OMiN - Opportunistic Microblogging Network

Neil Wells, supervised by Tristan Henderson

Abstract

OMiN is a pocket switched network running on smartphones. It allows users to send and receive microblog messages without using any global infrastructure such as the internet. Smartphones in close proximity to each other pass on messages according to a variation of the PROPHET routing protocol. Steps have been taken to verify the security of the network by protecting it against known attack vectors.

Declaration

Contents

Introduction

TODO

Objectives

Primary Objectives

- design and implement a protocol for discovering nodes in close proximity and passing messages and necessary metadata between them.
- Create a core library to manage message storage and routing.
- Implement a simple epidemic routing algorithm to send messages to all available nodes.
- Design a routing algorithm using user metadata to route messages while disguising message content and metadata.

Secondary Objectives

- Create a smartphone UI.
- Implement a more advanced routing algorithm.
- Design and implement a mechanism to decide whether a node is trustworthy or not.
- Evaluate the performance of the implemented routing algorithms.

Tertiary Objectives

- Compare the real world vs simulated performance of the routing algorithms.

Context Survey

The following provides brief summary of the current state-of-the-art in opportunistic network technology. Only the most relevant subjects will be addressed in order to give the reader sufficient background information to fully understand the project.

Opportunistic Networks

An opportunistic network is a network where connections between nodes are sparse and a direct path from source to destination cannot be guaranteed. For example, a common form of opportunistic network (and the form we will focus on) is the Pocket Switched Network (PSN) - a network of smartphones carried around by people. Because of the predictable nature of human behaviour, much research has been done to improve PSN algorithms.

Opportunistic networks must be able to store messages and forward them when connections become available. Problems that have been solved in conventional connected networks (security, routing etc.) are much harder to solve in opportunistic networks.

Similar Projects

Haggle

One of the largest opportunistic platforms is Haggle (http://www.haggleproject.org)[1], a pocket switched network designed to run on smartphones. There are implementations for a number of clients including Android (play.google.com/store/apps/details?id=org.haggle.kernel) and Windows Mobile.

By monitoring use of the platform, the authors discovered trends in inter-contact times and contact durations, showing that existing algorithms are poorly suited to real world models[2].

FireChat

FireChat (opengarden.com/firechat) is a new smartphone application used for off-the-grid messaging between nearby users. It has been used to circumvent government restrictions in Iraq (http://www.theguardian.com/technology/2014/jun/24/firechat-updates-as-40000-iraqis-download-mesh-chat-app-to-get-online-in-censored-baghdad) and during the Hong Kong protests (http://www.theguardian.com/world/2014/sep/29/firechat-messaging-app-powering-hong-kong-protests).

However, the app mostly relies on an internet connection, and its simple protocol is insecure (http://breizh-entropy.org/~nameless/random/posts/firechat_and_nearby_communication) and unable to implement the store-and-forward functionality of a proper opportunistic network.

SWIM

The Shared Wireless Infostation Model (SWIM) proposes using an opportunistic network to monitor whales[3]. Small nodes are attached to the whales, which record data such as location and interaction with other whales. Connected nodes transfer this data between each other. Whenever data is transferred to a base station (the paper proposes using seabirds), it can be collected and stored.

Because data is shared between nodes, it is no longer necessary to find a whale with a sensor in order to acquire data from that sensor. This is a perfect example of the power of opportunistic networks in an environment with very limited connectivity.

Routing Algorithms

Routing in opportunistic networks can be reduced to the problem of finding a destination node in a constantly changing graph. Most routing algorithms therefore build on existing graph search techniques. However, because the graph is constantly changing and is not necessarily random, such techniques are not necessarily the most effective (as shown by the Haggle project).

Context Based Routing

Context based routing is a form of greedy searching, where a single message is continually passed to the node most likely to reach the destination. There are a variety of methods to compute the utility of a node, including CAR[12] and MobySpace[13]. While it is not guaranteed to find the optimum path to the destination, it uses very few resources as there is only one message being stored.

Epidemic Routing

The opposite of context based routing is epidemic routing - a form of uniform cost search[4]. Copies of the message are passed at every opportunity until the network is saturated. This is often likened to the spread of a virus. While this approach will always find the optimal path (because it takes all possible paths), it is very resource intensive - all nodes are expected to store every possible message. For this reason, routing protocols that use similar techniques (dissemination based routing) concentrate on avoiding unnecessary use of resources.

PROPHET

The Probabilistic Routing in Intermittently Connected Networks (PROPHET) algorithm[5] uses a form of the A* algorithm. A utility function (derived from recent encounters with nodes) is used to predict whether the message should be passed on to a node. This use of heuristics greatly improves performance when compared to epidemic routing.

Bubble RAP

The Haggle project discovered that algorithms that treat routing as a generic graph search problem are often unsuited to PSNs. Bubble RAP[6] works on the idea that a social connections graph has tree like structure, where close nodes form a community. In order to send messages to a different community, the message is moved up the tree towards highly connected nodes near the root, and then back down the tree towards the destination community and, eventually, the destination node.

This has been shown using the data collected from Haggle to be more effective than standard routing algorithms.

Security

Opportunistic networks (especially pocket switched networks) must consider the possibility of an unsecure node which does not conform to the protocol specification (either through accident or design) or attempts to discover private information. Common attack types include:

- Sybil attacks: impersonating another node in order to send messages that appear to be from that node or to receive messages intended for the node.
- Majority attack: by controlling a large number of nodes, an attacker can control a network which assumes that the majority of nodes can be trusted.
- Eavesdropping: gathering information such as message metadata to discover private information such as message contents and user location.
- Denial of Service: saturating the network with unwanted messages.
- Packet dropping: failing to pass on messages to either reduce resource usage or as part of another attack.

Trust Based Security

Trust based security mechanisms depend on generating a list of trusted or untrusted nodes. This is commonly based on trusting ‘friends’ in a social network[8] or distrusting nodes exhibiting strange behaviour[9]. While trust based security mechanisms can lower the chances of an attack, they are not infallible.

Certificate Based Security

In order to increase security, a security mechanism must reduce the number of trusted parties to an absolute minimum. This can be done using an infrastructure of public key certificates to verify the identity of a node. However, the problem of distributing this infrastructure without trusting an arbitrary number of nodes has not been solved yet. On the internet, it is done by trusted certificate authorities, who manage and release certificates. However, having a central trusted authority is infeasible in an opportunistic network, as all nodes would have to connect to it directly at some point. Some mechanisms, like the one proposed by Shikfa et al[10] do use a central server, but only require it to be available for nodes joining the network. Other mechanisms split the sdf. Mechanisms for distributed certificate distribution require some level of trust in network nodes. For example Capkun et al's approach[11] does this by building a graph of certificates determining who trusts who. Any abnormalities in the trust graph may indicate foul play. However, this scheme is still vulnerable to a majority attack.

Identity Based Security

Identity based encryption is an increasingly common form of encryption where the user's details (such as a username) acts as a public key, and a private key will be generated by some central server. However, having a central server is problematic in a distributed network. Some security frameworks assume that there is a central server that can and will be accessed occasionally[14]. Others split up the central server into multiple nodes, all of whom must collaborate to generate a private key[15].

Requirements Specification

User Requirements

Non-Functional Requirements

- High: The user shall be able to create a unique identity.
- High: The user shall be able to send text messages to all others who follow the user or a hashtag in the message.
- High: The user shall be able to 'follow' any user and receive messages sent by that user.
- Low: The user shall be able to 'follow' any hashtag and receive messages containing that hashtag.
- Low: The user shall be able to send encrypted direct messages to a single user.
- Low: The user shall be able to send multimedia messages to all others who follow the user or a hashtag in the message.

Functional Requirements
None

System Requirements
Non-Functional Requirements
- High: The system shall work on smartphones or tablets capable of connecting to a wifi network.
- High: The system shall allow creation of user identities with a unique cryptographic identity.
- High: The system shall automatically connect to nearby nodes and pass on relevant information such as messages.
- Medium: The system shall provide a mechanism for securely distributing the cryptographic identity of a user.
- Medium: The system shall protect user metadata such as location and friends list from all other nodes.
- Medium: The system shall ensure that messages cannot be modified in transit or that such modifications can be detected.
- Medium: The system shall ensure that nodes cannot send a message that appears to be from another user.
- Medium: The system shall be robust and able to continue functioning when it encounters an unexpected state such as a malfunctioning or untrustworthy node.
- Medium: The system shall ensure that encrypted direct messages cannot be read by third parties.
- Low: The system shall be able to support multiple user identities on a single node.

Functional Requirements
- High: The system shall collect anonymous logging data for debugging and profiling purposes.
- Medium: The system shall ensure that a reasonable number of messages reach their intended recipients.
- Medium: The system shall restrict the size of the message buffer by evicting messages.
- Medium: The system shall use a minimal amount of the available power.

Software Engineering Process

TODO

Ethics

In order to test the real world performance of the network, we may ask people to use the application. In this case, some metadata will be collected on users, with their consent. This may include:

- An anonymous user ID.
- Anonymised 'Friends list' (or equivalent) of users.
- Times and locations of encounters between anonymous users.
- Metadata of messages passed during encounters, including message ID and origin ID, but NOT message contents.

Design

Use cases

Our network is designed to enable connectivity in areas where access to communications infrastructure (such as the internet) is impossible or infrequent. This includes disaster areas where such infrastructure has been destroyed and remote areas (such as oceans and deserts) where communications infrastructure does not exist. Considering these use cases, the problem parameters are as follows:

- No external communications infrastructure while the network is in use.
- Assume an internet connection during setup (planned expedition into remote areas) OR tightly connected network (small disaster area).

Authentication Models

Authentication is a major issue in opportunistic networks: how can we be sure that a message was sent by a user, and has not been modified.

The simplest option is to forfeit authentication entirely and trust every node to behave correctly. This is possible for small networks, but as the network grows, the chances of having a deceitful user grows (even if each user is 99.99% trustworthy, a network with 7000 users has a 50% probability of having a deceitful user). In order to build a scalable network, we must discount this strategy as being unfeasible.

Networks which may contain deceitful users must use measures to prevent this. A common approach is for each user to identify nodes which may be trustworthy or untrustworthy. Social authentication is a common strategy: only 'friends' of users are considered trustworthy. The IRONMAN algorithm takes a different approach by detecting nodes displaying unusual behaviour. These nodes are then marked as untrustworthy. This approach of using heuristics to separate trustworthy and untrustworthy nodes is useful, but still susceptible to attack.

The best approach to avoid trusting an attacker is to trust as few nodes as possible. The only way to verify that a message actually came from a user is to have some certificate irrefutably proving that a message was sent by a user. If a message comes with a cryptographic certificate then anyone with their public key can verify its origin. So we now encounter a new problem: how can we distribute a public key while trusting as few nodes as possible?

Public Key Distribution

In order for a key based approach to work, we must be able to verify that an arbitrary user identifier (i.e. a username chosen by the user) is uniquely associated with an arbitrary (randomly chosen) public key. Modern Internet based techniques involve a trusted certificate authority (CA) who will distribute certificates guaranteeing that a key is associated with a user. This is impractical in a pure opportunistic network setting (totally free of infrastructure), as it involves every node having contact with the CA, which considerably restricts the size of the network.

In cases where a sporadic internet connection is available, a central CA can be used to allow identity based encryption by generating a private key for a user, so their identity (username) can be used as their public key. This has the added advantage of preventing sybil attacks because the CA can ensure that every user has a unique username.

Instead of formally verifying identities, we could build up a trust based infrastructure for distribution of public keys. While this is a common method used in opportunistic networks, trust based approaches cannot be 100% secure (in particular, they are vulnerable to attackers controlling the majority of nodes).

Another approach is to remove the username altogether. The only public piece of information about a user is their public key. If this can be distributed securely outwith the network (as phone numbers and email addresses are), our system can be made totally secure. However, public keys are large and difficult to transfer without electronic means (even QR codes are only just able to fit a 2046 bit key). sfddfdfdg

Perhaps the only foolproof method to verify a public key is to physically meet the owner and get them to verify the key. The only user we trust is the owner of the public key (because nothing they do during the verification would allow them to compromise the network). However, our network is designed as a microblogging platform, and we do not want to have to physically meet every person we are interested in.

Lets take a step back and look at the problem in a different light: we are trying to use the opportunistic network to verify that an arbitrary user identifier (i.e. a username chosen by the user) is uniquely associated with an arbitrary (randomly chosen) public key. But is this approach really necessary. Social convention dictates that a person is free to choose their own username. However if a user can choose a specific username then an attacker (or another user with a similar taste in usernames) can also choose the same username (a sybil attack). We can avoid these unnecessary complications simply by creating a username that is dependent on the public key (e.g. a truncated version of the public key). When a node tells us that a specific username is tied to a specific public key, we don't have to trust them - we can check that it is true.

We can extend this argument further by saying that if a username is wholly dependent on the public key then it is inconsequential. A user can be identified by their public key alone. However, real life undermines this beautiful scheme - public keys are too large to pass between people easily. Small, arbitrary usernames can be written down or embedded in a QR code but large public keys are too unwieldy to transfer in physical form and we cannot guarantee that other methods of digital communication are available.

Based on the above logic, we propose the following scheme for verifying messages passed through an opportunistic network:

Users generate a random public/private key pair using 4096-bit RSA. Their username is the base-64 representation of the SHA-256 hash of the public key with some parity bits (a 4 character string - this seems small but the chances of collisions are minute). The public key/username pair can then be distributed throughout the network. Messages sent by the user are signed with their private key.

If I wish to obtain the public key of a user, I must gain their username without using the opportunistic network (in a similar way to obtaining an email address). Once I have this, I can ask other nodes for the associated public key. This request may be obscured by various means such as those described in the SSNR-OSNR paper. I can verify that the public key is indeed related to that username and I can now verify that any message received from that user definitely originated from that user, without ever having to trust a network node.

The main disadvantage of this technique is that users are no longer able to choose meaningful usernames. We believe that this is outweighed by the benefits of having total trust in the network.

# Selfish Nodes

While this scheme is effective at protecting from sybil attacks and message modification, it is unable to protect against selfish nodes which only receive messages and never pass them on. Such nodes can be detected and avoided using algorithms such as IRONMAN. However, because our routing algorithm is dissemination-based, such nodes are unlikely to impact the network so our network will make no attempt to prevent this behaviour.

Routing Algorithm

Now we come to the problem of routing. Simply put, a routing algorithm takes all available information about a message and uses that information to decide where to send it. In trust based networks, this information is freely available to all trusted nodes (friends lists, recipients, previous paths etc). However, we have opted for a model where all nodes are considered untrustworthy. This means that we must obscure or remove all of this information, while still allowing a routing algorithm to use it.

In our network, the goal is to broadcast messages from a sender to all users interested in messages from that sender (we will ignore the addition of hashtags for now). Pocket switched networks are mobile networks without infrastructure. This means that it is very hard (though not impossible) to use location based or hierarchy based techniques to route messages. There are two main approaches to routing in an infrastructure-less network: dissemination based and context based routing. Dissemination based routing protocols are a form of controlled flooding, where the message is forwarded to any node that may be interested. This tends to create many unnecessary copies of the message, so a good buffer eviction algorithm is needed. Context based protocols have a more discerning approach where limited copies of the message are passed around. Messages sent with this approach tend to take more time to reach the recipient. Our network is a blogging platform where messages are being broadcast to a potentially large number of users. Because context based algorithms limit the number of copies of a message, this approach is unsuited to our needs. For this reason, we will use a dissemination based approach.

Epidemic routing is the simplest and most well known version of dissemination based routing. In this approach, copies of the message are passed to every node that is encountered. While it is the most optimal solution to get a message to the destination it is also the most resource intensive, requiring lots of unnecessary message passing and storage. For this reason we should attempt to use a more selective algorithm.

Burns et al propose a clever location based routing algorithm, the MV (Meeting/Visit) algorithm. This uses knowledge of user location and movements to efficiently forward messages to their destination. This uses static nodes at certain locations to increase the likeliness of a successful transfer. While these static nodes can greatly improve the efficiency of the network, they restrict the network to locations where static nodes are in place. For this reason, we consider the MV algorithm to be unsuitable for our network.

The PROPHET algorithm takes a simpler, probability based approach. Every node stores a probability of being able to forward a message to the recipient. This probability is transitive: if node A is able to forward messages to a recipient and node B regularly meets node A, then node B is also able to forward the message to the recipient. This can be adapted to suit our interest based model where nodes have an interest in receiving messages from a certain user: the probability of being able to forward a message to a recipient is based on how interested the recipient is in the message. The transitive property also holds. We can use the use the SSNR-OSNR algorithm to obscure metadata such as interest lists: By adding random values and encrypting metadata in a bloom filter, we can allow a routing algorithm to determine whether a value is probably in a set of data without revealing the full contents of the set. We can use probability based variants of the bloom filter to allow an approximation of the PROPHET algorithm.

The BUBBLE Rap algorithm is a well known social based algorithm. It works by identifying popular nodes in a community capable of passing messages to other communities. The algorithm attempts to send messages via these popular nodes, as they are more likely to be able to pass the message on to a distant community. While this algorithm has been shown to be more effective that algorithms such as PROPHET, we do not see how it can be adapted to suit the broadcast based model of our network.

# Message Buffer Eviction

When the message buffer is too large, it must evict a message. This message should be the least likely to be used again. The simplest approach is the LRU (Least Recently Used) algorithm. Here, the oldest messages are removed on the assumption that they are probably the most widely distributed.

However, we have better ways of measuring message distribution. Give each message a distribution count initialised to 0. Every time the message is sent or a node is encountered which already has the message, this distribution count is incremented. At eviction time, the message with the highest distribution count is evicted.

TODO

Implementation

TODO

Evaluation and Critical Appraisal

TODO

Conclusions

TODO

References

[1] Scott J, Crowcroft J, Hui P, Diot C, Others. Haggle: A networking architecture designed around mobile users. In: WONS 2006: Third Annual Conference on Wireless On-demand Network Systems and Services; 2006. p. 78-86.

[2] Chaintreau A, Hui P, Crowcroft J, Diot C, Gass R, Scott J. Impact of human mobility on opportunistic forwarding algorithms. Mobile Computing, IEEE Transactions on. 2007;6(6):606-620.

[3] Small T, Haas ZJ. The Shared Wireless Infostation Model: A New Ad Hoc Networking Paradigm (or Where There is a Whale, There is a Way). In: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking &Amp; Computing. MobiHoc '03. New York, NY, USA: ACM; 2003. p. 233-244. Available from: http://doi.acm.org/10.1145/778415.778443.

[4] Vahdat A, Becker D, Others. Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University; 2000.

[5] Lindgren A, Doria A, Schelén O. Probabilistic Routing in Intermittently Connected Networks. SIGMOBILE Mob Comput Commun Rev. 2003 Jul;7(3):19-20. Available from: http://doi.acm.org/10.1145/961268.961272.

[6] Hui P, Crowcroft J, Yoneki E. Bubble Rap: Social-based Forwarding in Delay Tolerant Networks. In: Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing. MobiHoc '08. New York, NY, USA: ACM; 2008. p. 241-250. Available from:http://doi.acm.org/10.1145/1374618.1374652.

[7] Pelusi L, Passarella A, Conti M. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. Communications Magazine, IEEE. 2006 Nov;44(11):134-141. Available from: http://dx.doi.org/10.1109/MCOM.2006.248176.

[8] Trifunovic S, Legendre F, Anastasiades C. Social Trust in Opportunistic Networks. In: INFOCOM IEEE Conference on Computer Communications Workshops, 2010; 2010. p. 1-6. Available from: http://dx.doi.org/10.1109/INFCOMW.2010.5466696.

[9] Bigwood G, Henderson T. IRONMAN: Using Social Networks to Add Incentives and Reputation to Opportunistic Networks. In: Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on; 2011. p. 65-72. Available from: http://dx.doi.org/10.1109/PASSAT/SocialCom.2011.60.

[10] Shikfa A, Onen M, Molva R. Bootstrapping security associations in opportunistic networks. In: Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on; 2010. p. 147-152. Available from:http://dx.doi.org/10.1109/PERCOMW.2010.5470676.

[11] Capkun S, Buttyan L, Hubaux JP. Self-organized public-key management for mobile ad hoc networks. Mobile Computing, IEEE Transactions on. 2003 Jan;2(1):52-64. Available from: http://dx.doi.org/10.1109/TMC.2003.1195151.

[12] Musolesi M, Hailes S, Mascolo C. Adaptive routing for intermittently connected mobile ad hoc networks. In: World of Wireless Mobile and Multimedia Networks, 2005. WoWMoM 2005. Sixth IEEE International Symposium on a; 2005. p. 183-189. Available from:http://dx.doi.org/10.1109/WOWMOM.2005.17.

[13] Leguay J, Friedman T, Conan V. Evaluating Mobility Pattern Space Routing for DTNs. In: INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings; 2006. p. 1-10. Available from: http://dx.doi.org/10.1109/INFOCOM.2006.299.

[14] Kamat P, Baliga A, Trappe W. An Identity-based Security Framework For VANETs. In: Proceedings of the 3rd International Workshop on Vehicular Ad Hoc Networks. VANET '06. New York, NY, USA: ACM; 2006. p. 94-95. Available from: http://doi.acm.org/10.1145/1161064.1161083.

[15] Kong J, Petros Z, Luo H, Lu S, Zhang L. Providing robust and ubiquitous security support for mobile ad-hoc networks. In: Network Protocols, 2001. Ninth International Conference on; 2001. p. 251-260. Available from:http://dx.doi.org/10.1109/ICNP.2001.992905.

Appendices

Appendix A - Testing Summary

TODO

Appendix B - Status Report

TODO

Appendix C - User Manual

TODO

Appendix D - Maintenance Document

TODO