Kadupul: Livin' on the Edge with Virtual Currencies and Time-Locked Puzzles

Magnus Skjegstad, Anil Madhavapeddy, Jon Crowcroft
Computer Laboratory
University of Cambridge
firstname.lastname@cl.cam.ac.uk

ABSTRACT

Devices connected to the Internet today have a wide range of local communication channels available, such as wireless Wifi, Bluetooth or NFC, as well as wired backhaul. In densely populated areas it is possible to create heterogeneous, multihop communication paths using a combination of these technologies, and often transmit data with lower latency than via a wired Internet connection. However, the potential for sharing meshed wireless radios in this way has never been realised due to the lack of economic incentives to do so on the part of individual nodes.

In this paper, we explore how virtual currencies might be used to provide an end-to-end incentive scheme to convince forwarding nodes that it is profitable to send messages on via the lowest latency mechanism available. Clients inject a small amount of money to transmit a message, and forwarding engines compete to solve a time-locked puzzle that can be claimed by the node that delivers the result in the lowest latency. Our approach naturally extends congestion control techniques to a surge pricing model when available bandwidth is low and does not require latency measurements.

1. INTRODUCTION

Devices connected to the Internet today have a wide range of local communication channels available. For example, most new wifi-routers and access points have two or more radios (one for 2.4 GHz and one for 5GHz communication). Connected to each access point there are clients with several radio technologies available, such as Bluetooth and NFC. Other physical communication channels also exist, for example LEDs, cameras [14] and microphones [10] depending on available hardware.

In urban areas it is possible to create heterogeneous, multihop communication paths using these technologies. As radio waves propagate at the speed of light, these paths offer lower-latency communication. However, there are few economic incentives for edge nodes to act as low-latency data forwarders, and the disincentive of wasting their batteries on other nodes' traffic.

As an example of how wireless edge nodes can be used for faster forwarding, consider a user A who wants to send messages to user B a few kilometers away, as illustrated in Figure 1. Using a traditional forwarding path the messages may first have to be delivered through the core network of A's ISP (or mobile operator), then be forwarded to the core network of B's ISP, before finally being delivered to user B. The forwarding latency in this example depends on the number of hops and distance to travel via the core networks of the ISPs, not the geographical distance between the nodes. As a result, nodes A and B may experience the same latency whether the geographical distance between them is one or tens of kilometers.

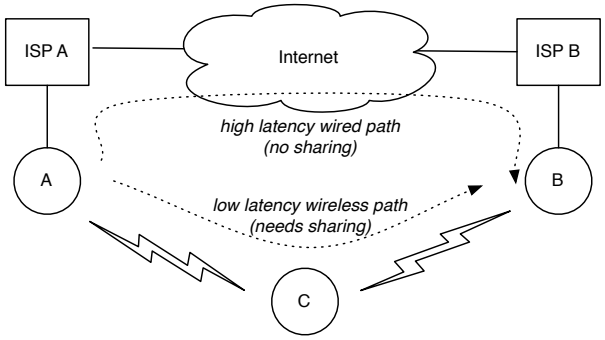


Figure 1: ISP and edge forwarding paths between nodes A and B

An alternative forwarding path could be established as a wireless path through intermediate radio devices between A and B, such as node C in Figure 1. This forwarding path could potentially achieve significantly lower latency than traditional methods, as well as being resilient to wide-area networking failures since it only depends on the local communications network. There are however few devices today that are willing or able to participate in the network as low-latency edge forwarders. We argue that the primary reason for this is not technical, but caused by lack of incentives compared to the increased workload and need for investment in the edge nodes. For example, it is not uncommon for edge nodes to participate as forwarders in city-wide mesh networks due to bad or expensive Internet connectivity, as in Athens [1], or to act as forwarders to improve communication during a political crisis. Examples of the latter are the Occupy Wall Street movement mesh

network [9] and the Open Mesh Project in Egypt [2]. In these cases the incentives caused by external factors outweigh the forwarding cost.

The perceived cost of forwarding a message depends on the workload imposed on the edge node. Depending on the technology, this workload may be low (forward on regular Wifi), but one could imagine higher workloads – such as for a mobile phone that has to turn on Bluetooth discovery for long time periods. The owner of the node may also have to install custom software or invest in upgraded hardware to forward messages faster. Other factors, such as power consumption may also play an important role.

A forwarding incentive can be created by simply offering payment to the forwarders, for example by using a decentralized virtual currency like Bitcoin [13]. A useful feature of many virtual currencies is that micropayments can be issued with minimal transaction costs and without relying on a centralized authority. It is more difficult to create a payment system with incentives for minimizing latency, since it is difficult to measure latency objectively in a way that can be accepted by all nodes. For example, the latency observed by the sender, a forwarder and the final recipient may not be the same, and all parties have economic incentives to under- or overreport the latency.

In this paper we present *Kadupul*, a system that incentivises low-latency forwarding between edge nodes without relying on latency measurements. This is accomplished by creating a reward system based on time-locked puzzles [12]. Time-locked puzzles [3] can be used to hide information until the puzzle is solved or the solution is provided by the creator or a third-party. Recently, a time-locked puzzle based on Bitcoin was proposed and implemented [16], which allows Bitcoin rewards to be locked for a given time period and be collected by the first node that solves the puzzle (or is told the solution).

We build on Todd's time-lock puzzle mechanism [16] to propose a forwarding model for rewarding forwarders by giving them an advantage in solving a puzzle. A forwarder can collect a reward if it provides the correct solution to a puzzle protecting it. Each forwarder is provided with a solution to a reward after it has forwarded a message. The catch is that *each puzzle is public and solvable by anyone after a known amount of time*. This creates a race to forward the message before the puzzle has been solved by other nodes.

The main contribution of this paper is a new forwarding model that creates incentives for low-latency forwarding without having to measure the latency. Economic incentives for optimizing other parameters in ad-hoc networks, such as bandwidth or power, have been proposed in earlier works (see e.g. [7]).

Note that although this paper primarily discusses Bitcoin as a reward system, the ideas described here can be used with other virtual currencies as well, as long as they provide a similar underlying P2P protocol.

We will now describe the basic mechanisms in Kadupul and propose several forwarding models based on time locked puzzles as incentives (§2). We then discuss future directions and conclude the paper (§3).

2. DESIGN

We describe the core functionality of Kadupul by first discussing how the forwarding paths are established and negotiated. We then describe in detail how time locked puzzles can be used as a low-latency forwarding incentive and propose several forwarding models based on the mechanism.

We assume that most of the nodes that participate as forwarders in Kadupul rarely change their geographical location (although they can). This means that although nodes may be anonymous as users of the virtual currency, they will over time be able to gather information about other radio nodes in their region. Kadupul nodes may use this to estimate risk by observing the behaviour and reliability of nodes they cooperate with over time.

A message forwarded by Kadupul can be of any size. For example, a high quality video can be transferred as a single message.

2.1 Establishing forwarding paths

Some coordination must be performed in advance to form heterogeneous multihop communication paths. The forwarders along the path may be able to discover each other directly using radio or other techniques, but if a wide range of technologies are being used over large geographical areas this may not always be possible.

In the following we provide an example of how a forwarding path may be set up with the help of a decentralized P2P network on the Internet.

We assume that in most cases the forwarding path is established by the sender. This requires an initial discovery step where the sender finds potential forwarders that together can forward information from the sender to the receiver. Potential forwarders can for example be discovered based on their location and local communication range, using a decentralized, Internet-based P2P network, such as proposed in for example [15] or [11]. Using the protocol proposed in [15], the sender can search for other nodes that are able to transmit to nodes in a given geographical region.

When a potential forwarder is found, the sender contacts the node directly over the Internet and requests a forwarding quote. If the price and capabilities of the forwarder is acceptable, the sender attempts to find other nodes that are able to receive the message when it is forwarded. The process is repeated with these nodes. If

no recipients are found (or the prices are unacceptable), the sender must find a different forwarder.

Note that a forwarder may have different radio technologies available, be able to transfer on different frequencies and at different power levels. Each configuration may have a different expected forwarding latency and price.

2.2 Time-locked forwarding rewards

After establishing the forwarding path, the sender must publish a set of rewards. We now describe the mechanism and protocol used to publish, as well as collect rewards for the forwarding nodes in more detail.

Todd proposes an interesting scheme for implementing time-locked puzzles with Bitcoin rewards [16]. A chain of rewards can be hidden in puzzles and published. Each puzzle has a value in Bitcoins that can be collected by any node that knows the solution to the puzzle. The solution can either be provided or can be calculated after a known amount of time. The puzzle is constructed in such a way that it *can be created in parallel, but only solved in serial.*

More specifically, the scheme from [16] uses multiple rounds of a SHA256 [5] hash to calculate a Bitcoin secret key from a randomly chosen initialization vector for each block of the reward puzzle chain. If the reward chain has for example 10 blocks, 10 initialization vectors are chosen and SHA256 is executed iteratively on each vector. Based on the result, a Bitcoin secret key is generated, which in turn is used to generate a public key and a Bitcoin address. The number of iterations determines how long it will take to recover the key pair if only the initialization vector is known. When the key pair is found, the public key must be revealed to collect the reward.

When the reward chain is made public, the initialization vector in each block (except the first) is obfuscated by XORing it with the SHA256 of the public key of the previous block. Thus, to decode a reward key pair iteratively you would first need the public key used to collect the reward from the previous block. This prevents the puzzles from being solved in parallel, while forcing reward collectors to reveal the missing piece of information required to collect the next reward.

To estimate the time it takes to unlock the reward we must assume a lower bound for how long it takes to perform the hash operation. As SHA256 is used to mine Bitcoins there have for several years been strong economic incentives to create faster hardware and software solutions to calculate it. A list of known Bitcoin hardware miners showing speeds in million double SHA256 hashes per second (Mhash/s) and power consumption (MHash/J) is maintained online in [4]. For example, the fastest ASIC-based device currently available can calculate 5.500.000 Mhash/s at 1506 Mhash/J. However, these devices rely on parallel operations and can not be used to crack a time-locked puzzle that must be solved in serial. A more realistic lower bound is thus similar to a very fast single core CPU or FPGA. The lower bound can be adjusted over time if it is observed in the block chain that a puzzle is broken faster than intended.

This mechanism can be adapted to create forwarding incentives in several ways, and we discuss four such forwarding models next. Note that control traffic (but not the actual data) is transferred over a higher latency control plane, which for example could be the Internet.

2.2.1 Double incentive forwarding

The objective in this model is to create a mechanism that makes the forwarders lose their reward unless they forward the message intact to the next hop as soon as possible, but also to create an incentive for assisting other forwarders. The full process is shown in Figure 2.

Figure 2a gives an overview of the initial tasks that must be performed before forwarding can begin. First, the sender must find the forwarders and negotiate the forwarding fees. The sender then generates and makes public a chain of rewards using time-locked encryption. For simplicity, we assume that one reward block is generated for each forwarder. The reward attached to each block may differ in value depending on the terms that were negotiated with the respective forwarder.

As shown in Figure 2b, the sender proceeds to send two values to each forwarder. The first value is a secret that enables the previous hop on the forwarding path to retrieve its reward. The second value is a nonce that when combined with a secret from the next hop on the forwarding path, as well as the hash of the message, results in the key required to unlock one of the rewards.

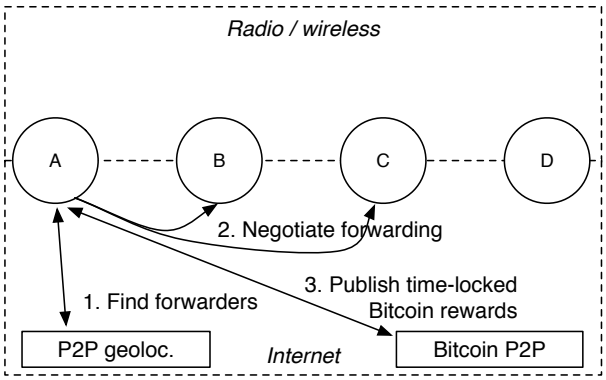
To avoid having to store the full message in each node, the nodes may use a rolling hash function to hash the message. The nodes then only need to maintain a buffer that contains a window with enough information to create the hash and to act as a message queue for forwarding.

Now the forwarding itself can begin, as shown in Figure 2c. The message is forwarded along the path and acknowledged by the next hop by sending the secret back to the forwarder. The acknowledgement address in the control plane is also sent with the message - for example an IP address and port number.

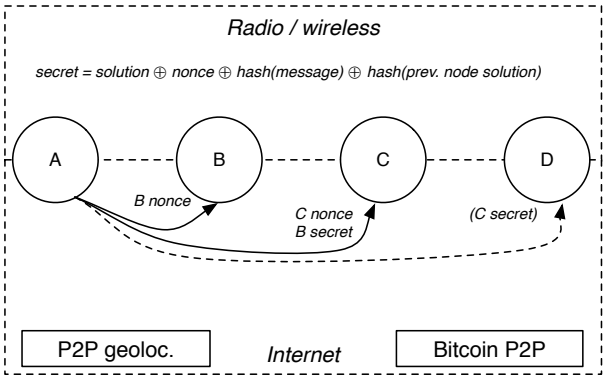
Figure 2d illustrates the reward collection process. After a node has forwarded the full message and received the required secret from the next node, it can reconstruct the puzzle solution and collect a reward. A forwarder will only be able to claim a reward when:

1. The previous node in the routing path was able to claim their reward and thus revealed the public key of the previous block

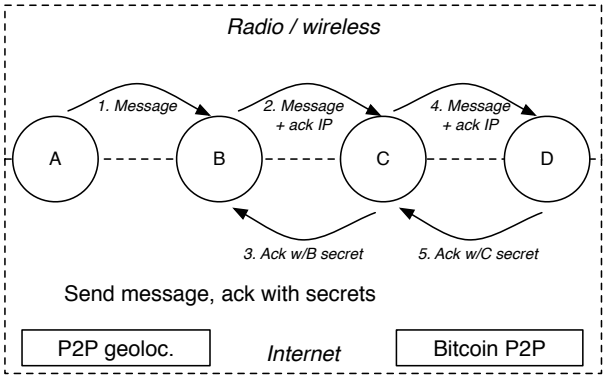
Figure 2: Negotiation, forwarding and reward collection with double incentive forwarding.



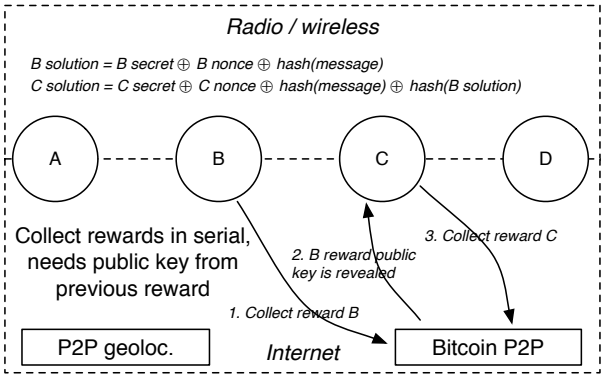
(a) Sender prepares forwarding path



(b) Sender distributes secrets and nonces



(c) Sender sends message, nodes acknowledge with secrets



(d) Forwarders reconstruct solutions and collect rewards

2. It has received the full message successfully so that it can generate a hash

3. The next node in the path has revealed the necessary secret generated by the sender, thus acknowledging that the full message was forwarded.

If the receiver is untrusted, the sender may encrypt the message with a key derived from the public key of the final reward to make the message unreadable until all forwarders have been rewarded.

This mechanism ensures that all nodes have incentives both to forward the message to the next hop (to obtain the missing secret to unlock the reward) and to supply the previous hop with its secret (otherwise they are unable to decode their own reward).

As the reward has a time-lock, the nodes are also given an incentive to perform the forwarding as fast as they can, or the encryption may be broken by other nodes in the Bitcoin network trying to claim the reward by brute forcing the keys. The node that can most effectively balance high-throughput and low-enough latency forwarding stands to profit most from solving the majority of puzzles it sees and claiming the rewards.

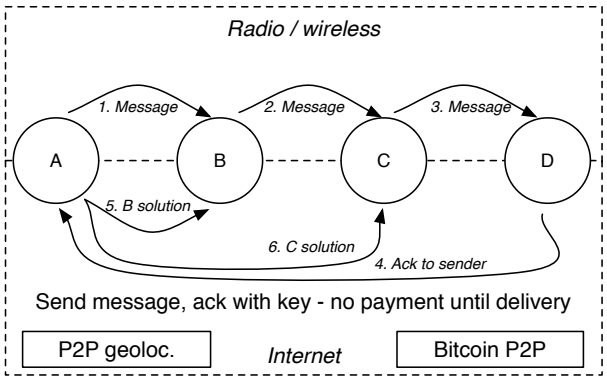
2.2.2 *All or nothing*

Note that in the forwarding model discussed in Section 2.2.1, the identity of every forwarder must be revealed to the next forwarder to allow the message to be acknowledged with the correct secret. In a broadcast network this may not be necessary. For example, a forwarder may be instructed to listen to a specific Wifi channel in promiscuous mode to receive the message and then just forward the message as it was received over another broadcast link. It may not be possible for the forwarders to reply with an acknowledgment in the same way if it has a weaker radio than the previous hop.

It is possible to use an alternative acknowledgement mechanism to avoid revealing the identity of the forwarders. This forwarding scheme is shown in Figure 3 and assumes a forwarding mechanism that can hide the Internet address of the forwarder, for example by using anonymous broadcast [17]. Instead of distributing secrets in advance, the final receiver acknowledges the receipt of the message to the sender and the sender then unlocks the puzzles for all the forwarders. This scheme requires more coordination between sender and receiver, but makes it difficult for the forwarders to collude. It also increases the forwarding risk as none of the nodes will receive their reward if the message is lost or delayed along the way, which may affect the forwarding price.

This forwarding model can be useful in "off-the-grid" mesh networks, where one of the goals is to avoid eavesdropping of traffic by global passive sniffing of conventional networks.

Figure 3: Broadcast forwarding without revealing forwarder identity to other forwarders.



2.2.3 Contract forwarding

Kadupul forwarding may also be used without establishing the forwarding path in advance. In this case the sender negotiates a forwarding contract with another node to bring the message to the recipient. It is then up to the node that accepted the forwarding contract to deliver the message as fast as possible. This node may use any number of subcontractors for the message to reach its final destination. This forwarding model is potentially simpler to implement and use for the sender, but the sender is no longer in control of the path the message travels. It may also increase the forwarding price as more work is left to the forwarders.

This forwarding model can especially be useful in combination with the other forwarding models. For example, a *pull* based delivery system can be constructed by letting the recipient negotiate a contract with the sender. When the contract has been accepted, the sender uses another forwarding model to deliver the content to the recipient.

2.2.4 Competing forwarders

The "all or nothing" model (§2.2.2) can be extended to create a competition between forwarders along multiple paths. This can be accomplished by using Random Linear Network Coding [8] (RLNC) or fountain codes [6] to encode partial messages and then forwarding the messages along multiple paths at the same time. RLNC and fountain codes are useful because the original message can be reconstructed when enough coded packets have been received. Fountain codes are end-to-end, but RLNC also allows recoding at intermediate nodes. When the recipient has received enough partial messages to reconstruct the original message, rewards are distributed to the forwarders depending on how much *innovative information* they forwarded.

This forwarding model can be especially useful for transferring messages that should be distributed to multiple edge nodes in the same area, such as for multicasting video streams.

2.2.5 Edge node caching

Kadupul naturally creates incentives for edge nodes to cache content. If an edge node is able to store content that is delivered frequently, it can volunteer to deliver the full content during the negotiation process and collect the full reward for the delivery. It would also be able to deliver the content in much shorter time than if it would have to be forwarded again, allowing it to provide a better offer than its competitors. Since each node must pay for its own cost of storing the cached data in the hope of future requests, they also have an incentive to develop efficient prediction and cache eviction algorithms.

3. DISCUSSION AND CONCLUSIONS

We have proposed a mechanism for creating an economic incentive for low-latency data forwarding and described how the mechanism can be useful for establishing low-latency forwarding paths between edge nodes on the Internet. The mechanism is decentralised and does not require latency measurements. We have mainly focussed on edge networks in this paper, but the mechanism can also be used in other types of networks where low latency should be rewarded.

An advantage of Kadupul in edge networks is that forwarding nodes have incentives to avoid congestion and long processing delays. If an area is congested, nodes that are able to forward with low latency would be able to demand a higher price. This increased price encourages central nodes to invest in special hardware to reduce the congestion. Two nodes may for example cooperate and set up long-range, point-to-point links that increases hop length and reduces interference. Similarly, nodes that are near a congested area, may set up point-to-point links that forward data across the congested area to a node on the other side. Commercial operators may also take advantage of the system, for example by selling low latency access using a privately owned frequency range.

Although all of the pieces needed to implement this type of forwarding exists today, some technical challenges remain before it can be fully realised.

For instance, it may take a long time to set up the initial forwarding path because several potentially slow tasks must be performed before forwarding can begin. For example, the reward puzzles can be generated in parallel, but may still be time consuming to produce if the puzzle is to be protected over longer time periods. In addition, if Bitcoin is used, transactions are relatively slow and it takes a while to publish rewards. Similarly, it may take some time for the forwarders to collect their rewards after receiving the solution to the puzzle. These delays require that additional time must be added to the

puzzles to make sure that rewards do not expire until the forwarders have been able to collect them. Today, this makes the method mostly suitable for applications that will use the path for longer time periods, such as for large, planned data transfers.

However, we are confident that these delays will become much smaller in the near future. As virtual currencies are becoming more popular, their protocols and software implementations are constantly being optimized to reduce delays. Furthermore, the number of reward puzzles that can be generated in parallel can dramatically increase if they are calculated using GPUs or FPGAs. Multiple rewards can be given to each hop while still being required to be solved in serial, and so the number of rewards created in parallel can exceed the total number of hops.

Kadupul helps to balance the economic needs of service providers and users to deliver a viable model for deploying reliable multihop edge networks, as well as enhancing the resilience of the global network by only using local links when possible.

4. ACKNOWLEDGEMENTS

We would like to thank Marcelo Bagnulo Braun, Thomas Gazagnaire, Andrius Aucinas, Pedro Ramos Pintos, Richard Mortier, Jeremy Yallop, Steven Murdoch and Petter Tveøy for helpful discussions of the ideas in this paper. The research leading to these results has received funding from the European Union's Seventh Framework Programme FP7/2007-2013 under Trilogy 2 project, grant agreement no 317756.

5. REFERENCES

[1] Athens Wireless Metropolitan Network (AWMN) web portal. Available online, July 15th 2014: `http://www.awmn.net/`.

[2] Open Mesh Project web page. Available online, July 15t 2014: `http://www.openmeshproject.org`.

[3] TIME-LOCK ENCRYPTION. Available online, June 16th 2014: `http://www.gwern.net/Self-decrypting\%20files`.

[4] Mining hardware comparison, last modified 28 February 2015, at 03:25. Available online, March 6th 2015: `https://en.bitcoin.it/wiki/Mining_hardware_comparison`.

[5] D. Eastlake 3rd and T. Hansen. US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF). RFC 6234 (Informational), May 2011.

[6] John W. Byers, Michael Luby, Michael Mitzenmacher, and Ashutosh Rege. A digital fountain approach to reliable distribution of bulk data. *SIGCOMM Comput. Commun. Rev.*, 28(4):56–67, October 1998.

[7] Jon Crowcroft, Richard Gibbens, Frank Kelly, and Sven Östring. Modelling incentives for collaboration in mobile ad hoc networks. *Perform. Eval.*, 57(4):427–439, August 2004.

[8] Tracey Ho, M. Medard, R. Koetter, D.R. Karger, M. Effros, Jun Shi, and B. Leong. A random linear network coding approach to multicast. *Information Theory, IEEE Transactions on*, 52(10):4413–4430, Oct 2006.

[9] Sarah Kessler. How Occupy Wall Street Is Building Its Own Internet [VIDEO], 14th Nov. 2011. Available online, July 15th 2014: `http://mashable.com/2011/11/14/how-occupy-wall-street-is-building-its-own-internet-video/`.

[10] Anil Madhavapeddy, Richard Sharp, and David Scott. Context-aware computing with sound. In *Fifth International Conference on Ubiquitous Computing*, volume 2864 of *Lecture Notes on Computer Science*, pages 315–332. Springer, 2003.

[11] M. Picone, M. Amoretti, and F. Zanichelli. GeoKad: A P2P distributed localization protocol. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, pages 800–803, March 2010.

[12] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock Puzzles and Timed-release Crypto. Technical report, Cambridge, MA, USA, 1996.

[13] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. Available online, June 17th 2014: `https://bitcoin.org/bitcoin.pdf`.

[14] David Scott, Richard Sharp, Anil Madhavapeddy, and Eben Upton. Using visual tags to bypass bluetooth device discovery. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(1):41–53, January 2005.

[15] Magnus Skjegstad, Brage Ellingsater, Torleiv Maseng, Jon Crowcroft, and Øivind Kure. Large-scale distributed internet-based discovery mechanism for dynamic spectrum allocation. In *Dynamic Spectrum Access Networks (DYSPAN), 2014 IEEE International Symposium on*, pages 404–415, April 2014.

[16] Peter Todd. Timelock: time-release encryption incentivised by Bitcoins, June 4th 2014. Available online, June 5th 2014: `https://www.mail-archive.com/bitcoin-development@lists.sourceforge.net/msg05547.html`.

[17] Mahdi Zamani, Jared Saia, Mahnush Movahedi, and Joud Khoury. Towards provably-secure scalable anonymous broadcast. In *Presented as part of the 3rd USENIX Workshop on Free and Open Communications on the Internet*, Berkeley, CA, 2013. USENIX.