement period the auctioneer sends the list of bidders' public keys to participating bidders. This list allows bidders to make an informed decision on how much they will bid given the level of interest in the auction. The list of public keys also allow the bidders to verify the authenticity of key sharing messages during the decryption and settlement period. This design allows for the possibility that an auctioneer will include public keys of participants who have not registered to participate in the auction in the bidders' keys list. The motivation for allowing this to happen is twofold. Firstly, knowledge of how many bidders are in the auction allows bidders to make strategic decisions on how much to bid. Secondly, in the case where the auctioneer has falsely inflated the number of auction participants, the information is still of use to the bidders, as the number is always the maximum possible number of participants that can participate in the bidding process.

5.3 Conclusion

The auction protocol provides defenses against the attacks and challenges identified. Much of the defense has been provided by the fact that this protocol implements a sealed-bid auction. As bidders encrypt bids themselves and do not reveal the decryption keys until after the auction the auctioneer is prevented from acting maliciously during the bidding phase.

A design decision has been made to allow for the possibility of an auctioneer claiming an auction has more participants than it really has by including their public keys in the list of registered bidders. This decision has been made because informing bidders of the number of bidders participating in auction before bidding begins helps to prevent sniping. Although bids are encrypted bidders close to the auctioneer can still count the number of bid messages to determine the level of interest in the auction and may use this information that is not available to peripheral bidders to make a judgement on how much to bid. As bidders are provided with the list of participants before they bid each bid is made with the assumption that all of the bidders listed will bid and the bid value is set accordingly. A malicious bidder can no longer infer that a low number of bid messages being routed to the auctioneer means that a low bid may be able to win the auction.

Chapter 6

Conclusion

6.1 Conclusions

This research presented a method of conduction auctions over ad hoc networks. It is hoped that applications conducting auctions over ad hoc networks can utilise this protocol.

In order to identify the real world threats that such a protocol faces this research focused on a specific problem space, i.e. auctions for parking spaces. Analysis showed that the network communication issues, malicious behavior by auction participants and high node mobility were the key challenges in the development of the protocol.

Based on the threat and challenge analysis it was determined that existing protocols were unsuitable for conducting auctions of smart city resources. The protocol was designed taking the limitations of these protocols into account.

The protocol was successfully implemented in a sample application demonstrating that mobile devices are capable of supporting it.

In the evaluation of the protocol each of the identified attacks were shown to have been prevented or have their effectiveness dramatically reduced.

The protocol was successfully biased using the data re-use and sharing capabilities of the Yarta middleware presented in Toninelli et al. (2011).

6.2 Future Work

At present the application has only run on the Android emulator with its limited functionality. The application should be tested on a real ad hoc network or ad hoc network simulation in order to determine how well the protocol runs in the real world.

The issue of limiting the geographical distance a broadcast message may travel was raised during this research but has not yet been implemented. Determining an optimal distance for auctioneers to broadcast messages may be an interesting area of future research.

The protocol could be deployed in other applications to determine its suitability for conducting auctions of other resources.

Other methods of biasing the auction could be examined, perhaps using data shared from other applications, such as applications that allow users to organise carpooling.

Bibliography

Borcea, C., Gupta, A., Kalra, A., Jones, Q. & Iftode, L. (2007), The mobisoc middleware for mobile social computing: challenges, design, and early experiences, *in* 'Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications', MOBILWARE '08, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, pp. 27:1–27:8.
URL: *http://dl.acm.org/citation.cfm?id=1361492.1361526*

Chen, R., Park, J.-M. & Snow, M. (2006), Care: Enhancing denial-of-service resilience in mobile ad hoc networks, *in* 'Computer Communications and Networks, 2006. ICCCN 2006. Proceedings.15th International Conference on', pp. 5 –10.

Diffie, W. & Hellman, M. (1976), 'New directions in cryptography', *Information Theory, IEEE Transactions on* **22**(6), 644 – 654.

Doghri, I. (2008), Formal verification of wahs: an autonomous and wireless p2p auction handling system, *in* 'Proceedings of the 8th international conference on New technologies in distributed systems', NOTERE '08, ACM, New York, NY, USA, pp. 20:1–20:10.
URL: *http://doi.acm.org/10.1145/1416729.1416754*

Foth, M. (2006), 'Facilitating social networking in inner-city neighborhoods', *Computer* **39**(9), 44 –50.

Fourati, A. & Agha, K. A. (2006), Deploying auctions over ad hoc networks, *in* 'e-Business

Engineering, 2006. ICEBE '06. IEEE International Conference on', pp. 9 –16.

Paillier, P. (1999), Public-key cryptosystems based on composite degree residuosity classes, *in* 'Proceedings of the 17th international conference on Theory and application of cryptographic techniques', EUROCRYPT'99, Springer-Verlag, Berlin, Heidelberg, pp. 223–238.
URL: *http://dl.acm.org/citation.cfm?id=1756123.1756146*

Parkes, D. C., Rabin, M. O., Shieber, S. M. & Thorpe, C. A. (2006), Practical secrecy-preserving, verifiably correct and trustworthy auctions, *in* 'Proceedings of the 8th international conference on Electronic commerce: The new e-commerce: innovations for conquering current barriers, obstacles and limitations to conducting successful business on the internet', ICEC '06, ACM, New York, NY, USA, pp. 70–81.
URL: *http://doi.acm.org/10.1145/1151454.1151478*

Parsons, S., Rodriguez-Aguilar, J. A. & Klein, M. (2011), 'Auctions and bidding: A guide for computer scientists', *ACM Comput. Surv.* **43**, 10:1–10:59.
URL: *http://doi.acm.org/10.1145/1883612.1883617*

Pietiläinen, A.-K., Oliver, E., LeBrun, J., Varghese, G. & Diot, C. (2009), Mobiclique: middleware for mobile social networking, *in* 'Proceedings of the 2nd ACM workshop on Online social networks', WOSN '09, ACM, New York, NY, USA, pp. 49–54.
URL: *http://doi.acm.org/10.1145/1592665.1592678*

Shamir, A. (1979), 'How to share a secret', *Commun. ACM* **22**, 612–613.
URL: *http://doi.acm.org/10.1145/359168.359176*

Toninelli, A., Pathak, A. & Issarny, V. (2011), Yarta a middleware for managing mobile social ecosystems, *in* J. Riekki, M. Ylianttila & M. Guo, eds, 'Advances in Grid and Pervasive Computing', Vol. 6646 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 209–220.

Zheng, S., McAven, L. & Mu, Y. (2007), First price sealed bid auction without auctioneers,

in 'Proceedings of the 2007 international conference on Wireless communications and mobile computing', IWCMC '07, ACM, New York, NY, USA, pp. 127–131.
URL: *http://doi.acm.org/10.1145/1280940.1280967*