RN/06/04
March 14, 2006

Research
Note

Censorship-Resistant Communication over Public Networks

Michael Rogers

Electronic Mail: m.rogers@cs.ucl.ac.uk
URL: http://www.cs.ucl.ac.uk/staff/M.Rogers/

Abstract

The rapid growth of peer-to-peer networks and social networking websites has demonstrated the internet's potential as a medium for grassroots collaboration. This report describes ongoing research into the use of friend-to-friend overlay networks for censorship-resistant communication. Decentralised mechanisms for resource allocation, unforgeable acknowledgements and adaptive routing are proposed.

Department of Computer Science
University College London
Gower Street
London WC1E 6BT, UK

Contents

1 Introduction

In times of universal deceit, telling the truth will be a revolutionary act. – George Orwell

Censorship Resistance

The goal of this work is censorship-resistant communication. Censorship resistance can be defined as the combination of privacy, unlinkability, and robustness. Privacy means that nobody intercepting a message should be able to discover the contents of the message; unlinkability means that nobody should be able to determine whether two people are communicating with one another beyond the fact that they are each communicating with someone; and robustness means that nobody should able to prevent two people from communicating with one another if they both wish to do so.

The Automation of Surveillance

Increasing reliance on communication technology in all areas of society has multiplied the opportunities for government surveillance, and for the covert collection of personal information by companies and individuals. Agencies that were once obliged to be selective in their intelligence-gathering can now use computers to conduct surveillance cheaply and automatically, on an unprecedented scale.

Privacy-enhancing communication technologies can help to redress the balance by concealing from an eavesdropper the identities of the communicating parties and the nature (including the content, size, timing, and structure) of their communication.

Anonymous and pseudonymous communication tools can also enable communicating parties to conceal their identities from one another, which can protect journalistic sources, whistle-blowers and political dissidents, as well as maintaining personal privacy.

The Decentralisation of Communication

The rapid growth of peer-to-peer networks and internet social networks has demonstrated the internet's potential as a medium for grassroots collaboration. When compared with the dominant media of just ten years ago – television, radio, and newspapers – the web is radically inclusive and decentralised. Peer-to-peer networks are more decentralised still, and with the development of friend-to-friend networks [29, 109] where users only connect to people they know, the social connections between users have become a vital part of the medium itself.

Marshall McLuhan [177] famously claimed that "the medium is the message": the most important aspect of a communication medium is not the content it carries, but the mode of interaction it creates. If television's mode of interaction is passive, attentive, and absorbing, then a friend-to-friend network's mode of interaction is social, collaborative, and collusive. If television draws attention – and authority – to the centre, then friend-to-friend networks draw it to the edge, and perhaps out of public view altogether.

Applying a friend-to-friend structure to private communication could create a medium that, like the *samizdat* network in the Soviet Union [251, 20], would be almost impossible to comprehensively monitor, filter, or suppress.

2 Thesis of Research

The thesis of this work is that **censorship-resistant communication is possible over public networks such as the internet, if users relay messages for one another in a friend-to-friend overlay network. Sustainable resource usage can be encouraged in such a network without central control or coordination. Given certain assumptions about the overlay topology, robust and unlinkable communication does not require a public key infrastructure.**

2.1 Problem Domain

This research connects three active and rapidly expanding fields: cooperation in decentralised networks, unlinkable communication, and routing in dynamic networks. It also has links to game theory, estimation theory, and the study of social networks.

2.2 Assumptions

The term "censorship" suggests the action of a government or other powerful organisation, and any discussion of censorship-resistance must assume the existence of a powerful and well-funded adversary. The attacker may aim to prevent certain individuals from communicating, or may aim to shut down the network entirely. This still leaves a wide range of possible threat models, which can be classified according to whether the attacker is internal or external to the system, passive or active, and static or adaptive [210]. Attackers can be divided into those who are able to monitor all communication links, known as global eavesdroppers, and those who can only monitor certain links, known as local eavesdroppers [175].

In this work we make the pessimistic assumption of an internal, active, adaptive attacker with global eavesdropping capabilities. We also assume the attacker can identify the owner of a node given its network address. However, it is assumed that the attacker is unable or unwilling simply to deny access to public communication networks, or arbitrarily to disrupt communication across those networks – attacks must be targetted and limited in scope. A second optimistic assumption is that users are able to keep their computers secure and their cryptographic keys secret.

Even if we assume the existence of a strong attacker, we should not ignore the possibility of weaker attackers. A global eavesdropper can see all communication and must therefore be assumed to know the structure and membership of any overlay network, but it is possible and desirable to hide this information from those without global eavesdropping capabilities – structural information may help an attacker to identify weak points in the network [7], while a membership list can be used to harass or threaten users.

A different kind of threat comes from the selfish behaviour of the users themselves. A peer-to-peer network is a public good, where private contributions lead to collective benefits. Unfortunately, the difficulty of encouraging responsible use of public goods is well known: each user has an incentive to enjoy the contributions of others without making a contribution in return [114, 66]. Our threat model must therefore include "free riding" and other forms of selfish behaviour. In general, it should not be assumed that users will follow any step of a protocol unless it is in their interest to do so [190].

The problem of resource allocation does not disappear even if we assume that a few altruistic users donate enough resources to compensate for free riders: an attacker who can monopolise these resources may be able to deny service to other users.

Some assumptions must also be made about the social relationships between users. In a friend-to-friend network, direct connections are only made between users who know one another. This means the structure of the communication network will reflect the network of social ties between users. Of course, not all social connections will be reflected in the communication network, and indeed unlinkable communication would be impossible under a global eavesdropper if that were the case. Nevertheless it seems reasonable to assume that users who wish to communicate will generally be closer to one another in the social network – and therefore closer in the communication network – than users chosen at random.

The structure of technologically mediated social networks is a matter of ongoing research [76, 129, 178, 41], but for our purposes it is sufficient to assume two properties: an average distance between nodes that grows logarithmically in the number of nodes, and a low probability that the network will be disconnected by the removal of randomly-chosen nodes. These properties are shared by small world networks [261, 147], scale-free networks [22, 7], and random networks [28], among others.

2.3 Contributions

This work offers three contributions to the field of censorship-resistant communication:

1. A mechanism to encourage users of decentralised networks to contribute resources equal to those they consume
2. A way to produce unforgeable acknowledgements for datagrams, without a public key infrastructure
3. Two adaptive, anonymous routing schemes for dynamic networks, based on unforgeable acknowledgements

3 State of the Art

This section briefly reviews the state of the art in five relevant fields: anonymity and unlinkability; robust storage and routing; decentralised communication networks; cooperation in decentralised networks; and the structure of social networks. A more detailed review of each area is given in Appendix A.

3.1 Anonymity and Unlinkability

Much of the research in anonymous communication stems from Chaum's 1981 *mix network* proposal [44]. A mix attempts to hide the relationship between incoming and outgoing messages by decrypting, reordering, and delaying them. *Onion encryption* can be used to send a message through a chain of mixes without any single party (except the sender) learning the identities of both the sender and receiver - this property is called unlinkability [203]. Mix networks also provide anonymity for senders, but not for recipients.

The degree of anonymity and unlinkability provided by various types of high-latency mixes has been well studied [56, 144, 134, 24, 210, 233, 63, 64, 75], usually under the assumption of an active, internal, adaptive attacker who is also a global eavesdropper.

Indirection and onion encryption can also be applied to low-latency communication, although it is not practical to delay or reorder messages in this case. Low-latency mix networks typically use virtual circuits or *tunnels* rather than routing each message independently [106, 74]. Traffic analysis of low-latency mixes is much easier, because encryption does not hide timing patterns from a passive eavesdropper [275, 197, 164]. Traffic analysis can also be combined with active attacks [19, 186].

Some circuit-based mix networks combine sender anonymity with recipient pseudonymity by allowing recipients to build tunnels to 'rendezvous points' or 'gateways' that can be contacted by anonymous senders who know the recipient's public key.

In peer-to-peer mix networks, users run their own mixes and provide anonymity for one another rather than relying on a central set of servers [97, 212, 128]. Secure peer discovery and key distribution are the major difficulties with this approach.

Some peer-to-peer anonymising networks use indirection without onion encryption, avoiding some of the problems of key distribution. Some of these networks anonymise existing protocols [211, 131], some anonymise end-to-end communication between peers [187, 21], and others anonymise access to a decentralised, distributed cache [53, 23]. The anonymity provided by these systems is generally weaker than that provided by mix networks [68, 154], as they do not attempt to protect against global eavesdroppers.

Anonymous broadcast networks [45, 77, 104, 235] use broadcast groups rather than indirection to provide anonymity. This limits scalability and makes it possible to jam the anonymous channel.

One of the weaknesses of peer-to-peer anonymising networks is that any user can learn the addresses of a large number of other users. This 'address harvesting' could provide information for intersection attacks or traffic blocking, and in some circumstances mere participation in an anonymous communication network might be regarded as suspicious.

3.2 Robust Storage and Routing

Censorship resistance can be seen as an extreme example of the well-studied problem of availability: that is, ensuring that information remains available in the face of intentional attacks and accidental failures.

Robust storage systems replicate data across a large number of servers, which can be centrally managed [79, 5, 60, 120] or autonomous [123, 10, 259, 258, 182].

Robust link state [117, 49, 195], distance vector [240, 272, 124], and source routing protocols [125, 194, 225, 143] aim to prevent malicious nodes from interfering with route discovery, while Byzantine robust routing protocols [200, 15, 16] have the stronger aim of guaranteeing the delivery of data packets if there is a fault-free path between the source and the destination.

3.3 Decentralised Communication Networks

Peer-to-peer overlays and mobile ad hoc networks have been the focus of much recent research. These decentralised networks share three characteristics that set them apart from traditional communication networks: open membership, autonomous nodes, and the absence of central administration. The resulting issues of self-organisation, scalability and stability are relevant to the design of censorship-resistant networks.

Mobile Ad Hoc Networks

A mobile ad hoc network is a group of wireless nodes that communicate among themselves without infrastructure or manual configuration. Either routes must be discovered on demand, or changes in the topology must be broadcast to all interested nodes [199, 122].

The scalability of ad hoc routing can be improved through location awareness [150, 142, 266], virtual coordinates [48], or efficient flooding [47, 198, 208, 65, 265].

Peer-to-Peer Networks

A peer-to-peer network is an internet overlay composed of connections between autonomous peers. Most peer-to-peer systems construct a multi-hop search overlay to enable peers that wish to communicate to find one another [216]. Other systems use central servers [6, 156, 54], but unlike client-server systems the peers communicate with one another as well as with the server.

Peer-to-peer search overlays can be divided into two major families [168]. In the first family, often referred to as unstructured peer-to-peer networks, the topology of the overlay is only loosely controlled and searches are nondeterministic: matching items are not guaranteed to be found [103, 100, 53, 187]. Probabilistic search [169, 170, 252, 227, 90], virtual coordinates [183], efficient flooding [179, 137, 220, 93, 136], and two-tier architectures [132, 100, 269] have been proposed to improve scalability.

In the second family, referred to as structured peer-to-peer networks, tighter control of the overlay topology allows scalable, deterministic searches [276, 80, 209, 246, 176, 173, 116].

Private peer-to-peer networks [253, 247, 112, 236, 84] allow encrypted communication within small, invitation-only groups. Friend-to-friend networks [29, 109, 205, 52, 224] only permit direct connections between friends.

3.4 Cooperation in Decentralised Networks

Any system in which the infrastructure is provided collectively by the users faces the problem of encouraging users to contribute resources as well as consuming them. In other words, users must cooperate with one another. This problem can also be seen as an aspect of robustness [190], since many denial of service attacks work by exhausting resources.

Four main approaches have been taken to this problem:

1. **Micropayments:** nodes pay one another for services using a digital currency [36, 107, 255, 9]. To prevent double spending, micropayment systems require either tamper-resistant hardware or a central bank.
2. **Audits:** nodes test one another and exclude misbehaving nodes from the network [189, 171, 113]. Excluded nodes must not be able to re-enter the network under new identities, so audit systems require certified identities or other barriers to entry.
3. **Reputations:** nodes publicise the outcomes of their interactions, and combine other nodes' reports with their own observations when deciding whether to cooperate [174, 1, 140, 33, 34]. Like audits, reputations require limits on the creation of identities.
4. **Reciprocation:** pairs of nodes dynamically adjust the level of service they offer each other based on the level of service received [54, 156, 57, 249, 2, 88, 250]. Reciprocation does not require any centralised infrastructure.

3.5 Social Networks

The study of the structure of social networks has accelerated in recent years as computers have made it possible to gather and analyse large datasets with relative ease, especially for technologically mediated social networks [76, 129, 178, 41]. Several graph theoretic models of social networks have been proposed [261, 22, 188], and the question of whether members of large social networks tend to be linked by short chains of acquaintances – the so-called 'small world problem' [180, 149, 76] – has received particular attention [262, 261, 146, 147, 148, 94].

The structural properties of social networks will affect the scalability [224] and resilience [7, 184, 185] of friend-to-friend communication.

3.6 Drawbacks of Existing Systems

Anonymous communication systems that use layered encryption face the problem of distributing the public keys of mixes in a robust and tamper-proof way. Some use trusted directory servers [74, 131], creating a central point of failure. A list of active users can be obtained by eavesdropping on the directory servers, which enables intersection

attacks without global eavesdropping. This is also true of systems that use central servers for peer discovery and relaying [211, 6, 112, 236].

Tarzan [97], MorphMix [212], and I2P [128] use peer-to-peer techniques for key distribution, but these require every user to be able to learn the address of every other user, which allows intersection attacks and makes it possible to harass or threaten users.

'Address harvesting' is also possible in anonymising networks that do not use layered encryption [53, 23, 187], and in private peer-to-peer networks that allow any member of a group to connect to any other [253, 247, 112, 84].

Many of the proposals for encouraging resource contribution in decentralised networks are unsuitable for censorship-resistant communication. Reputations and audits require global identities, and necessarily reveal information about usage patterns, while micropayments require either a central bank [9] or trusted hardware [36].

4 Overview of the Proposed Design

4.1 Friend-to-Friend Communication

A friend-to-friend network, in which users only connect to people they know and trust, provides limited scope for address harvesting without requiring highly available, universally trusted servers.

Unfortunately, onion encryption [44] is not possible in friend-to-friend networks, because it requires the sender to discover the public key of each relay in advance, contradicting the requirement that users only communicate directly with people they trust. (Discovering the keys indirectly without a trusted certificate authority would allow man-in-the-middle attacks.) Therefore the design proposed here does not use onion encryption – if an attacker controls more than one node on the path taken by a packet, the attacker's nodes will be able to see that they are on the same path. This may help them to identify the sender or recipient of the packet. They cannot perform a predecessor attack [264], however, because each node consistently connects to the same neighbours.

Packets should be padded to a fixed size to prevent external eavesdroppers from performing traffic analysis based on packet sizes [119]. Random delays do not prevent timing analysis [197], but quantising the transmission time is an alternative approach that deserves further investigation. Cover traffic is feasible given the restricted topology; combined with quantised transmission times this would produce a completely opaque traffic pattern.

The network should offer the simplest and most general service possible, minimising the complexity of relays and allowing new applications to be deployed incrementally [223]. The proposed design provides an unreliable datagram service similar to that provided by the internet protocol.

4.2 Reciprocation

The problem of encouraging users to contribute resources to the network was described in Section A.3. Reciprocation is a suitable approach for a friend-to-friend network because it relies on relatively long-term interactions between neighbours, but does not require certified identities or central record-keeping. Reciprocation is rational behaviour for selfish nodes, and encourages them to contribute resources to the network. It may also help to prevent denial of service attacks, since an attacker will not be able to consume the network's resources without contributing resources in return. The details of the proposed mechanism are described in Section B.1.

4.3 Unforgeable Acknowledgements

The unforgeable acknowledgement scheme decribed in Section B.3 enables relays in a multi-hop network to verify the end-to-end delivery of the packets they forward, without knowing the identities of the source or the destination. Nodes can use this information to find reliable routes and to measure the level of service received from their neighbours, which is crucial for reciprocation.

4.4 Anonymous Routing

In most routing protocols, packets are labelled with a destination address. Even in overlay networks where the destination's IP address may be unknown to the sender, packets generally carry some form of overlay address. This is a disadvantage from the point of view of unlinkability, because relays can see when a destination is receiving packets: the destination is pseudonymous rather than anonymous.

Onion encryption allows the recipient's address to be hidden from relays, but the sender must obtain each relay's public key and decide the route in advance. This means that the membership and topology of the network must be common knowledge, which would negate the advantages of the friend-to-friend approach.

Another problem is that in a network with open membership, relays cannot necessarily be trusted to forward packets, so discovering the topology is only the first step to finding a reliable route [16]. Indeed, the Sybil attack [78] calls into question the very notion of topology discovery in an open network, since a single entity may be able to impersonate an arbitrary number of nodes and links.

Rather than attempting to find routes across a topology that may or may not exist, we propose an *evidence-based* approach to routing, using proof of reliability to guide forwarding decisions. From a pragmatic point of view, all that matters when deciding whether or not to forward a packet is whether the next hop is likely to deliver the packet. It is not necessary to speculate about the reasons for packet loss, and indeed attempting to do so may allow dishonest users to manipulate the system.

In evidence-based routing, routes are discovered on demand, and forwarding decisions are based on constant feedback from the destination. Section B.4 describes two schemes for evidence-based routing that make use of reciprocation and unforgeable acknowledgements.

4.5 Identity Management

A friend-to-friend network involves three related but distinct kinds of actors: users, identities, and nodes.

A user can have one or more identities; each identity has a nickname, a public/private key pair, and a list of friends, each with a nickname and a public key. The list of friends may be public or private.

A node is a piece of software that exchanges packets with other nodes on behalf of one or more identities. Each node has a public/private key pair and a list of neighbours, which are nodes from which traffic will be accepted, each with an associated public key and one or more link addresses (eg IP address, telephone number or MAC address). Link addresses are not relied on for authentication, but they are used to make initial contact so that authentication can take place. If a link address changes (for example a dynamic IP address), neighbours may need to contact one another indirectly through the overlay to exchange up-to-date link addresses before making a direct connection.

Neighbours are not the same as friends – neighbour relationships exist between nodes and are visible to an eavesdropper, whereas friend relationships exist between identities and might not involve direct communication. Thus *friends can remain unlinkable, whereas neighbours cannot*.

Friends must agree on encryption and authentication keys before they can communicate. The Diffie-Hellman key agreement protocol [72] allows a shared secret to be agreed in the presence of eavesdroppers, but it is vulnerable to a man-in-the-middle attack, so one friend must first obtain the hash of the other's public key, either face-to-face or through a mutually trusted third party. Likewise, neighbour keys can be exchanged in person or through a trusted third party.

Multiple identities per user, lists of friends, and introductions through mutual friends are already familiar concepts to users of instant messaging software and social networking websites. A similar interface seems appropriate for friend-to-friend communication software. Third-party introductions can be made almost automatic, for example by embedding a friend's public key in a message whenever the friend's nickname is mentioned [245]. However, the distinction between friends and neighbours and the possibility of a man-in-the-middle attack during third-party introductions must be made clear to users.

5 Proposed Experiments

5.1 Hypotheses

The thesis stated in Section 2 rests on five hypotheses:

1. When selfish individuals interact repeatedly, they can sustain cooperation without central coordination or control, avoiding the "tragedy of the commons".
2. Given a method of producing unforgeable acknowledgements, reciprocation between immediate neighbours can support interactions that require the cooperation of a chain of individuals, such as multi-hop packet forwarding.

3. Adaptive routing based on unforgeable acknowledgements ("evidence-based routing") can operate without location-awareness or global identifiers, and scales better than simple flooding.

4. Evidence-based routing is robust to a wide range of attacks and failures, without requiring a public key infrastructure.

5. Evidence-based routing can support unlinkable communication.

5.2 Experimental Method

The five hypotheses will be tested using simulations. This will require some modelling assumptions to be made, but unfortunately the scale of peer-to-peer networks makes it impractical to deploy a live experimental network, and the anonymity requirements would make such a network difficult to study.

The initial experiments will use a simple round-based simulator. Later experiments requiring a greater level of realism will use a packet-level discrete event simulator such as OMNet++ [193].

5.3 Reciprocation

The first set of experiments will test the reciprocation mechanism described in Section B.1. Briefly, each node evaluates the requests it receives according to the amount of reciprocation it expects to obtain by serving each request, and serves the request with the highest expected utility. This mechanism is based on results from game theory showing that when the same agents play one another repeatedly, conditioning their actions on the outcomes of previous games, mutual cooperation can emerge in games where the most profitable short-term choice would be to exploit the cooperation of the other player [155, 17, 218].

The experimental setting is a file sharing network in which nodes download pieces of a file from one another [54]. There are three types of node: altruists, free riders, and reciprocators. Altruists upload pieces to their neighbours in round-robin order; free riders never upload; and reciprocators upload to the neighbour that is expected to provide the largest number of downloads in return, based on past experience. This does not mean always uploading to the same neighbour, because uploading to a neighbour reduces its download/upload ratio, but it does mean that free riders will only receive a small number of uploads before they are passed over in favour of more cooperative neighbours.

The experiments measure the number of downloads received by nodes of each type and by nodes overall, when the population contains various proportions of altruists, free riders and reciprocators.

Expected results:

1. Reciprocators will receive at least as many downloads as free riders under all conditions

2. The overall number of downloads received will increase with the number of reciprocators, when holding the number of altruists constant

The first point demonstrates the incentive for free riders to switch to reciprocation, and the second point demonstrates the public benefit when free riders switch to reciprocation.

Preliminary results from these experiments are given in Section 6.

5.4 Multi-Hop Cooperation

The second set of experiments will test whether reciprocation between immediate neighbours can support interactions that require the cooperation of a chain of nodes.

The experimental setting is a multi-hop datagram network. The source and destination of each packet may be separated by one or more intermediate nodes. The route from the source to the destination is chosen by an oracle. Each successful delivery results in an acknowledgement that is forwarded back along the route taken by the packet.

As before, nodes are divided into altruists, free riders and reciprocators. Altruists forward packets in the order they arrive, treating their own packets in the same way as other nodes' packets. Incoming packets are dropped if there is no room in the queue. Free riders never forward packets, but they transmit their own packets and acknowledge the packets they receive.

Reciprocators derive utility from acknowledgements for their own packets, and always transmit the packet that is expected to obtain the most utility. Utility can be obtained directly in the case of the node's own packets, or indirectly in the case of packets forwarded for a cooperative neighbour – acknowledgements for a neighbour's packets can have indirect value if they are expected to result in reciprocal delivery of the node's own packets. Reciprocators keep packets in a priority queue sorted by expected utility, and the packet with the lowest expected utility is dropped if there is no room in the queue. In this initial experiment, reliability is not taken into account when calculating expected utility.

The experiments measure the number of acknowledgements received by nodes of each type and by nodes overall, for various proportions of altruists, free riders and reciprocators.

Expected results:

1. Reciprocators will receive at least as many acknowledgements as free riders under all conditions
2. The overall number of acknowledgements received will increase with the number of reciprocators, when holding the number of altruists constant

5.5 Evidence-Based Routing

These experiments will test the ability of nodes to discover reliable routes by a process of local adaptation. Whereas in the previous experiment routes were chosen by an oracle, in this experiment the only information used for route discovery and route maintenance is feedback in the form of acknowledgements. Each node uses an exponentially weighted moving average to keep track of each neighbour's reliability with respect to each destination address. When a neighbour's reliability for a particular address is unknown, it is estimated using the neighbour's overall reliability for packets with unknown addresses, again using an EWMA.

As before, altruists forward packets in order of arrival, free riders do not forward packets, and reciprocators forward packets in decreasing order of expected utility. Reciprocators take reliability into account when calculating a packet's expected utility, resulting in local adaptation as reliable routes are given priority.

Local adaptation will be compared with two control conditions: flooding and random forwarding. Delivery ratios for routes of different lengths will be compared across conditions. In all three conditions duplicate packets are dropped, but if an acknowledgement is received it is forwarded to every neighbour that sent a copy of the packet.

Expected results:

1. Delivery ratios with local adaptation will be higher than with flooding or random forwarding
2. The improvement will be correlated with the average node lifetime

The effect of varying the decay constant of the EWMA will also be investigated.

Expected results:

1. The decay constant used for reliability measurement will affect the delivery ratio
2. The optimal value of the decay constant will be correlated with the average node lifetime

5.6 Anonymous Datagrams

These experiments will test the efficacy of local adaptation when packets do not carry destination addresses. Under these conditions, the only pieces of information on which to base an estimate of reliability are the packet's previous hop and next hop. Each node uses an exponentially weighted moving average to keep track of each next hop's reliability with respect to packets from each previous hop, and with respect to the node's own packets.

Each packet is sent from a randomly chosen source to a randomly chosen destination. The delivery ratio with local adaptation will be compared to flooding and random forwarding, for routes of different lengths.

Expected results:

1. For local adaptation, delivery ratios will be lower than they were with addresses
2. For flooding and random forwarding, delivery ratios will be the same as they were with addresses
3. As before, delivery ratios with local adaptation will be higher than with flooding or random forwarding, and the improvement will be correlated with the average node lifetime
4. As before, the optimal value of the EWMA decay constant will be correlated with the average node lifetime

5.7 Flow Length

Whereas in the previous experiment the source and destination of each packet were independent, in this experiment each randomly chosen source sends an exponentially distributed number of packets to the same randomly chosen destination. This is expected to allow more effective local adaptation, as the reliability of previous packets will be a better predictor of the reliability of future packets.

Expected results:

1. Delivery ratios will be higher than they were for datagram
2. The improvement will be proportionally larger for longer flows
3. The improvement will be correlated with the average node lifetime

5.8 Anonymous Flows

These experiments will investigate the effect of adding link-local flow identifiers to packets with the same source and destination, as described in Section B.4.2. As before, flow lengths are exponentially distributed.

Each node uses an exponentially weighted moving average to keep track of each neighbour's reliability with respect to each flow. When a neighbour's reliability for a particular flow is unknown, it is estimated using the neighbour's overall reliability for packets from unknown flows, again using an EWMA.

Expected results:

1. Delivery ratios will be higher than they were for anonymous flows
2. As before, the improvement will be proportionally larger for longer flows, and will be correlated with the average node lifetime

5.9 Robustness

This set of experiments will test the ability of evidence-based routing to cope with malicious nodes. Attackers on the path of a flow will attempt to prevent the flow from reaching its destination in four ways:

1. Filtering: the attacker drops some or all of the packets in the flow
2. Poisoning: the attacker adds to the flow packets that the destination is unable to acknowledge
3. Diverting: the attacker adds to the flow packets that the destination is unable to acknowledge, but another attacker off the path is able to acknowledge, in an attempt to divert the flow to the second attacker
4. Truncating: the attacker delivers the first few packets of the flow and then drops the rest

In each case, the delivery ratio and interarrival variance of attacked flows will be compared with those of unattacked flows. The overall delivery ratio and interarrival variance will be measured for various proportions of attackers.

Expected results:

1. All attacks will decrease the delivery ratio and increase the interarrival variance of the attacked flows
2. Poisoning or diverting a flow will have the same effect as filtering from the point of view of upstream nodes

3. Filtering or truncating a large proportion of flows will cause the attacker to be avoided due to local adaptation

4. If the proportion of attackers is small enough and the average node lifetime is long enough:

    - Local adaptation will cause attackers to be avoided
    - The damage caused to a node's delivery ratio and interarrival variance will depend on its proximity to the nearest attacker

It is very unlikely that local adaptation will be able to resist every kind of deliberate manipulation, but it may be possible to confine the damage to nodes near the attackers. In a friend-to-friend network, these nodes belong to friends (or dupes) of the attackers.

5.10 Unlinkability

If the previous experiments have been successful and reciprocation can support robust multi-hop communication, it will be time to consider the degree of unlinkability such communication can provide. How much an attacker can learn will depend on the dynamics of evidence-based routing, so a quantitative assessment of possible attacks will require a detailed packet-level simulation, with realistic models of traffic, churn, and network structure. For the moment, a qualitative discussion of some possible attacks is given in Appendix C.

6 Preliminary Results

This section presents preliminary results from the experiments described in Section 5.3. The experiments used a round-based simulator, with 1000 nodes updated in a random order each round. Node lifetimes were exponentially distributed with a mean of 100 rounds; each new node was attached to 4 existing nodes chosen uniformly at random. Recording did not start until the network reached its full size. Results were averaged over 20 runs of 200 rounds each.

Qualitative Analysis

The triangular plots in Figure 1 give a qualitative indication of the effects of free riding, altruism and reciprocation on the number of downloads per node.

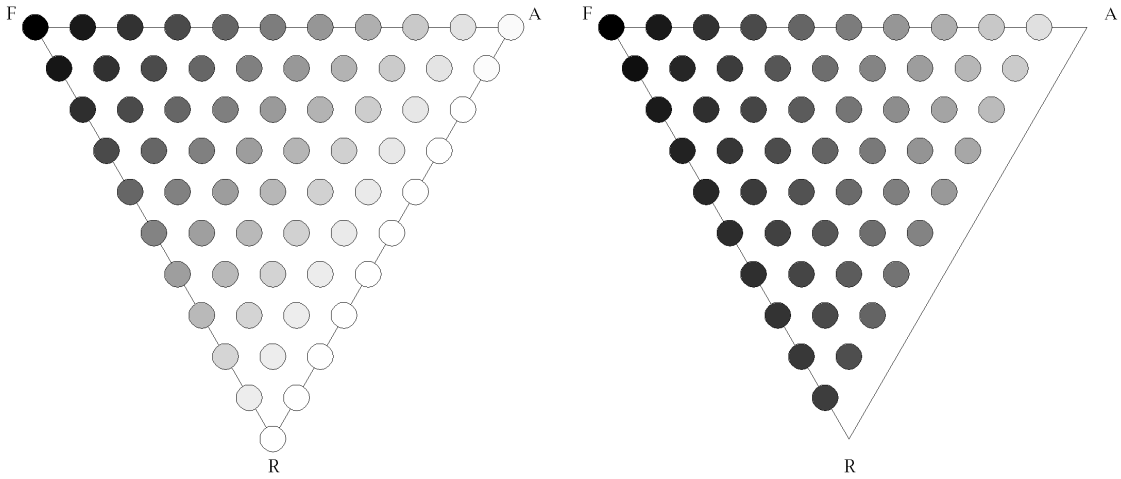
Figure 1(a) shows the average number of downloads per node in populations containing various proportions of free riders, altruists and reciprocators. Each point in the triangle represents a different mixture of the three types. At the corner labelled **F** the population consists entirely of free riders, while along the opposite edge **AR** there are no free riders in the population. Similarly, at **A** all nodes are altruists, while along **FR** there are no altruists; and at **R** all nodes are reciprocators, while along **FA** there are no reciprocators. The centre of the triangle represents an equal mix of the three types.

The colour of each circle represents the average number of downloads per node. Each node can upload at most once per round, so the average number of downloads per node can vary from 0 to 1, depending on the number of nodes that choose to upload. Black represents zero downloads, and white represents the maximum of one download per node per round.

The effect of free riding is immediately visible in Figure 1(a): the circles are much darker near **F** than along the opposite edge, indicating that fewer downloads are received overall when the population contains a large proportion of free riders.

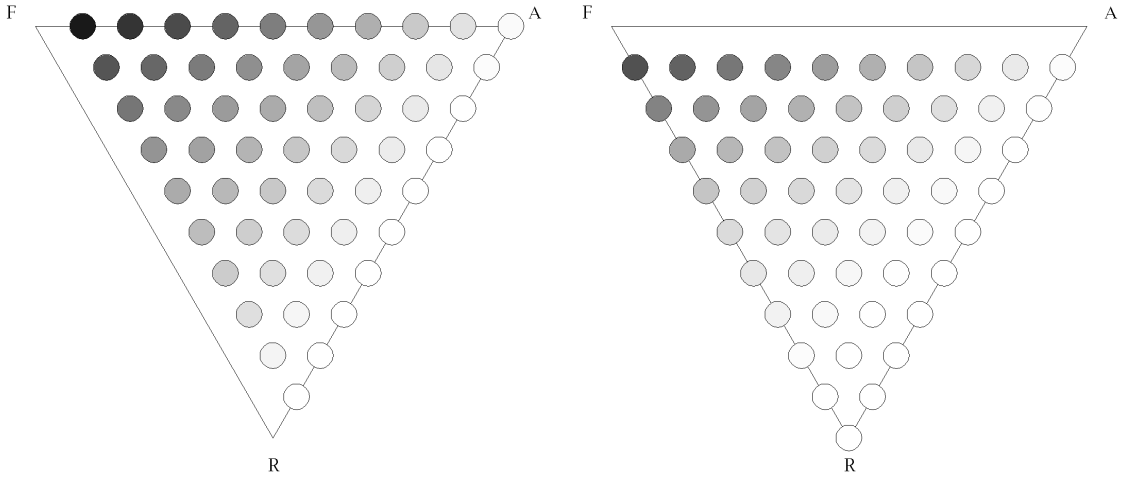
Figures 1(b) to 1(d) give a more detailed breakdown of the same set of results. Figure 1(b) shows the number of downloads received by free riders, Figure 1(c) shows the number of downloads received by altruists, and Figure 1(d) shows the number of downloads received by reciprocators. No circles are shown where the type in question is not represented in the population, so each figure has one edge without circles.

Figure 1(b) is relatively dark except near **A**, indicating that free riders receive few downloads except when the population contains a large proportion of altruists. In contrast, Figures 1(c) and 1(d) are consistently light from **A** to **R**, indicating that altruists do well in a population dominated by reciprocators and vice versa. The area around **F** is dark in all three figures: no type does well in a population dominated by free riders.



(a) Number of downloads received by nodes overall

(b) Number of downloads received by free riders



(c) Number of downloads received by altruists

(d) Number of downloads received by reciprocators

Figure 1: The average number of downloads per node in populations containing various proportions of free riders, altruists and reciprocators. Black is the minimum (zero downloads) and white is the maximum (one download per node per round). Figure 1(a) shows the average number of downloads overall, and Figures 1(b) to 1(d) show the number of downloads received by nodes of each type. No circles are shown where the type in question was not represented in the population.

Quantitative Analysis

The predicted results were that the population as a whole would benefit from free riders switching to reciprocation, and that free riders themselves would have an incentive to switch.

By following the line from **F** to **R** or any line parallel to it in Figure 1(a), it can be seen that when the number of altruists is held constant, the number of downloads increases with the number of reciprocators. For every line parallel to **FR**, the number of downloads is highly correlated with the proportion of reciprocators ($r > 0.988$, Pearson product-moment correlation coefficient). Given the number of samples, this level of correlation is highly significant ($P < 0.0001$, Student's t test, two-tailed[1]). This shows that the network as a whole benefits from free riders switching to reciprocation.

To see whether free riders have an incentive to switch to reciprocation, the number of downloads received by free riders can be compared with the number of downloads received by reciprocators under the same conditions. For all conditions where both types are present in the population, reciprocators receive significantly more downloads than free riders ($P < 0.01$, Mann-Whitney U test, two-tailed[2]).

Similar results were seen when the number of nodes, mean lifetime and mean degree were varied. Results were also similar when the degree of new nodes was normally or exponentially distributed rather than constant, and when the upload capacity of new nodes was normally or exponentially distributed rather than constant. The parameter space has not been fully explored, but these initial results show that reciprocation can be effective in heterogeneous networks, which will be important for any real application.

7 Conclusions

The spread of digital communication networks and advances in surveillance technology have placed our privacy under increasing threat. This report has described the problem of censorship-resistant communication, reviewed existing solutions, and proposed a new approach with the following key points:

- Users construct a friend-to-friend overlay to prevent address harvesting without relying on central servers
- Reciprocation encourages users to contribute resources to the network, and restricts the resources available to attackers
- Unforgeable acknowledgements are used to measure reliability
- Evidence-based routing attempts to find reliable routes across a hostile network, without revealing the names or pseudonyms of the communicating parties, and without requiring a public key infrastructure

Most of the proposed ideas have not been rigorously tested – the next step is to complete the programme of experiments outlined in Section 5. The proposed experiments can all be carried out using hardware available within the department, and freely available software. Anonymised data collected from the LiveJournal website [178, 270] and the PGP web of trust [41] may be used in some of the larger simulations.

There is also a significant amount of work to be done on reliability estimation, robust transport protocols, and a formal model of utility-based reciprocation, but these are fairly self-contained tasks that can be tackled in future work.

7.1 Draft Table of Contents for Thesis

1. Introduction (based on Sections 1 and 2 of this report)
    - Problem Domain: Censorship-Resistant Communication
    - Thesis of Research
    - Assumptions
    - Contributions

[1] The t statistic was used to test the strength of the correlations rather than to compare the number of downloads received by nodes of different types.

[2] Student's t test is not suitable for comparing reciprocators to free riders because they are not independently and randomly drawn from the same normally distributed population. The Mann-Whitney U test was chosen because it makes fewer assumptions about the population distribution, which in this case is bimodal.

2. State of the Art (based on Appendix A of this report)
    - Anonymity and Unlinkability
    - Robust Storage and Routing
    - Decentralised Communication Networks
    - Cooperation in Decentralised Networks
    - Social Networks
3. Proposed Design (based on Appendix B of this report)
    - Requirements
        - Unlinkability
        - Robustness
        - Scalability
    - Details of Design
        - Overview
        - Identity Management
        - Utility-Based Reciprocation
        - Unforgeable Acknowledgements
        - Evidence-Based Routing
        - Anonymous Flows
        - Estimating Reliability
4. Experiments (partly based on Section 5 of this report)
    - Experimental Method
    - Details of Experiments
        - Reciprocation
        - Multi-Hop Cooperation
        - Evidence-Based Routing
            * Anonymous Datagrams
            * Flow Length
            * Anonymous Flows
        - Robustness
        - Unlinkability
5. Analysis of Results
    - Unlinkability (metric: entropy of the anonymity set)
    - Robustness (metrics: delivery ratio, interarrival variance)
    - Scalability (metric: asymptotic message complexity)
6. Critical Assessment
    - Suitability of the Chosen Approach
    - Discussion of Tradeoffs
        - Unlinkability
        - Robustness
        - Scalability
    - Comparison with Existing Systems
        - Peer-to-Peer Mix Networks
        - MUTE
        - WASTE
        - Freenet 0.7

7. Conclusions and Future Work
    - Estimating Reliability
    - Transport Protocols for Censorship-Resistant Communication
    - A Formal Model of Utility-Based Reciprocation
8. Bibliography

7.2 Work Plan

March: Reciprocation experiments. Submit a short paper on unforgeable acknowledgements to PET 2006 (see the accompanying research note [219]).

April: Multi-hop cooperation and routing experiments. If these are successful, submit a paper on evidence-based routing to P2P 2006.

May: Write a detailed description of the proposed design based on Appendix B. This will form chapter 3 of the thesis (Proposed Design).

June: Study the dynamics of evidence-based routing with and without flow identifiers. Write a plan for simulating attacks on reliability and anonymity. Together with Section 5, this will form chapter 4 of the thesis (Experiments).

July-August: Packet-level simulation of a friend-to-friend overlay, possibly using OMNeT++. Develop realistic models for churn, topology, and end-to-end traffic representing email, instant messaging, file transfer and voice. For each application, evaluate the network's robustness and anonymity from the point of view of local and global attackers.

September: Write up the results of the packet-level simulation. Together with the results of the reciprocation and routing experiments, this will form chapter 5 of the thesis (Analysis of Results).

October: Review the state of the art and update the literature review in Appendix A to form chapter 2 of the thesis (State of the Art).

November: Write chapter 6 of the thesis (Critical Assessment), including a comparison with Freenet 0.7 if it has been deployed.

December: Write the final chapter of the thesis (Conclusions and Future Work). Aim to submit by the end of 2006.

A Literature Review

A.1 Anonymity and Unlinkability

A communication system is said to provide sender anonymity if the recipient of a message cannot determine the identity of the sender, and recipient anonymity if the sender cannot determine the identity of the recipient. Unlinkability means that no observer can determine whether a given sender and recipient have communicated [203].

Many so-called anonymous communication systems are actually pseudonymous: users are identified by cryptographic pseudonyms instead of their real names. The difference between pseudonymity and anonymity is important because of two statistical attacks: the intersection attack [175] and the predecessor attack [264]. In the intersection attack, an eavesdropper compiles a list of the users who are active each time a certain pseudonym is active. The lists are intersected to find users who are likely to own the pseudonym. Alternatively, intersecting the lists of users who are active at the same time can reveal pairs of users who are likely to be communicating.

The predecessor attack is used against systems where the path of each message (or circuit) is chosen at random. The attacker identifies several messages sent or received by the same pseudonym. The owner of the pseudonym must appear on the paths of all the messages, whereas each other node will tend to appear less frequently. If each attacker-controlled node on any of the paths records the identity of its immediate predecessor, the owner of the pseudonym is likely to be recorded more often than any other node. The probability of false positives decreases as the network grows.

Cryptographic pseudonyms can also be used to prove that a series of documents come from the same source. It is sometimes suggested that this will allow readers to build up trust in a pseudonym over time, despite having no knowledge of its owner. However, like other uses of public keys without prior authentication, pseudonymous publishing is vulnerable to a man-in-the-middle attack: the attacker can republish the same documents under a different pseudonym, and wait for readers to build up trust in that pseudonym. Then the attacker can start publishing false documents under the trusted pseudonym. Even if the readers notice that the same documents are being published under two pseudonyms, they cannot know which one to trust. Clearly the readers are no better off then they would be without cryptographic pseudonyms, and they may be worse off if they trust false documents because they were published under a reputable pseudonym.

Dingledine and Mathewson [73] discuss the relationship between usability, configurability and security in anonymising networks. The anonymity set should preferably be large and hard to partition, which means that ease of use and lack of configurability are desirable.

A.1.1 Measuring Anonymity

Several ways have been proposed to measure the degree of anonymity provided by a communication system. In all of them, anonymity is measured from the perspective of a particular observer, and the same message (or set of messages) may have different degrees of anonymity for different observers.

Pfitzmann and Köhntopp [202] define anonymity as ”the state of not being identifiable within a set of subjects, the anonymity set”. Berthold *et al.* [24] measure the degree of anonymity in bits, as the base 2 logarithm of the size of the anonymity set. This implicitly assigns equal probability to all members of the set.

Reiter and Rubin [211] consider the degree of anonymity as a continuum between absolute privacy and provable exposure. Two interesting points along this continuum are probable innocence, where the observer believes that the actual sender of a message is less likely to be the sender than not, and possible innocence, where the observer believes that there is a non-trivial probability that someone else is the sender. Díaz *et al.* [70] point out that for a system with a large number of possible senders, the largest probability assigned to any user may be smaller than 1/2, yet much larger than the probability assigned to any other user. Thus the absolute probability assigned to the sender may not be sufficient to describe the sender’s degree of exposure. Instead, Díaz *et al.* measure anonymity using the entropy of the probability distribution the observer assigns to the set of possible senders. Serjantov and Danezis [232] use a similar definition, but their measure is normalised according to the maximum possible degree of anonymity the system could provide, giving a value between 0 (the sender is assigned a probability of 1) and 1 (all users are assigned equal probabilities).

A.1.2 Message-Based Mix Networks

An anonymous remailer is a server that delivers email while concealing the sender’s identity. The simplest kind of remailer just removes the sender’s address and other identifying headers from the message before delivering it. To

prevent an eavesdropper from linking the incoming and outgoing messages, the incoming message (including the recipient's address) may be encrypted with the remailer's public key. However, the remailer's operator still knows the identities of the sender and the recipient, and may be pressured to reveal them [11]. To reduce the risk, a message can be relayed through a chain of remailers using *onion encryption*: the message and the recipient's address are encrypted with the public key of the last remailer in the chain, the resulting message and the address of the last remailer are encrypted with the public key of the second-to-last remailer, and so on. Each remailer removes a layer of encryption and delivers the revealed message to the revealed address. Only the first remailer knows the sender's identity, and only the last remailer knows the recipient's identity. The disadvantages of onion encryption are that the route must be chosen in advance, and the sender must know the public key of each remailer, which creates a key distribution problem.

As well as supporting onion encryption, some cypherpunk remailers [59] allow messages to be delayed for a random or user-specified time to make it more difficult for an eavesdropper to link inbound and outbound messages by timing. However, if an attacker is able to prevent new messages from reaching a remailer, existing messages can be 'starved out'. A message's route through the remailer network can be traced using a replay attack, in which the attacker replays the encrypted message at each remailer and looks for duplicate outbound messages. Messages can also be identified by their size. Cypherpunk remailers can remove a user-specified amount of padding to alter the size of the outbound message, but messages never get larger – an eavesdropper who sees the unpadded message being delivered knows the minimum size of the padded message, and may be able to rule out certain senders.

Cottrell discusses the weaknesses of cypherpunk or Type I remailers and presents Mixmaster [56, 158], a Type II remailer based on Chaum's mix network design [44]. To prevent an eavesdropper from recognising a message by its size, messages are broken into fixed-size blocks. Each remailer or *mix* decrypts incoming blocks with its public key, reorders them, and outputs them in batches so that blocks cannot be starved out or linked by timing. Duplicate blocks are recognised by their hashes and discarded to prevent replay attacks.

An attacker who controls the first mix in the chain can 'tag' an incoming message by flipping certain bits. When a message emerges from the mix network with the corresponding bits apparently flipped, the sender and recipient can be linked. Mix networks must therefore use message authentication codes at every hop as well as encryption.

If a mix simply waits for enough messages to arrive before outputting a batch, an attacker can trace a single interesting message through the mix by flooding the mix with messages so that it outputs batches just before and just after the interesting message arrives [210]. The interesting message is likely to be the only unknown message in the second batch. To combat this flushing attack, a Mixmaster remailer keeps a pool of messages and only outputs a certain fraction of them in each batch.

A Stop-and-Go mix [144] delays a message for a period chosen by the sender from an exponential distribution. The sender can thus predict when his or her message will leave the mix, but to an eavesdropper the departure times appear random. Flushing attacks are prevented because the departure time does not depend on traffic levels. The expected time of arrival at each mix is encoded in the message, allowing mixes to detect starvation attacks that work by delaying messages, but not those that work by dropping messages altogether.

In Flash mixing [134], batches of messages are reordered and reencrypted by a group of highly-synchronised mixes. Mixes that do not perform the correct reencryptions are detected by inserting dummies in the list.

Díaz and Serjantov [69] analyse existing batching strategies and propose a new strategy in which each message is flushed with an independent probability that depends on the number of messages in the mix.

Chaum [44] also proposed *reply blocks* to allow the recipient of an anonymous message to reply through the mix network by attaching a message to an onion-encrypted header created by the sender. The route through the mix network is not revealed to the recipient. However, reply blocks create the possibility of a replay-like attack in which multiple distinct messages are sent with the same reply block to trace the path back to the sender.

Mixminion [64] is a Type III remailer that provides single-use reply blocks: a mix will not process two blocks with the same header even if they have different bodies. Mixes can only store the hashes of previously seen blocks for a limited time, so to prevent long-term replay attacks a Mixminion remailer changes its public key before discarding the hashes of previously seen blocks. Key rotation also provides forward secrecy: if a mix is compromised, its current key cannot be used to decrypt old messages.

Whereas Mixmaster and older remailers use SMTP [206] to forward messages between mixes, Mixminion uses encrypted, authenticated TLS connections [71]. The link keys are rotated periodically to provide forward secrecy. Encrypted links make replay attacks more difficult, because the attacker cannot recognise identical output messages.

A nym server [64] is an email server that provides mailboxes to anonymous users, who register their accounts and collect their email using a mix network. Each mailbox is associated with a reusable reply block or with a public key that can be used for authenticating single-use reply blocks. Kinateder *et al.* [145] suggest using a distributed hash table to store reply blocks.

The rewebber network [105] uses onion-encrypted URLs to retrieve onion-encrypted files through a chain of HTTP proxies.

Serjantov [231] suggests using a mix network to access a distributed hash table.

As an alternative to mix networks, Berthold *et al.* [24] propose 'mix cascades' in which every message follows the same route. This hinders intersection attacks and makes it possible to provide cover traffic between mixes. Unlike a mix network, a mix cascade remains secure when an attacker controls all but one of the mixes. However, users cannot choose to avoid mixes they do not trust, and blocking access to a single entry point may be easier than blocking access to every mix.

A.1.3 Circuit-Based Mix Networks

To reduce the overhead of exchanging several messages between the same endpoints, some mix networks allow the creation of virtual circuits or *tunnels*: the initiator uses asymmetric cryptography to exchange symmetric keys with each mix in the tunnel, and the symmetric keys are used to onion-encrypt subsequent messages. Circuit-based designs are preferred for low-latency communication because of the high computational cost of asymmetric cryptography.

Onion Routing [106] is a low-latency mix network for TCP communication between anonymous clients and non-anonymous servers. Tunnels are constructed telescopically – only the first mix in the tunnel is contacted directly. Packets are encrypted with the session key of each mix in turn, starting with the last. Each layer of encryption also contains a connection identifier. As the packet passes through the tunnel, each mix removes a layer of encryption and forwards the packet according to the revealed connection identifier. The last mix in the tunnel assigns a unique TCP source port to the connection and delivers the unencrypted packet to the server. Reply packets from the server are encrypted by each mix using its session key and forwarded back along the tunnel.

Tor [74] is the successor to Onion Routing. Tor's tunnels are 'leaky pipes' – packets can exit the mix network at any point along the tunnel, making traffic analysis more difficult. Dummy messages can also be sent partway along the tunnel. Integrity checking at the exit point prevents tagged packets from being sent in the clear – the tunnel is torn down if the exit point detects a corrupt packet. Unfortunately this reveals nearly as much information as transmitting the tagged packet: an attacker could tag a packet and then look for connections being dropped shortly afterwards.

The list of mixes is published on a small number of directory servers (currently three). Mixes accept a list if it is signed by a majority of the directory servers, so an attacker only needs to compromise two servers to bring down the network. Every client must contact the directory servers regularly to update its list of mixes, so an attacker can get an up-to-date list of clients by eavesdropping on one of the directory servers.

Tor provides rendezvous points that allow anonymous users to run servers. If Bob wishes to receive connections from Alice without either of them discovering the other's identity, he creates anonymous tunnels to one or more introduction points, which are similar to the anonymous mailboxes offered by nym servers. He then publishes the locations of his introduction points along with his public key. To request a connection, Alice creates a tunnel to a rendezvous point, encrypts its location and an authentication cookie with Bob's public key, and sends them to one of Bob's introduction points. If Bob wishes to accept the connection, he creates a tunnel to Alice's rendezvous point and passes it the cookie from Alice's message. The rendezvous point checks the cookie and connects the tunnels, forming a two-way anonymous connection.

The rendezvous protocol uses four tunnels: one between each participant and the introduction point, and one between each participant and the rendezvous point. This prevents denial of service attacks against Bob's server, because he can selectively ignore requests.

Scarlata *et al.* [230] present a general scheme for turning any initiator-anonymous protocol into a responder-anonymous protocol; the two protocols can then be composed to provide mutual anonymity.

Back *et al.* [19] describe how routes through a low-latency mix network can be identified if the attacker can determine the latency of the links, especially if the latency can be manipulated, for example by creating congestion. Murdoch and Danezis [186] demonstrate an attack of this kind against Tor.

Partridge *et al.* [197] apply signal processing techniques to traffic data from an encrypted wireless network. It is possible to determine how data is routed through the network without decrypting any packets. Adding random

delays does not effectively prevent traffic analysis: the delays appear as noise and can be filtered out using standard techniques.

Zhang and Paxson [275] use timing analysis to detect 'stepping stones', where a user connects to one host and then connects from there to a second host. Their algorithm looks for pairs of connections with correlated idle periods. The same technique could be applied to onion-encrypted connections, allowing mixes to discover whether they are part of the same tunnel.

Pipenet [61] is a proposal for a mix network is which traffic through each circuit is padded to a constant rate to conceal the true traffic patterns. To prevent an attacker from making one circuit stand out by delaying messages, the entire network stalls when a message is delayed. Unfortunately this creates a trivial denial of service attack. An alternative would be to allow the node that is waiting for the delayed message to transmit a dummy message in its place. However, for the dummy message to be indistinguishable from a genuine message there would have to be integrity checks between the initiator and the exit point instead of between the initiator and every mix, which would allow an attacker-controlled entry point to signal to an attacker-controlled exit point by corrupting packets [62]. This covert channel could be used to link the initiator to the responder.

A.1.4 Peer-to-Peer Mix Networks

In the mix networks described so far, anonymity is provided to a large number of clients by a small number of well-known servers. This approach has certain advantages: the servers can be maintained by experts; faulty or untrustworthy servers can be excluded; and the servers are easy for clients to find. On the other hand, the servers are obvious targets for attack, clients may not be able to agree on a list of trustworthy servers, and access to the servers can be monitored or blocked.

Senders and recipients only communicate with the servers when sending and receiving messages, respectively. (The same applies to initiators and responders in circuit-based networks.) Thus connectivity patterns give some indication of communication patterns, and provide a starting point for replay and intersection attacks.

Tarzan [97, 96] is a peer-to-peer circuit-based mix network that aims to avoid these weaknesses by providing decentralised mix discovery and requiring all users to run mixes, making it hard for an eavesdropper to distinguish initiators from relays.

Each mix communicates with a small number of other mixes called its *mimics*, which are selected verifiably at random. The constrained topology makes it possible to use cover traffic between mimics to frustrate traffic analysis.

A tunnel is constructed by randomly choosing the next mix from among the current mix's mimics. To avoid building tunnels that are entirely controlled by a single entity, IP addresses are hashed in such a way that a mix and its mimics tend to have distinct network prefixes. The assumption is that an attacker may control any number of machines but is likely to have access to a limited number of unique network prefixes. However, an attacker might violate this assumption by using a botnet [121], or by paying private individuals to run mixes.

Mixes are discovered by gossiping. Every mix must know about every other mix with high probability, and must contact every mix to validate its address and public key before propagating them. This could make it very expensive to maintain the network in the face of churn. It might also be unacceptable to users who do not want their use of the system to be widely known. Users who are behind firewalls cannot use Tarzan, because they cannot be contacted and verified by other mixes.

The advantage of gossip is that a mix cannot partition the anonymity set by giving different public keys to different mixes. However, a mix can behave differently when different mixes attempt to validate it. For example, a corrupt mix might respond to validation requests from some mixes but not others, creating an inconsistent view of the network that could partition the anonymity set and interfere with mimic selection and tunnel construction.

Every Tarzan tunnel ends at a network address translator that can establish outgoing connections to internet servers and listen for incoming connections from internet clients, allowing anonymous users to run servers. Since it works at the IP layer, Tarzan can provide anonymity to any application that uses IP, including DNS lookups, web browsing, email, and even other peer-to-peer applications.

MorphMix [212, 213] is a similar system that attempts to prevent a single entity from controlling the entire tunnel by employing randomly chosen 'witnesses' during key exchange and by compiling statistics on how often mixes suggest one another for inclusion in tunnels. Good mixes must change their neighbours frequently to make colluding mixes stand out.

Unlike Tarzan it is not necessary for every mix to know about every other mix, but addresses must be learned from several sources to avoid being surrounded by colluding mixes. The suggested sources include a local cache of previously active mixes, and information servers that return the addresses of currently active mixes. A corrupt information server could partition the anonymity set by giving different information to different requesters, for example by giving each requester a different port number for the same corrupt mix. Whenever the mix was employed as a witness, the attacker would be able to identify the initiator by the port number of the incoming request.

I2P [128] provides mutual anonymity in a similar way to Tor: each node constructs separate inbound and outbound tunnels, and publishes the address of its inbound tunnel gateway in a distributed hash table [176] under a public key pseudonym. The distributed hash table also stores membership information, removing the need for directory servers.

Like Tor, I2P's low-latency tunnels may be vulnerable to timing analysis, which could be used to gather samples for a predecessor attack.

In Cebolla [32], lists of mixes are retrieved from one or more sources and checked for consistency – initiators might be identified by their choice of mixes. Encryption is optional, which provides a small performance advantage to initiators with low anonymity requirements at the cost of reducing the size of the anonymity set for encrypted traffic.

A.1.5 Other Peer-to-Peer Anonymising Networks

Crowds [211] provides client anonymity for web browsing. Clients form a peer-to-peer 'crowd' coordinated by a central membership server. Each client constructs a path through the crowd by sending a request to a randomly chosen peer. The peer decides randomly whether to forward the request to another member of the crowd, or to act as a web proxy, communicating with servers on the client's behalf.

A client sends all communication along the same path until it receives a synchronisation message from the membership server, which tells all clients to construct new paths. Synchronisation has two purposes. First, it prevents new crowd members from being linked to new paths, because a client does not construct its first path until it receives its first synchronisation message, at which point all other clients are constructing paths. Second, it prevents clients from constructing paths too frequently, which would expose them to predecessor attacks.

A typical web client produces bursts of related requests, eg for a page and the images contained within it. It might be possible for a peer to guess the number of peers between itself and the client by comparing the timing of requests with the page structure [119]. To prevent this kind of traffic analysis, the web proxy parses the page and requests all the embedded items from the server, returning them along with the page. Crowds therefore cannot handle encrypted HTTPS connections or dynamic content such as JavaScript.

Connections between peers are encrypted using keys obtained from the membership server, but there is no end-to-end encryption, and no reordering or padding.

Hordes [163] is a similar proposal that uses IP multicast to return information to the client. The web server must run a Hordes-aware proxy to handle the asymmetric communication. Unlike Crowds, each packet is routed independently, so Hordes is vulnerable to the predecessor attack.

The Invisible IRC Project [131] is an initiator-anonymous version of Internet Relay Chat [192]. Clients connect to a standard IRC server through a network of anonymising relays.

Freenet [53, 51, 50] is a peer-to-peer file-sharing network that provides anonymity for publishers and readers. Unlike most file-sharing networks, files are not stored by their publishers. Instead each file is inserted into the network under a key, and forwarded towards the node which is judged to be the 'epicentre' for that key. Requests for the key are routed in the same way, and if the file is found it is forwarded back along the path to the requester. Nodes cache copies of the files they forward, so files that are frequently requested or inserted will tend to be widely replicated, and requests for popular files will tend to be satisfied in a small number of hops.

When a node forwards an inserted file it stores a routing hint associating the file's key with the address of the next node. A node returning a requested file creates a routing hint associating the file's key with the address of the epicentre, which is given in the reply message. This allows nodes to learn which nodes are good sources for which areas of the key space, which in turn results in each node receiving requests for keys that are close to those it has supplied in the past, causing its routing and data caches to specialise in certain areas of the key space.

Freenet provides anonymity for publishers and readers because a node cannot tell whether a request originated at the previous node or was forwarded on behalf of another node. All data travelling between nodes is encrypted, as are the data and routing caches when they are stored on disk. However, traffic between nodes is not padded or reordered, so Freenet may be vulnerable to passive traffic analysis.

Unlike circuit-based mix networks, the publisher and reader do not have to be active at the same time, which makes it more difficult to perform intersection attacks.

If every node knew the epicentre for every key then messages would only travel one hop, destroying anonymity. In general, the shorter the path, the more likely it is that the previous node is the originator of the message. Specialisation undermines anonymity in a similar way: if the network reaches a state in which every node specialises in a few areas of the key space, then a request for a key outside the previous node's specialisation probably originated at the previous node, since no other node would be likely to send it such a request. To ensure that paths do not become too short or nodes too specialised, occasionally a node that is returning a requested file will replace the address of the epicentre with its own address, hiding the true source of the file.

No single node is responsible for storing a published file, so there is no way to ensure that a file inserted into Freenet will be retrievable in the future. Any node can store a permanent copy of any file, but requests for that file will not necessarily reach that node, as it is unlikely to be the 'natural' epicentre for that key. The only way to ensure a file's longevity is popularity, since frequently requested files will tend to be widely replicated. In practice it seems to be necessary to reinsert or request files periodically to prevent them from being pushed out of Freenet by more popular (or more frequently polled) content.

Dhamija [68] describes several attacks against routing, anonymity and node discovery in Freenet.

Zhang *et al.* [274] propose that each Freenet node should actively specialise in files that fall into a particular region of the key space. This decreases the number of routing hints that each node must store, and may help the network to stabilise on a single epicentre for each key. Unfortunately it also makes it possible for a corrupt node to nominate itself as the epicentre for a key that it wishes to censor. By responding normally to all requests except those for the censored file, the node may gradually become known throughout the network as a reliable source for that area of the key space, and may be able to attract (and drop) requests for the censored file. The proposed solution to this problem, in which a node's neighbours collectively choose which region it will specialise in, is flawed because multiple nodes can be controlled by a single attacker [78]. Goel *et al.* [104] describe a protocol based on partial hash collisions that could be used to select a node's specialisation more securely.

FASD [153] is a distributed, anonymous search engine built on top of Freenet.

GNUnet [102, 23] is an anonymous file-sharing network in which files are broken into fixed-size blocks and each block is stored under its own key. Since this creates a large index relative to the amount of data, a Bloom filter [27] is used to reduce the size of the in-memory index. Queries use UDP rather than establishing a TCP connection and exchanging cryptographic keys, so brief exchanges between peers are more practical than in Freenet, but there is no link encryption, so a global eavesdropper can determine the source of any request. Unlike Freenet, files can be stored permanently on a single node rather than (or as well as) being inserted into the network. Queries are routed randomly – there is no concept of an 'epicentre' for a key.

Kugler [154] describes a predecessor attack against reader anonymity in GNUnet. If a file is held permanently on a single node rather than being inserted into the network, a similar attack can be used to find the publisher.

In the MUTE file-sharing network [187], file transfers as well as searches pass through an encrypted multi-hop overlay. MUTE's routing protocol is inspired by the collective foraging behaviour of ants, which use pheremone trails to guide one another to food sources. Each node counts the packets arriving from each source on each link, and packets are forwarded probabilistically based on the strength of the destination's 'scent'. Packets addressed to unknown destinations are flooded to discover short routes.

Packets are not signed, so an attacker could divert a victim's packets by adopting the victim's address. Even if packets were signed, reverse-path routing assumes that a good route *from* a node is also a good route *to* the node. If an attacker dropped packets addressed to a victim but continued to forward packets generated by the victim, neighbouring nodes would not attempt to route around the attacker.

AntsP2P [12] adds end-to-end encryption to MUTE, but intermediate nodes can carry out a man-in-the-middle attack unless one of the endpoints obtains the other's public key out-of-band. AntsP2P uses a two-tier structure to improve scalability.

One of the weaknesses of peer-to-peer anonymising networks is that users can learn the network addresses of a large number of other users. This kind of 'address harvesting' could provide information for intersection attacks or traffic blocking, and in some circumstances mere participation in an anonymous communication network might be regarded as suspicious.

Hazel and Wiley [118] describe Achord, a modified version of Chord [246] in which each node only reveals its participation to $O(logN)$ other nodes. An attacker cannot discover the address of the node that is responsible for a given key unless it is one of the node's $O(logN)$ neighbours, which can be checked by hashing its IP address. However, this does not provide much protection against an attacker with access to a large number of IP addresses. If we take each node's number of neighbours to be $log_2 n$, the probability that an attacker with access to a IP addresses can become the neighbour of a given node is $1 - ((n - log_2 n)/n)^a$. For example, in a network of 10,000 nodes, an attacker who controls a single node and 256 IP addresses can expect to harvest the addresses of 29% of the participants. If some of the participants use dynamic IP addresses then the attacker can harvest additional addresses over time.

It might seem that Chord's routing protocol does not provide strong anonymity for initiators, because the clockwise distance to the requested key decreases at each hop, allowing a node to estimate the probability that the previous node is the initiator. However, O'Donnell and Vaikuntanathan [191] show that because the previous hop could be forwarding the request for a large number of nodes, the size of the anonymity set grows linearly with the number of nodes.

A.1.6 Anonymous Broadcast Networks

The dining cryptographers protocol [45] provides provable sender and recipient anonymity within a broadcast group. Each pair of participants shares a secret key that is used to seed a pseudo-random number generator. In each round of the protocol, each participant XORs the outputs of her pseudo-random number generators. A participant who does not wish to send a message transmits the result, while a participant who wishes to send a message transmits the XOR of the message and the result. Each participant XORs the values she receives along with the value she transmitted. If no message was sent, the result is all zeroes, because each pseudo-random bit was transmitted twice. If one message was sent, it is received by all participants but none of them can identify the sender. If more than one message was sent, the result is unreadable.

Sender and recipient anonymity are guaranteed, but the protocol scales poorly and is vulnerable to denial of service attacks: any participant can anonymously jam the channel by transmitting in every round.

In XOR trees [77] a variant of the dining cryptographers protocol is used to communicate over a spanning tree.

Herbivore [104] is a peer-to-peer network in which cliques of around 100 nodes implement the dining cryptographers protocol. A structured overlay [246] routes messages between cliques. An eavesdropper can observe communication between cliques, but cannot tell which member of each clique is communicating. Anonymous connections can also be made to internet servers. (CliqueNet [239] is an earlier proposal by the same authors.)

Herbivore's secure entry protocol uses computational puzzles to prevent nodes from joining arbitrary cliques, but the average clique contains 128 nodes, so an attacker who controls a small fraction of the nodes has a reasonable chance of having a node in any given clique, allowing the attacker to jam the clique's communication anonymously. For example, in a network of 100 cliques, an attacker who controls just 1% of the nodes can jam an expected $1 - (99/100)^{128} = 72\%$ of the cliques.

Nodes leave a clique if they are unable to transmit for a certain number of rounds, but this exposes them to an intersection attack: if an attacker observes a clique communicating with an internet host the attacker wishes to censor, the attacker can jam the clique (assuming it controls a suitable node) until the transmitting node leaves. The attack is then repeated against any clique that attempts to communicate with the censored host, each time intersecting the membership sets of the cliques until the transmitter is identified.

P^5 [235] uses broadcast groups to hide the source of messages, but does not rely on the dining cryptographers protocol. Senders join broadcast groups that are arranged in a binary tree, with each group having a variable-length identifier that indicates its position in the tree: the root group has an empty identifier, its two children have the one-bit identifiers 0 and 1, and so on. Each node belongs to several randomly chosen groups, keeping the tree connected with high probability.

Messages are addressed to prefixes, and are received probabilistically by every group with a matching identifier. For example, a message addressed to the empty prefix will reach the entire tree, while a message addressed to the prefix 0 will reach the left half of the tree. Messages are delivered unreliably, and the loss rate is generally higher for shorter prefixes, because of the larger number of matching groups. This allows each user to trade off anonymity against efficiency: by revealing a longer prefix to a sender, a recipient reduces the size of his or her anonymity set, but also reduces the number of groups that will receive the sender's messages, improving the signal to noise ratio. Each correspondent chooses a suitable degree of anonymity, and communication takes place using the longest common prefix.

P^5 also supports onion encryption, allowing messages to be forwarded through multiple groups to hide the sender's identity. Router keys are distributed by broadcast, which might allow a man-in-the-middle attack.

A.2 Robustness

Censorship resistance can be seen as an extreme example of the well-studied problem of availability: that is, ensuring that information remains available in the face of intentional attacks and accidental failures. Availability requires both robust storage of the information, and robust communication between those who are storing the information and those who wish to access it.

A.2.1 Robust Storage

Robust storage systems make it difficult to delete documents that have been published, without necessarily providing anonymity for publishers or readers.

Usenet [123] is a public discussion forum divided into topics or newsgroups. Anyone with access to a Usenet server can read and publish messages, although messages sent to some newsgroups must be approved by moderators, and most users cannot create new groups.

The NNTP protocol [141] that is used for most Usenet connections does not distinguish between clients and servers – messages are published, propagated, and read using a simple peer-to-peer flooding mechanism. When a server receives a new message, it sends the message's identifier to all neighbours (clients and servers) that have expressed an interest in the newsgroup to which the message is addressed. Neighbours that have not already seen the message request it and propagate it to their own neighbours.

Usenet allows publishers and readers to remain anonymous to everyone except the local news server. Likewise, servers can be concealed from everyone except their neighbours. NNTP connections can be encrypted using SSL, although this does not prevent traffic analysis. Message identifiers are not cryptographically secure, so a message can be prevented from propagating by sending out a different message with the same identifier. It is hard for users to remove messages from the system once they have been published, but a node at an articulation point in the network can choose which messages pass from one subnetwork to the other – for example a server can filter what is available to local clients. Servers generally store the most recent messages in each group and discard old messages to save space, so an attacker may be able to force a message to expire by sending a large volume of messages to the same group.

Anderson [10] describes the Eternity Service, a robust network of servers that store files in exchange for anonymous payment in a digital currency. Although it has never been implemented, the Eternity Service has influenced many later designs.

Publius [259, 257] is a censorship-resistant publishing system in which a small number of well-known servers store encrypted documents on behalf of clients. To publish a file, a client encrypts it with a secret key and then breaks the key into n shares, any $k < n$ of which can be used to reconstruct the key. Each share is stored on a different server along with a copy of the encrypted file. No single server can decrypt the file it is holding, but a client that retrieves enough shares (using a multi-part URL which is assumed to be unknown to the servers) can reconstruct the key and thereby decrypt and tamper-check the file.

There is a password-based update mechanism that allows a file's publisher to replace the file with a pointer to a newer version. This mechanism would be susceptible to a dictionary attack by a server operator, or a slower (but parallelisable) dictionary attack by a client.

Publius uses standard web server and client software plus a local proxy on the client side, so it is easy to deploy. However, the list of servers must be well-known and agreed by all clients, so central administration is required, and the servers may be attacked or access to them may be blocked. Files are published and retrieved using a standard HTTP connection – the client's identity is not hidden from the server or from eavesdroppers, although clients requiring anonymity could connect through a mix network.

Tangler [258] is a similar system that does not require central administration. Servers can join and leave dynamically, but there must be general agreement about the list of participating servers. Server and data identifiers are mapped onto a circular address space using a hash function. Each server is responsible for several regions of the address space selected by hashing its public key and the date. Each region changes hands on average every 14 days. A server can make itself responsible for an arbitrary region if it can find a suitable public key. (Goel *et al.* [104] describe a protocol that mitigates this problem using partial hash collisions.)

A *storage credit* is a cryptographically signed promise to store a block of data for a period of two weeks. The amount of storage a server can request depends on the amount it makes available. If a server that has stored a block fails to return it, the requesting server removes the failed server from its list and presents the signed storage credit to another server (a witness), which requests the same block from the failed server. This process continues until the block is retrieved or the failed server is evicted from the network. The server can rejoin the network immediately under a different key, but new servers must supply free storage for one month before requesting storage credits, limiting the amount of damage they can inflict.

Documents are published in hierarchical collections and the root of each collection is signed with the publisher's private key, allowing publishers to update their collections and readers to check that documents have not been tampered with. A collection can contain signature-based 'soft links' and hash-based 'hard links' to files in other collections.

Tangler's most interesting feature is *document entanglement*. This is a way of combining a new file with previously published files in such a way that reconstructing the new file requires access to the files with which it was entangled. This makes it difficult to delete certain files from the system without affecting others, although a file is not protected until newer files have been entangled with it. Entanglement creates an incentive for publishers to replicate one another's files, but it does not create an incentive to entangle one's files in the first place (entanglement doubles the number of credits needed to store a file).

To entangle a file, it is first broken up into data blocks. For each data block, two previously published server blocks are requested at random from the Tangler network. A variation of Shamir's secret sharing algorithm [234] is used to transform the three blocks into four shares, two of which are the previously published server blocks and two of which are new server blocks. Any three of the four shares are sufficient to reconstruct the data block. The new server blocks are stored in the Tangler network and may be further entangled with future documents.

For each new server block that must be stored, the publisher requests a storage credit from any Tangler server. Servers can apply any policy they like to storage requests – they might require payment or registration, might allow anonymous access through a mix network, etc. Requests are blinded so that the server does not see the block it is requesting storage for, but knows which server is responsible for the corresponding region. The request is forwarded to the responsible server, and the resulting storage credit is returned to the publisher, who uses it to store the server block on the responsible server. Servers exchange storage credits at the same time each day, which hinders traffic analysis but could create load and synchronisation problems.

PAST [79] is a distributed storage system built on top of the Pastry overlay network [80]. Files are replicated across multiple nodes to keep them available in the face of attacks and failures; requests are routed to the nearest replica according to a metric such as geographic distance or network latency. To prevent users from using more storage than they contribute, PAST has a quota system based on centrally administered smart cards. This provides a single point of attack for anyone wishing to shut down the network.

FARSITE [5] also uses a hierarchical public key infrastructure to control access to a decentralised, distributed file system. A Byzantine agreement protocol [40] is used to prevent corrupt nodes from tampering with directory metadata.

Like Tangler, CFS [60] allows users to publish hierarchical collections of files under digitally signed root blocks. Each block of a collection is stored on a server selected by hashing the block; the Chord protocol [246] is then used to locate the appropriate server. Files are stored for a finite time unless the publisher renews them, subject to a storage quota for each IP address – this could create problems for clients who wish to store files anonymously through a mix network.

HiveCache [120] is a distributed backup system that uses content hashes to identify duplicated files on a local network, and distributes copies of unique files to machines with free storage space. Mnet [182] is a wide-area distributed file store based on similar principles.

Song *et al.* [243] describe methods for searching an encrypted file without revealing the search terms or the contents of the file to the server performing the search. This could be used to combine keyword searches with 'plausible deniability', in which servers should not understand the content of files they are storing.

A.2.2 Robust Routing

Byzantine Robustness

Perlman [200] considers the problem of routing in the presence of faults that produce arbitrarily complex behaviour, which are known as Byzantine faults [157]. A network layer is considered *Byzantine robust* if it can deliver a packet

provided there is a fault-free path between the source and the destination. This means that faulty components off the path must not be able to interfere with the discovery or use of the fault-free path.

Perlman's solution uses reserved buffers, round-robin scheduling, non-wrapping sequence numbers, and public key authentication. A buffer is reserved at each router for each source, and a packet is only admitted to the buffer if it is signed with the source's private key and has a higher sequence number than any packet previously seen from that source. Packets are flooded through the network, so if a fault-free path exists between a source and a destination then it will be found. Round-robin scheduling on each link ensures that every source will have access to the link within a predictable time; non-wrapping sequence numbers prevent faulty nodes from causing interference by replaying authenticated packets; reserved buffers ensure that no interference occurs between authorised sources; and public key authentication excludes unauthorised sources from the network.

If a router receiving an authenticated packet already has in its buffer an authenticated packet from the same source with a higher sequence number, it replies with the buffered packet and does not buffer the received packet. This ensures that every router eventually obtains a copy of the latest packet from every source, even after crashes or partitions. It also provides a way for a source to reacquire its own sequence number after a crash.

A source is responsible for ensuring that it does not generate a new packet until its previous packet has been received by the destination. Otherwise the new packet may overtake the old packet and prevent it from reaching the destination. To allow sources to transmit data more quickly, the design can be extended to multiple buffers per source at each router.

The robustness of this scheme depends on authentication: to allocate resources correctly, each router must have a complete list of sources along with their public keys. To reduce the amount of manual configuration required, Perlman suggests delegating the distribution of public keys to one or more *trusted nodes*. The trusted nodes' public keys are manually installed at each router, and each trusted node uses the robust flooding described above to distribute signed public key lists containing the identities and public keys of authorised sources.

In the event that one or more of the trusted nodes develops a Byzantine fault, the public key lists received from different trusted nodes may disagree. (This is possible even in the absence of faults, if updates to the public key list are received from different trusted nodes at different times.) The definition of Byzantine robustness used in this situation is more restrictive: a node is only considered non-faulty if there is a non-faulty path from at least one non-faulty trusted node to the node in question. (In other words, a node that does not have at least one correct public key list is considered faulty.)

Two mechanisms limit the damage that can be caused by faulty trusted nodes. First, the number of sources that can be listed by any trusted node is bounded by the manually configured parameter N. If the number of trusted node keys installed on a router is t, then it must reserve a maximum of tN buffers. If a trusted node lists more than N sources then it is assumed to be faulty.

Second, every source checks that its own identity and public key are in the list. If they are missing then the trusted node that signed the list is assumed to be faulty and its list is ignored.

Zhou and Haas [277] describe a different technique for distributed management of public key lists. Whereas Perlman uses a different certifying key pair for each trusted node, Zhou and Haas use a single key pair distributed across multiple nodes. To prevent a single faulty node from creating false certificates, the private key is broken into shares using a secret sharing algorithm [234]. The shares can only be used to generate partial signatures, several of which must be combined to produce a valid certificate. The scheme is secure against attackers who can compromise new nodes over time, referred to as 'mobile attackers', because the certifying nodes periodically recombine their shares of the private key to produce a new sharing of the same key, such that partial signatures created with old shares cannot be combined with partial signatures created with new shares. It is also impossible to reconstruct the private key from a mixture of old and new shares. Re-sharing the key in this way effectively revokes the old shares without changing the key. To reconstruct the private key, the attacker must compromise all the necessary nodes between one re-sharing and the next. However, the paper does not explain how compromised nodes can be detected and excluded from the re-sharing process.

Perlman's flooding scheme is robust but inefficient, because every packet must be transmitted and authenticated on every link. An alternative scheme by the same author, robust link state routing [200], provides higher efficiency but lower robustness. Every node uses the robust flooding scheme to broadcast a signed link state packet listing its neighbours. As with the robust flooding scheme, the network may contain several trusted nodes and their public key lists may not agree, so each source constructs a separate 'view' of the network for each public key list. A link between two nodes is added to a view if both nodes report the link and both nodes are in the view's public key list.

Each view is then used to calculate node disjoint paths between the source and the destination, and packets are source routed along the paths.

The link state scheme is less robust than the flooding scheme because a fault-free path will be rejected by the path selection algorithm if it shares one or more nodes with a faulty path.

Avramopoulos *et al.* [15, 14] propose using message authentication codes instead of digital signatures to authenticate packets in robust link state routing. This decreases the computational overhead but increases the bandwidth overhead, as a separate message authentication code is needed for each node along the path. Acknowledgements, timeouts and fault announcements are proposed to detect and avoid faulty links.

There are no proposals that achieve Byzantine robustness without trusted nodes. Awerbuch *et al.* [16] describe a way to detect and avoid faulty links using probe messages, but without certified identities it is possible for an attacker to advertise any number of imaginary nodes and links, replacing them with new ones as soon as they are detected [78].

Secure Ad Hoc Routing Protocols

Link State Procotols

Hauser *et al.* [117] propose using one-way hash chains to authenticate link state updates. Separate chains are used to indicate whether a link is active or inactive. Cheung [49] suggests using hash chains and delayed disclosure [201] to generate message authentication codes. This requires loose clock synchronisation and optimistic acceptance of unverified updates.

Nodes in the Secure Link State Protocol (SLSP) [195] maintain topological information for a local zone. Hash chains are used to prevent signed link state updates from propagating out of the zone. Routing between zones is not discussed.

All these protocols require certified identities, and all are vulnerable to a tunnelling attack in which two corrupt nodes construct a tunnel between themselves and pass encapsulated packets through it [225]. This makes the corrupt nodes appear to be neighbours, so the route appears shorter than it actually is, which will cause many routing protocols to prefer it.

These protocols authenticate routing packets, but none of them attempts to ensure the delivery of data packets once a route has been discovered. An attacker might participate correctly in the routing protocol and then drop data packets.

Distance Vector Protocols

Smith *et al.* [240] suggest using digital signatures to authenticate the immutable fields of distance vector updates, and a signed predecessor field together with a path-finding algorithm (in other words a link state protocol) to authenticate the mutable hop count field.

Secure Ad Hoc On Demand Distance Vector routing (SAODV) [271, 272] also uses digital signatures to authenticate immutable fields. Hash chains are used to prevent corrupt nodes from decreasing the hop count, although they cannot be forced to increment it. Secure Efficient Ad Hoc Distance Vector (SEAD) [124] uses hash chains in place of DSDV's destination sequence numbers.

As with the link state protocols, all these protocols require certified identities, are vulnerable to tunnelling, and fail to protect data packets.

Source Routing Protocols

Ariadne [125] uses hash chains and delayed disclosure [201] to authenticate a DSR-like source routing protocol. This requires loose clock synchronisation and certified identities.

Papadimitratos and Haas [194] describe an on-demand source routing protocol in which route requests and replies are authenticated end-to-end using message authentication codes. The source and destination do not need to share keys with intermediate nodes, so no certificate authority is required. However, the protocol is vulnerable to tunnelling and does not protect data packets.

ARAN [225] is an on-demand protocol that finds the quickest (rather than the shortest) path from the source to the destination in order to prevent tunnelling attacks. Unused paths are eventually deleted by intermediate nodes. A certificate authority is required, and data packets are not protected.

The Secure Border Gateway Protocol (S-BGP) [143] uses certified identities to authenticate inter-domain route advertisements and to check that every autonomous system along the route is authorised to advertise the destination.

A.3 Cooperation

Any system in which the infrastructure is provided collectively by the users faces the problem of encouraging users to contribute resources as well as consuming them. In other words, users must cooperate with one another. This problem can also be seen as an aspect of robustness, since many denial of service attacks work by exhausting resources.

Nielson *et al.* [190] classify the various 'rational attacks' that might be carried out by self-interested nodes in a communication network. They distinguish between altruistic or obedient nodes, which obey the system's protocols regardless of their own self-interest; malicious nodes, which attempt to damage the system; and rational nodes, which attempt to benefit from the system, possibly by disobeying its protocols.

Adar and Huberman's widely cited paper *Free Riding on Gnutella* [4] presents measurements from a large Gnutella network, which show that the majority of participants make no files available for download, and the majority of downloads are from a tiny minority of peers. It is suggested that this 'free riding' phenomenon will eventually lead to the collapse of the network, although the large number of downloads observed could equally be seen as evidence of the network's continuing health, albeit at the expense of a small number of altruists. Gnutella-based networks continue to attract millions of users [167], but file-sharing networks with incentive mechanisms appear to be more popular still [196].

Four main approaches have been taken to the problem of encouraging resource contribution in communication networks:

1. **Micropayments:** nodes pay one another for services using a digital currency. To prevent double spending, micropayment systems require either tamper-resistant hardware or a central bank.
2. **Audits:** nodes test one another and exclude misbehaving nodes from the network. Excluded nodes must not be able to re-enter the network under new identities, so audit systems require certified identities or other barriers to entry.
3. **Reputations:** nodes publicise the outcomes of their interactions, and combine other nodes' reports with their own observations when deciding whether to cooperate. Like audits, reputations require limits on the creation of identities.
4. **Reciprocation:** pairs of nodes dynamically adjust the level of service they offer each other based on the level of service received. Reciprocation does not require any centralised infrastructure.

Huang *et al.* [126] question the usefulness of incentive mechanisms in mobile ad hoc networks. They argue that incentives to cooperate are unlikely to be necessary during the early stages of adoption, when most users are likely to be enthusiasts who will not mind incurring costs to use a new technology. Incentive mechanisms may even hinder adoption by increasing complexity and adding unnecessary performance overheads. It is suggested that incentive mechanisms could be grafted on later at the application layer if they turn out to be necessary. However, incentive mechanisms in ad hoc networks are meant to enable multi-hop communication by encouraging nodes to forward packets. It is not clear that application-layer incentives could achieve this goal, since intermediate nodes might not be interested in the same applications as the endpoints. Application-layer incentives would seem to imply the construction of a separate ad hoc network for each application, whereas network effects [166] suggest that a single general-purpose network would be more useful.

A.3.1 Micropayments

Golle *et al.* [107] present a game theoretic analysis of free riding in centralised file-sharing networks. Micropayments are proposed to encourage sharing.

Crowcroft *et al.* [58] establish a model of bandwidth and energy costs for source, sink and transit nodes in a multi-hop wireless ad hoc network. A node incurs costs whenever it receives or transmits a packet, and it is therefore not in a node's interest to forward packets for other nodes. Micropayments are suggested to encourage forwarding, but the paper does not explain the operation of the micropayment system. Xue *et al.* [267], Chandan and Hogendorn [43], and Anderegg and Eidenbenz [9] also develop pricing models for wireless networks.

Buttyán and Hubaux [36] propose a virtual currency called 'nuglets', implemented with tamper-resistant hardware and a public key infrastructure.

Ioannidis *et al.* [133], Vishnumurthy *et al.* [255], Li *et al.* [165], and Figueiredo *et al.* [91] describe micropayment systems for peer-to-peer networks.

A.3.2 Audits

Blanc *et al.* [26] model peer-to-peer routing as a random matching game: in each round the nodes are randomly paired, and each pair plays a single-round prisoner's dilemma [155]. Defection is reported to a reputation system, the operation of which is not explained. A node retrieves the reputation of its opponent before playing, and defection is punished for a certain time period, during which the punished node must always cooperate and tolerate defection, otherwise the period restarts. It is shown that if the punishment time is long enough, an equilibrium exists in which all nodes cooperate. However, the mechanism is vulnerable to noise and assumes regular traffic patterns. If a node can compress its traffic into bursts, it can defect at the end of each burst and then wait for the punishment period to expire before playing again. A user who can create multiple identities can use them in rotation, defecting and then leaving the identity idle until its punishment is over.

Catch [171] is a protocol that encourages forwarding in multi-hop wireless networks by making it risky for nodes to lie about their connectivity to avoid forwarding. Nodes use anonymous challenges to test their neighbours' connectivity; nodes that fail to retransmit challenge messages are excluded from the network. Punishment depends on the assumption that nodes cannot generate new identities, but MAC addresses can be changed freely on most existing hardware, and indeed the Catch protocol relies on the ability to scrub the source address from outgoing packets. The authentication mechanisms in IEEE 802.11i [130] are cited as preventing nodes from changing identities, which means that Catch is only applicable to networks with a central authentication infrastructure.

Ngan *et al.* [189] describe a peer-to-peer storage system in which every node publishes a signed list of the files it is storing on behalf of other nodes, and the files other nodes are storing on its behalf. Random audits are used to detect and exclude nodes that publish false claims.

A.3.3 Reputations

Resnick *et al.* [214] provide an overview of reputation systems on the internet. Three requirements for a successful reputation system are identified:

1. Long-lived entities that inspire an expectation of future interaction.
2. Capture and distribution of feedback about interactions.
3. Use of feedback to guide trust decisions.

The first requirement is perhaps the hardest to guarantee in an open system. A Sybil attack [78] involves the use of multiple identities by a single attacker. Douceur shows that such attacks are possible in almost any system without a central administrative component. Introducing barriers to the maintenance of multiple identities, such as computational puzzles [81], can limit the extent of the attack but not prevent it entirely; the limit cannot be tight if there is significant heterogeneity in node capabilities, because a single strong node can impersonate several weak nodes.

Buchegger and Le Boudec [33] propose a Bayesian reputation system for ad hoc networks. Nodes exchange observations of good or bad behaviour with their neighbours, integrating the rumours with their own observations using a Bayesian rule which attempts to exclude observations that do not fit prior expectations. In this way the effect of slander is minimised. This allows the reputation system to incorporate both positive and negative information without being vulnerable to liars. However, if a node is free to generate new identities then it can discard its negative reputation, or use separate concurrent identities for good and bad behaviour.

A more recent proposal from the same authors [34] uses separate performance and honesty metrics. A node is considered honest if its reports of other nodes' performance are consistent with first-hand observations. If a node does not meet a certain threshold of honesty, its reports are not incorporated. It might be possible to exploit this mechanism by performing well for some nodes and badly for others, causing them to mistrust one another and thus preventing the spread of bad performance reports.

P-Grid [1] is a peer-to-peer network incorporating a reputation system [67]. Performance reports are stored in a structured overlay and can be retrieved by any interested node. Each report is weighted according to the reporter's trustworthiness, which is based on first-hand experience of its performance – performance and honesty are conflated. The weighted reports are then aggregated using maximum likelihood estimation.

It is assumed that liars do not collude, because the reputation system is vulnerable to an attacker who controls two identities: the first identity misbehaves, and the second identity behaves well in order to appear trustworthy, but gives false positive reports about the first identity.

Marti *et al.* [174] propose a two-part solution to selfish behaviour in ad hoc wireless networks. They do not address malicious or collusive behaviour. The first component of their solution is the 'watchdog': nodes listen in promiscuous mode to overhear their neighbours forwarding packets. The second component is the 'pathrater': nodes that forward less than a certain fraction of packets are reported to the source, and routes are selected so as to avoid uncooperative nodes. It might be argued that this is exactly what uncooperative nodes want, since they will save bandwidth and battery power by being excluded from forwarding. However, well-behaved nodes also benefit because they avoid retransmitting lost packets.

A.3.4 Reciprocation

SLIC [249] is an incentive mechanism for peer-to-peer search overlays. Each node allocates capacity to its neighbours' queries based on the number of search results they have supplied in the recent past. The SLIC incentive mechanism is very simple, with a single value between 0 and 1 representing a node's assessment of the quality of each neighbour. The value is updated periodically based on the number of search results returned. Clearly this creates an incentive for nodes to manufacture search results, but a similar mechanism could be applied to verifiable tokens.

Simulations show that nodes which share fewer files than average receive worse than average service from the network, creating an incentive to share. Nodes that share more than the average number of files experience better than average service, although the improvement eventually levels off. Similar relationships hold for the number of connections and the amount of capacity allocated to neighbours' queries. A malicious node cannot flood the network by allocating most or all of its capacity to its own queries, since by doing so it decreases its neighbours' satisfaction with its performance and thus decreases the capacity allocated to it. However, a selfish node can increase the number of results it receives by sending slightly more queries than the average. It is not clear what determines the level at which the number of results starts to decrease again due to decreased cooperation from neighbours, or what would be the effect of all nodes increasing their query rates.

Queries and results travel multiple hops while reciprocation only takes place over a single hop, so the resources used and contributed by a node are not measurable by its immediate neighbours. Another complicating factor is that searches and search results compete for bandwidth – it is not clear whether the capacity used by a search result is 'charged' to the searcher, the responder, or neither.

Further simulations show that if nodes are allowed to drop an unsatisfactory connection and establish a new connection while keeping the total number of connections constant, nodes with fewer than average files or higher than average query rates tend to experience reduced utility, while other nodes tend to experience increased utility – in other words the topology adapts to exclude free riders. SLIC addresses the problem of free riders rejoining the network under new identities by assigning a value to new connections that depends inversely on the node's level of satisfaction with its current connections. A node that is satisfied gives a low value to new connections, while a dissatisfied node is more likely to take a risk on an unknown neighbour in the hope of improving its situation. There seems to be a danger of positive feedback here, with dissatisfied nodes becoming increasingly vulnerable to free riders.

BitTorrent [54] is a peer-to-peer protocol for distributing large files. It aims to shift the burden of distributing a file onto those people who are interested in obtaining it, by allowing them to download pieces from one another. Each peer connects to a central *tracker* to find other peers that are downloading the same file. Peers contact one another and share information about which pieces of the file they have downloaded.

BitTorrent uses a technique inspired by Tit For Tat (see Section A.3.5) to encourage peers to upload. Each peer maintains a fixed number of 'unchoked' uploads, with all other connections 'choked' (nothing is uploaded). Periodically, one randomly chosen connection is unchoked – this roughly corresponds to Tit For Tat's policy of cooperating on the first move. The other unchoked connections are those that have recently provided the best download speed – this corresponds to Tit For Tat's policy of rewarding cooperation and punishing defection. When all downloads appear to be choked at the other end, additional connections are randomly unchoked. This allows peers to recover from cycles of mutual punishment.

BitTorrent achieves excellent download speeds in practice, but the centralised tracker limits scalability and creates a single point of failure. Recent versions of BitTorrent allow peers to discover one another using a distributed hash table [176, 18].

The eDonkey2000 [83] and eMule [85, 156] file-sharing networks also allocate bandwidth in a reciprocal fashion. The criteria used by eDonkey2000 are not documented, but eMule uses a combination of the amount of data sent, the amount of data received, the priority assigned to the requested file by the uploader, and the length of time the requester has spent in the queue [156].

Cooper and Garcia-Molina [55] describe a reciprocal backup system in which peers bid for storage space on other peers by offering space in return. There is no mechanism for dealing with peers that delete the files they have been asked to store.

Storage relationships in the Samsara [57] backup system do not have to be symmetric – instead, each peer that is asked to store a block issues an equal-sized 'claim' in return. The claim is generated using a secret key, so the owner does not need to keep a copy of it. Peers that are storing blocks periodically test the holders of the corresponding claims, discarding the blocks if the claims have been discarded. There are no third-party audits [189].

Claims consume up to half of Samsara's storage capacity, so to save space a peer can forward a claim to a peer that asks it to store a block, rather than manufacturing a new claim. If a claim is forwarded in a loop then it occupies no space because the owner does not need to store it, but simulations show that this rarely happens in practice.

Ackemann *et al.* [2] describe how reciprocation can be extended beyond a single resource or service, by looking for 'bartering rings' in which each node provides a service to the next node in the ring at a level determined by the quality of service received from the previous node. The goal is mutual benefit rather than *quid pro quo* fairness, since it may not be possible to compare the values of different services. Gauthier *et al.* [99] suggest a similar scheme; Anagnostakis and Greenwald [8] provide a protocol for finding suitable rings in a file-sharing network.

GNUnet's economic model [111] aims to allocate resources in such a way that people with no resources to offer can use the network's spare capacity, without making the network vulnerable to denial of service attacks. Nodes pay their neighbours to forward queries; payment balances exist between pairs of nodes but are not transferable, so no monetary system is required. Priority is given to 'paid' queries, and spare bandwidth and storage are allocated to 'unpaid' queries. Thus a denial of service attack can only use the network's excess bandwidth and storage unless the attacker contributes resources to the network, which would defeat the purpose of the attack.

SWIFT [250] uses payment balances between pairs of nodes to encourage nodes to upload files. Each node counts the number of bytes sent and received on each connection, and uploads in round-robin order to those neighbours with positive credit balances. Simulations show that nodes benefit by allocating a small fraction of their bandwidth to 'largesse', which is distributed equally among all neighbours, because this helps to avoid deadlock.

A.3.5 Game Theory

If a network is viewed as a group of self-interested individuals interacting according to rules specified by the protocol designer, then game theory provides tools for identifying vulnerabilities to free riders and attackers, and mechanism design can be used to create protocols that resist attack and reward cooperation.

Distributed algorithmic mechanism design [86] focusses on mechanisms that are computationally tractable and can be implemented in a distributed fashion. Shneidman and Parkes [237] give a brief overview of open questions in distributed algorithmic mechanism design and discuss the problem of rational behaviour in peer-to-peer networks.

Simple games can embody surprisingly complex problems, and perhaps no such game has received more attention than the prisoner's dilemma [155]. Formally, the prisoner's dilemma is a single-round game for two players. Each player chooses between two actions called cooperation and defection, and receives a payoff that depends on the choices of both players: T is the 'temptation' payoff for unilateral defection, R is the 'reward' payoff for mutual cooperation, P is the 'punishment' payoff for mutual defection, and S is the 'sucker' payoff for unilateral cooperation. The dilemma arises because the payoffs are in the order $T > R > P > S$, which means a rational player will defect unless she has reason to believe that her opponent intends to cooperate. Both players therefore defect, leading to a suboptimal payoff for both players, since $R > P$.

Axelrod [17] describes two computer programming tournaments in which programs submitted by game theorists from around the world competed in an iterated version of the prisoner's dilemma. In the iterated game, players face each other for an unknown number of rounds. Automatic defection is no longer necessarily the best strategy, because players have the chance to recognise cooperative opponents and gain a higher payoff through mutual cooperation. However, each player will be tempted to defect once her opponent has started to cooperate, in order to claim the highest payoff for unilateral defection.

The winner of Axelrod's first tournament was a simple program called Tit For Tat. This program cooperates in the first round, and in subsequent rounds copies its opponent's choice from the previous round. Thus it 'punishes' defection with a single round of defection, and 'rewards' cooperation with a single round of cooperation. The simplicity of this strategy made its success surprising, and in the second tournament many programs tried to exploit weaknesses of Tit For Tat that had been identified in the first tournament, such as its tendency to fall into cycles of mutual punishment. However, Tit For Tat also won the second tournament, suggesting that it is a stable and versatile strategy.

No matter what the length of the game, Tit For Tat loses against a program that always defects, because Tit For Tat cooperates in the first round. However, against a mixed population of strategies, Tit For Tat tends to do better overall than any other strategy. This is particularly true when the population contains a few other instances of Tit For Tat, since these instances will gain enough benefit from their occasional cooperative encounters with one another to offset their losses when they encounter defectors. In an evolutionary simulation, a small cluster of Tit For Tat players was able to invade a population containing a mixture of strategies, and it has been proven that a population of Tit For Tat players can resist invasion by any other single strategy, meaning that Tit For Tat is an evolutionarily stable strategy (ESS) [241].

Axelrod claims that Tit For Tat's success is due to four characteristics:

1. **Niceness:** Tit For Tat starts by trying to cooperate, and only ever defects in response to a defection from its opponent. Thus it always does well against other nice strategies.
2. **Responsiveness:** Tit For Tat reacts quickly to any change in its opponent's behaviour, minimising its losses with an opponent that defects and maximising its gains with an opponent that cooperates.
3. **Forgiveness:** Tit For Tat never punishes a single defection for more than one round. Less forgiving strategies are more likely to be caught in cycles of mutual punishment.
4. **Clarity:** Tit For Tat's response to any behaviour is easy to predict. While this might seem to leave it vulnerable to exploitation, it actually encourages cooperation since the best long-term strategy when playing against Tit For Tat is to cooperate.

Huberman and Glance [127] find that the emergent properties of simulations like Axelrod's can change drastically if the nodes are updated asynchronously rather than synchronously. Initial conditions that produce a complex and constantly changing mixture of cooperation and defection under synchronous updating converge to universal defection under asynchronous updating.

Félegyházi *et al.* [89] apply a variant of Tit For Tat to the forwarding problem in multi-hop networks. In their model, each node is the source of a single flow and can measure its end-to-end throughput. A node considers itself to be playing against the rest of the network and does not distinguish between nodes, thus avoiding the need for authentication. If the number of packets delivered on a node's behalf divided by the number of packets it forwards is above a certain threshold, the node continues to forward packets (this is called cooperation). Otherwise the node drops packets (this is called defection). A node's strategy can be expressed in terms of the threshold value: zero means the node always cooperates, while an infinite threshold means the node always defects. A threshold value of one divided by the average number of relays per route is called Tit For Tat, while any lower threshold is called Generous Tit For Tat. (However, without authentication a node cannot discover the number of relays per route and therefore cannot know whether or not it is being generous.)

Simulations are used to discover conditions under which cooperation can emerge without incentive mechanisms, first in a ring topology and then in a simulated mobile wireless network. In the ring topology, if all nodes play Tit For Tat then the result is universal cooperation, but if a single node defects then cooperation collapses. It is not in any node's interest to defect in such a situation – doing so can only harm its own throughput – so it is a Nash equilibrium [101] for all nodes to play Tit For Tat. Another equilibrium exists in which all nodes defect. It only takes a single defection to start the slide from full cooperation to full defection, so the cooperative equilibrium is very sensitive to noise.

In the wireless simulation, the cooperative equilibrium is rarely achieved using Tit For Tat, because routes have different lengths and some nodes have to forward more than their fair share of packets. If these nodes play a strict Tit For Tat strategy then they will start to defect, reducing the throughput of other nodes which will start to defect in turn. Cooperation can still be achieved if some or all of the nodes play Generous Tit For Tat strategies. The level of generosity required decreases as mobility increases, intuitively because mobility redistributes the load among the nodes, reducing the chance that a node will have to forward more than its fair share of packets for a long period.

A later paper by the same authors [88] considers continuous strategies, in which a node's level of cooperation can vary continuously between 0 and 1 [256]. As in the previous paper, each node treats the rest of the network as a single rational opponent. This approach lends itself to exploitation by rational nodes, which may be able to defect while another node suffers the consequences.

Urpi *et al.* [254] provide another game theoretic analysis of the forwarding problem, including the influence of mobility on cooperation. As mobility increases, nodes interact for shorter periods, making future reward or punishment less likely – in game theoretic terms, future payoffs are discounted. Nodes are therefore less inclined to cooperate

as mobility increases, in contrast to the model of Félegyházi *et al.* in which mobility makes cooperation more likely. Urpi *et al.* make the strong assumption that nodes can tell whether their neighbours are forwarding packets, which may not be possible in real wireless networks due to interference, variable-power transmitters and hidden terminals. They also use a utility function that could be said to beg the question, because a node derives utility from the delivery or further forwarding of the packets it forwards, as well as from its own throughput – this is not selfishness in the conventional sense of the word.

An important observation made in this paper is that a selfish, power-limited node maximises its satisfaction not only by avoiding forwarding packets for other nodes, but by avoiding sending its own packets if there is a low probability that they will reach their destination.

Leino [162] compares the energy cost of participating in a multi-hop wireless network to the cost of transmitting packets directly to the recipient. If free riders can be detected and excluded from the network then it is in the interest of most nodes to participate in forwarding. Nodes near the centre of the network are less likely to benefit from participating, due to higher forwarding load and lower average distance to recipients. Routing protocols that discover minimum-energy routes are more likely to encourage participation than protocols that discover minimum-hop or minimum-delay routes.

Feldman and Chuang [87] point out that if a packet fails to reach its destination in a multi-hop network, the source of the packet may not know which intermediate node is responsible for dropping it. Nevertheless it is shown that, if the source offers a payment to each relay that is conditional on the end-to-end delivery of the packet, it is possible to induce an equilibrium in which all nodes cooperate without per-hop monitoring. Transmission costs for each node and loss rates for each link are assumed to be known to the sender. The payment mechanism is not discussed – it is assumed that the source does not default on payment once its packet has been delivered.

A.4 Decentralised Communication Networks

Peer-to-peer overlays and mobile ad hoc networks have been the focus of much recent research. These decentralised networks share three characteristics that set them apart from traditional communication networks: open membership, autonomous nodes, and the absence of central administration. The resulting issues of self-organisation, scalability and stability are relevant to the design of censorship-resistant networks.

A.4.1 Mobile Ad Hoc Networks

A mobile ad hoc network is a group of wireless nodes that communicate among themselves without infrastructure or manual configuration. Either routes must be discovered on demand, or changes in the topology must be broadcast to all interested nodes. Perkins [199] and Hong *et al.* [122] provide useful reviews of the proliferation of ad hoc routing protocols, three of which have been adopted as internet standards [172]. Broch *et al.* [30] and Lee *et al.* [161] compare the performance of various ad hoc routing protocols at different levels of mobility.

Traditional routing protocols use address aggregation to reduce the number of routes that each node must store. For example, IP addresses are aggregated by prefix. Address aggregation breaks down if the nodes are mobile, which tends to limit the scalability of ad hoc routing protocols. Location-based routing [150, 142, 266] mitigates this problem by using a lookup service to map addresses onto physical locations. Packets can then be delivered using a stateless greedy forwarding algorithm. L+ [48] uses a similar approach, based on topological rather than geographic locations. Unfortunately these techniques are not suitable for anonymous or unlinkable communication, as they reveal the locations of the communicating parties.

Another factor limiting the scalability of ad hoc networks is the need to disseminate routing information to all nodes, whether in the form of proactive updates or reactive route requests. Many protocols use flooding with duplicate suppression, which has the advantage that it discovers the quickest route from the source to each node that receives the message. However, flooding is inefficient in areas of high node density, where the number of links can be far larger than the number of nodes. Span [47], SBA [198], and multipoint relaying [208] are techniques for reducing broadcast redundancy.

A dominating set of a graph is a subset of the nodes such that every node is either a member of the dominating set or has an edge joining it to a member of the dominating set. A connected dominating set forms a natural routing backbone for an ad hoc network. Das and Bharghavan [65] and Wu and Li [265] look at heuristic methods for finding small connected dominating sets.

A.4.2 Peer-to-Peer Networks

A peer-to-peer network is an internet overlay composed of connections between autonomous peers. Direct communication between any two peers is possible in principle, but maintaining a connection between every pair of peers is inefficient, particularly if peers join and leave the network at a high rate. Instead, most peer-to-peer systems construct a multi-hop search overlay to allow peers that wish to communicate to find one another. A direct connection is then established for data transfer.

Peer-to-peer search overlays can be divided into two major families. In the first family, often referred to as unstructured peer-to-peer networks, the topology of the overlay is only loosely controlled and searches are nondeterministic: matching items are not guaranteed to be found. In the second family, referred to as structured peer-to-peer networks, tighter control of the overlay topology allows scalable, deterministic searches.

Risson and Moors [216] give a comprehensive survey of the peer-to-peer search literature, and Lua *et al.* [168] provide a survey of deployed peer-to-peer systems.

JXTA [108, 139, 31] is a set of protocols intended to provide a cross-platform, interoperable, ubiquitous substrate for peer-to-peer computing. However, no application has so far demonstrated the advantages offered by this abstraction – separate peer-to-peer applications do not necessarily benefit from operating on a shared substrate.

Unstructured Peer-to-Peer Networks

Peers in the Gnutella file-sharing network [103] form an unstructured overlay in which messages are broadcast for a limited number of hops. The overlay is primarily used for searching; files are transferred by direct connections between peers. The original Gnutella software has been withdrawn, but many other file-sharing programs implement the Gnutella protocol [100].

The use of flooding limits Gnutella's scalability: if the query rate is too high then low-bandwidth nodes begin to drop messages, effectively causing the overlay to fragment [217]. Restricting the range of broadcasts raises the query rate at which fragmentation occurs, but it does not solve the underlying problem.

Miconi [179] compares the cost of broadcasts in trees and general graphs, and points out that loops in the Gnutella topology cause a large number of wasted messages due to the 'crossing letters' effect. A network with three connections per node is optimal in terms of the number of nodes reached per message, but the topology is likely to become fragmented if a single connection breaks. Miconi proposes idle connections, which serve to keep the overlay connected but do not carry search messages unless the number of active connections drops below three.

Searches in LightFlood [137] are restricted to locally constructed spanning trees after the first few hops, allowing them to reach the same number of nodes as flooding while using fewer messages.

Lv *et al.* [169] investigate the effect of replacing Gnutella's broadcast queries with multiple 'random walkers'. A walker is a message that is forwarded randomly for a limited number of hops, or until it returns an adequate number of results. Various strategies for replicating objects are also investigated. (An 'object' could be a file, or information about the location of a file.) It is shown that an ideal replication strategy creates a number of copies that is proportional to the square root of the frequency with the object is requested. A simple way to implement this strategy is to replicate the object on every node that forwards it back to the requester. To maintain the square root relationship, the cache replacement policy should not depend on object popularity – random replacement or first-in-first-out would be suitable, but least-recently-used would not.

Lv *et al.* [170] suggest combining random walkers with flow control tokens and a topology adaptation algorithm that redirects traffic away from overloaded peers. Chawathe *et al.* [46] develop these techniques further in a new file-sharing system called Gia, which uses biased random walkers, flow control tokens, topology adaptation and one-hop replication of file information.

Adaptive Probabilistic Search [252] uses biased random walkers that are forwarded probabilistically depending on the success of previous searches. The result is that searches tend to be directed towards nodes that return a large number of results, without requiring explicit knowledge of node capabilities or the overlay topology.

Yang and Garcia-Molina [268] compare four search techniques for unstructured networks. Breadth-first search, the method originally used by Gnutella, is the standard by which the other techniques are measured. Iterative deepening uses breadth-first searches of increasing depth until a satisfactory number of results are obtained. Directed breadth-first search only forwards queries to neighbours that have previously returned good results. The fourth technique, one-hop replication of file information, is the only technique that reduces aggregate bandwidth and processing requirements compared with breadth-first search without affecting the response time or the number of results.

The original Gnutella protocol has been extended to take advantage of heterogeneity by using a two-tier structure [132, 100]. Depending on its capabilities, each peer operates as either a 'leaf' or an 'ultrapeer'. An ultrapeer acts as a server to a number of leaves and as a peer to other ultrapeers and legacy peers. Overlay connections are not established between leaves, reducing the load on low-bandwidth peers. The proprietary FastTrack protocol used by Kazaa, Grokster and iMesh appears to have a similar structure [115].

Yang and Garcia-Molina [269] develop a detailed cost model for a Gnutella-like protocol and use it to derive rules of thumb for the design of two-tier networks.

In Gnutella's query routing protocol [220], which is a generalisation of a scheme suggested by Prinkey [207], a peer avoids receiving unnecessary queries by telling its neighbours which queries it is able to answer. This is done by creating a list of keywords (eg from the names of shared files) and compressing the list into a Bloom filter [27].

Dynamic querying [93] is another extension to the Gnutella protocol that aims to improve its scalability. When an ultrapeer receives a query from a leaf, it queries its neighbours one at a time until a satisfactory number of results have been returned, rather than immediately forwarding the query to all its neighbours. This reduces the bandwidth used for finding popular items, allowing searches for unpopular items to reach more peers.

Ripeneau *et al.* [215] used a 'crawler' to explore a large Gnutella network between November 2000 and May 2001, a period of considerable growth. They found significant variation in the lifetime and connectivity of nodes. The number of connections per node followed a power law distribution [3] in November 2000, but nodes with very few connections were rare by March 2001, leading to a 'hump' in the distribution around 10 connections. The authors speculate that this might make the network more robust against adversarial failures than a scale-free network [7]. However, Saroiu *et al.* [226] show that removing 4% of the nodes from a sampled Gnutella topology is enough to disconnect the network if high-degree nodes are chosen, whereas 30% of the nodes can be removed without causing disconnection if the selection is random.

Another interesting finding is that the length of time a node can be expected to remain active is proportional to the length of time it has been active so far – old nodes are likely to outlive young nodes. Bustamante and Qiao found a similar power law lifetime distribution in 2003 [35].

Stutzbach *et al.* [248] performed a series of much faster crawls of the Gnutella network between September 2004 and February 2005. Slow crawls can distort the measured degree distribution, because long-lived nodes may be reported as neighbours by a large number of short-lived nodes that are present at different times. The topologies recorded by Stutzbach *et al.* do not have power law degree distributions and are not vulnerable to the removal of high-degree nodes.

The skewed lifetime distribution leads to a network with a small, stable core of old nodes and a large, unstable periphery of new nodes. The emergent core prevents the network from fragmenting under churn.

Sarshar *et al.* [227] present a query routing protocol for networks with power law degree distributions. Each node's list of files and each query are propagated using random walks. A random walk on a scale-free network will usually reach a high-degree node, and the high-degree nodes will usually form a connected subgraph. Percolation theory shows that probabilistic flooding within this connected subgraph can be expected to reach all of the high-degree nodes using a relatively small number of messages, allowing any query to reach any list of files. However, the high-degree nodes must also have high capacity, since they carry most of the query traffic.

Sarshar and Roychowdhury [228] describe an algorithm for maintaining a power law degree distribution in the face of churn.

LMS [183] is a lookup protocol for unstructured topologies that is conceptually similar to the Freenet routing algorithm [53], except that instead of trying to create a global minimum or epicentre for each key, files and queries are replicated across multiple local minima. This means there is no need for topology adaptation. Each replica is randomly forwarded for a small number of hops and then deterministically forwarded to the nearby node with the identifier that most closely matches its key. Analysis and simulation show that a relatively small number of replicas can provide reliable performance in the face of random and adversarial failures.

Roussopoulos and Baker [222, 221] propose that popular metadata should be 'pushed' across the search overlay towards requesters. This reduces traffic on an overlay connection if the pushed item would have been requested at least once on that connection, so the decision to propagate should be based on expected popularity.

Konspire2b [151] also uses a push model: peers subscribe to authenticated broadcast channels and form ad hoc distribution trees when broadcasters wish to send files.

Structured Peer-to-Peer Networks

Collision-resistant hashing allows any file to be identified by a fixed-size hash that is unique with high probability. File hashes can be used to search for specific files in unstructured peer-to-peer networks, but there is no guarantee that a matching file will be found if it exists in the network. In contrast, many structured peer-to-peer networks implement a 'distributed hash table' (DHT) abstraction, providing a deterministic exact-match lookup service by structuring the overlay so that all lookups for the same hash value converge on the same small set of peers.

Plaxton *et al.* [204] present a method of locating a nearby copy of a resource in a distributed system. All nodes and resources are given unique identifiers of equal length. A request for a resource is forwarded by resolving the resource's identifier one digit at a time. If the request cannot be satisfied at the present node, it is forwarded to a node with an identifier that matches more initial digits of the resource's identifier. The size of each node's routing table is equal to the number of digits in an identifier times the base of the digits, regardless of the size of the network.

For each resource, one can imagine a spanning tree of the network rooted at the node with the identifier that matches the resource's identifier in the greatest number of initial digits. This node stores the location of the resource, and other nodes in the tree may store the locations of nearby copies. A request for the resource will travel up the tree, eventually reaching the root unless it first encounters a closer copy of the resource.

Tapestry [276] is a peer-to-peer overlay that uses the prefix routing algorithm of Plaxton *et al.*, adding methods to deal with node arrival, departure and failure, and with mobile resources. Pastry [80] is very similar to Tapestry, although they differ in the way each node keeps track of nearby nodes in the identifier space.

Castro *et al.* [39] examine three techniques for adapting a structured overlay's topology to the topology of the underlying network, to minimise traffic overhead and latency. Improved procedures for neighbour selection and topology maintenance in Pastry are presented.

Kademlia [176] uses a longest-prefix routing algorithm similar to Pastry's, with the distance between two identifiers defined as their bitwise exclusive or (XOR), interpreted as an integer. Kademlia tolerates churn by exploiting the discovery of Saroiu *et al.* [226] that old nodes are likely to remain in a peer-to-peer network for longer than new nodes. Nodes in each 'routing bucket' are therefore sorted by decreasing age, with new nodes added to the tail of the list and unresponsive nodes flagged and eventually removed when the bucket fills up. This provides stability in the face of high node turnover, without flushing the routing table if the node temporarily loses connectivity.

Routing data is learned from queries, so nodes that issue a large number of queries will tend to be widely known and will therefore receive a large number of queries in return, as long as they remain responsive. This could allow Kademlia to benefit from heterogeneity in node capabilities, whereas other distributed hash tables aim for uniform load balancing.

Kademlia overlays are used in some file-sharing networks [83, 85, 156], including decentralised variants of BitTorrent [18].

Chord [246] is a distributed hash table in which the key space can be imagined as a circle, and keys are hashed so that records are evenly distributed around the circle. A node is assigned an identifier by hashing its IP address, and it takes responsibility for the arc of the key space between its identifier and the identifier of the nearest anticlockwise node. Each node must store all records that fall into its arc.

When a node joins the network it calculates its identifier, contacts the nearest clockwise and anticlockwise nodes, and inserts itself between them. It inherits a number of stored items from its clockwise neighbour. Messages are routed in a clockwise direction using greedy forwarding. Lookups will succeed as long as every node maintains connections to its clockwise and anticlockwise neighbours, but for more efficient routing each node also maintains connections to *O(log N)* neighbours chosen so that the clockwise distance to the first neighbour is half the circumference of the circle, the distance to the second is a quarter of the circumference, and so on.

Chord provides a single operation, `lookup(id)`, which returns the address of the first node clockwise from the specified identifier. This operation is used to store and retrieve records, and to initialise and maintain each node's connections to its neighbours. Lookups can be implemented recursively (the current hop forwards the message to the next hop) or iteratively (the current hop returns the address of the next hop).

In a stable network, Chord has logarithmic worst-case message complexity for storage and retrieval, and a logarithmic load imbalance between nodes. These properties may not hold with a constantly changing population of nodes, however.

CAN (Content-Addressable Network) [209] is a structured overlay similar in functionality to Chord. CAN's multi-dimensional key space corresponds to the surface of a torus rather than the perimeter of a circle. Each node takes responsibility for a region of the key space and maintains connections to the nodes responsible for adjacent regions, as well as a few long-distance connections to allow efficient routing of messages. The latency of links is taken into account when choosing a region, with the aim of decreasing the load on the underlying network by assigning adjacent regions to nearby nodes.

The use of multiple simultaneous coordinate spaces (called 'realities') is discussed, as is the idea of putting each region under the control of a cluster of peers rather than a single node. These techniques could make CAN more robust at the cost of added complexity.

Symphony [173] is a distributed hash table similar to Chord, but with fewer long-distance links per node. This is supposed to reduce the cost of nodes joining and leaving the network. The distribution of links is based on Kleinberg's small world model [146]: a node maintains links to its closest neighbours in the identifier space, and to a small number of distant nodes with identifiers drawn from a harmonic distribution. Nodes must estimate the size of the network in order to distribute their links correctly. Unlike Chord, messages can be routed in either direction around the circle. Symphony also uses lookahead to reduce routing latency: nodes exchange neighbour lists and forward each message to whichever neighbour has a neighbour closest to the target.

Simulations show that Symphony's latency (measured by number of hops) is comparable to Tapestry and Pastry with a similar number of TCP connections, and lower than CAN or Chord. However, Symphony does not take the topology of the underlying network into account when selecting neighbours, so actual latency is likely to be higher than with Pastry or Tapestry.

Castro *et al.* [38] consider attacks against distributed hash tables, such as a node adopting a particular identifier in order to make itself responsible for a particular key, allowing the attacker to delete, modify or monitor the corresponding record. They propose a solution based on certified identities and redundant routing.

Most distributed hash table designs assume that single nodes join a well-known network by contacting an existing member. They do not address the problem of merging multi-node components after a partition. Single nodes could migrate from one component to the other, but without agreement as to which component is the 'official' network, the migration might never end. Reaching such an agreement may not be possible in a decentralised network [92].

SkipNet [116] is a structured overlay based on skip graphs [13] that can recover from partitions. Unlike distributed hash tables, SkipNet allows any object to be placed on any node. Nodes are arranged in the overlay according to their DNS names, and requests for an item located within the same organisation as the requester can be guaranteed not to pass outside the organisation. Local queries can be routed while the organisation is disconnected from the rest of the network, and the organisation's SkipNet segment can be reconnected to the rest of the overlay when connectivity is restored.

Law and Siu [159] take a different approach to the construction of scalable overlays. An H-graph is a regular random graph in which every node belongs to d Hamilton cycles (a Hamilton cycle on a graph visits every node exactly once). H-graphs have low expected diameter and are resilient to random node failures. A node joins the network by using random walks to insert itself at a random position in each cycle, and can leave the network with $O(d)$ messages. Crashes are dealt with by periodically building new cycles from the remaining nodes. Layered H-graphs can be used for prefix routing without the strict topology control of Pastry or Tapestry, although the expected number of hops is larger.

Ledlie *et al.* [160] describe a peer-to-peer network that uses Bloom filters [27] to summarise the files held by each node; files are identified by content hashes. Nodes organise themselves into ad hoc trees, with the root of each tree maintaining a combined Bloom filter summarising the files held by the entire tree. The filters are used to direct searches within and between trees. This is similar to Prinkey's tree-based scheme for keyword searches [207].

Freedman and Vingralek [95] suggest that skewed query distributions [244] could make uniform, static partitioning of the identifier space undesirable for peer-to-peer systems. Instead they propose a distributed trie that replicates key/value mappings on nodes that request them and lazily updates routing information to maintain the trie, piggy-backing routing information on responses to queries.

Private Peer-to-Peer Networks

The peer-to-peer systems described so far do not consider the real-world identities of the participants – anyone can share files in Gnutella or handle queries in Chord. This is both a strength and a weakness: it enables wide

participation, ensuring that a large amount of content is available in file-sharing networks for example, but it could also allow attackers to enter, monitor, and disrupt peer-to-peer networks. Highly publicised lawsuits brought by copyright holders against users of file-sharing networks have led to increasing interest in private, authenticated sharing between groups of friends [25]. Groupware designed for business collaboration has been adapted to this purpose, and new invitation-only file-sharing networks have been created.

While this development is interesting from a social point of view, it has so far received relatively little attention from the research community. Consequently, most of the references in this section are to websites rather than peer-reviewed papers.

Groove [253, 110] is a groupware application for creating 'shared spaces' that can span organisational boundaries. Each member of the group maintains a copy of the shared space's state, and encrypted deltas are transmitted to other members when the state changes. It is not necessary to maintain connections between every pair of members, and indeed firewalls may make this impossible; members who are unable to communicate directly can exchange messages through dedicated relays. Changes to the shared space can be made while members are offline, and synchronised when they reconnect.

Two kinds of shared space can be created. In a mutually trusting space, all changes to the state are authenticated using a single key. This makes it possible for members to spoof updates from other members. In a mutually suspicious space an authentication key is generated for each pair of members, preventing spoofing but increasing the size of update messages.

Members can only join groups by invitation. The inviter is responsible for communicating the new member's encryption and authentication keys to the group – a man-in-the-middle attack is possible at this point, so spoofing is possible even in mutually suspicious spaces. Any member can evict any other member from a group, which is done by creating a new group key and transmitting it to all members except the evicted member.

Strufe and Reschke [247] describe an overlay network that stores group information as well as file information, so users can create authenticated groups for file-sharing and instant messaging. Each group has a single owner who is responsible for adding and removing members.

Grouper [112] is similar to Groove, but instead of shared spaces it offers encrypted chat, file-sharing and audio streaming within groups. A connection to a central server allows peers to discover one another's current IP addresses, and the server can be used as a relay by firewalled peers. Shinkuro [236] also uses a central server for peer discovery and relaying. Peers on the local network can be discovered using Zeroconf [273]. Aimster [6] used a central server to enable file-sharing and instant messaging between friends.

WASTE [260, 84] is an encrypted peer-to-peer network created by Justin Frankel, the author of Gnutella. It takes its name from the underground postal service in Thomas Pynchon's novel *The Crying of Lot 49*. Like Gnutella it supports broadcast messages and reverse-routed replies; these are used to implement keyword searches, file transfer and chat. Peers relay one another's messages if all-to-all connectivity is not possible. Links are encrypted and optionally padded to a constant traffic level, but there is no end-to-end encryption or authentication, so peers can eavesdrop on one another and spoof messages. There are no group keys, so a peer can only be evicted by removing its key from every peer in the network.

Turtle [205] is a friend-to-friend network, meaning that users only connect to people they know in real life [29, 109]. Searches are broadcast through the overlay; search results and files are forwarded by reverse-routing. An economic model to encourage sharing is briefly discussed.

The next version of Freenet will also be a friend-to-friend network, to prevent attackers from collecting lists of Freenet nodes [52]. This will require a new routing algorithm, since the current algorithm depends on nodes learning addresses from successful queries (see Section A.1.5).

The new algorithm will use a circular address space; nodes will swap locations to arrange themselves on the circle in a way that allows greedy forwarding [224]. However, to the extent that this process succeeds in finding a one-dimensional embedding of the social network, it will also undermine anonymity: two nodes that are close together on the circle are likely to have owners who are close together in the social network, so an attacker who knows the owner of one node may be able to guess the owners of nearby nodes.

A.5 Social Networks

Milgram [180] was one of the first researchers to study the structural characteristics of social networks, in which the nodes represent people and the edges represent their social relationships. In one experiment, subjects were asked to

forward a letter towards a target person who was unknown to them personally, using only the target's location and occupation. The letters that reached the target were forwarded an average of six times, giving rise to the popular notion of 'six degrees of separation'. However, the great majority of letters never reached the target. Kleinfeld [149] criticises the methodology of Milgram's studies, but more recent studies using email have produced similar results, with an estimated median of five to seven degrees of separation between email users worldwide [76].

One of the difficulties with studying social networks is defining what constitutes a connection between two people. Technologically mediated social networks do not suffer from the same ambiguity, and as a result large amounts of data have be collected from sources such as scientific publication lists and social networking websites. These networks appear to have short average path lengths and heavy-tailed degree distributions, with nearly all of the nodes belonging to a single strongly connected component [188, 178].

The clustering coefficient of a graph describes the probability that two neighbours of the same node are neighbours of one another. Many real-world networks have clustering coefficients that are much higher than would be expected for a random graph of the same size. Iamnitchi *et al.* [129] show how a high clustering coefficient can result from viewing a bipartite graph as a unipartite graph. For example, a bipartite graph of actors and films, in which edges connect actors to the films in which they have appeared, can be reduced to a unipartite graph in which edges connect actors who have appeared in the same film. If the average film contains several actors then the resulting graph will have a high clustering coefficient.

A.5.1 Small World Networks

Watts and Strogatz [262] gave the first graph-theoretic treatment of the 'small world phenomenon', in which members of a large social network turn out to be separated by a small number of links. The authors created small world networks by adding a small number of random connections to a regular lattice. Like lattices, the resulting graphs were highly clustered – many pairs of neighbours had neighbours in common – but like random graphs, there was a small average distance between nodes.

Watts' subsequent book *Small Worlds* [261] is a thorough examination of small world networks, from a description of their characteristics to a model of their structure, and finally an account of their appearance in real-world systems such as power grids and the neural networks of nematode worms.

Kleinberg [146, 147] examines the problem of finding short paths in small world graphs using only local information. The graphs in question are augmented lattices similar to those described by Watts and Strogatz. Each node u has short-range connections to every node within p lattice steps, and q long-range connections. Each long-range neighbour v is selected with a probability proportional to $r^{-\alpha}$, where r is the lattice distance between u and v, and α is referred to as the graph's *clustering exponent*. Kleinberg shows that a decentralised greedy search algorithm can find a short path between two nodes in polylogarithmic time only if the clustering exponent of the graph is equal to the dimensionality of the underlying lattice. For any other value of α, no greedy search algorithm can succeed in polylogarithmic time.

Kleinberg then extends his results to small world graphs derived from a tree structure, and introduces a general model of group structures that covers both lattice- and tree-based small world graphs [148]. The restriction on search algorithms from the previous paper is shown to hold for the general model.

Searches in the general model do not make use of the lattice or tree edges – only the long-range edges are used. In fact Kleinberg's findings can be applied to any graph embedded in a metric space. To support a decentralised greedy search algorithm, the length of the edges according to the chosen metric must follow a power law distribution with exponent equal in magnitude to the number of dimensions in the metric space. Kleinberg's model assumes that all the nodes have equal degree, but Franceschetti and Meester [94] show geometrically that a similar result holds for arbitrary degree distributions.

A.5.2 Scale-Free Networks

A scale-free network [22] has a degree distribution that follows a power law rather than the Poisson degree distribution of random graphs. Like small world networks, scale-free networks tend to have small diameters relative to the number of nodes. However, scale-free networks do not necessarily have the same clustering properties as small world networks.

One simple way to generate a scale-free topology is to simulate the growth of a graph, attaching each new node to a number of existing nodes. If the probability of attaching to a node is proportional to its degree, the resulting network will have a power law degree distribution. Similar multiplicative growth models have been used in many other contexts to explain power law distributions [42, 98, 181, 238]. In fact there is a fundamental connection between

multiplicative growth processes and heavy-tailed distributions, which is related to the connection between additive growth processes and the normal distribution that is described by the central limit theorem [98, 242].

Sarshar and Roychowdhury [228] show that graphs generated by growth with preferential attachment do not retain their scale-free structure if nodes are randomly deleted as the graph evolves. They present a simple rewiring rule to maintain the scale-free structure: every time a node loses a link, it creates a new link using preferential attachment. However, the rule depends on each node losing links at a rate proportional to its degree, which means that nodes must be deleted uniformly at random. It is not clear whether the rule would have the same effect in a network where the distribution of node lifetimes is also heavy-tailed [226, 35].

Albert *et al.* [7] show that scale-free networks are resilient to random node failures: if the magnitude of the power law exponent is less than 3, almost all of the nodes can be removed without disconnecting the network. However, scale-free networks are relatively vulnerable to targeted attacks. If an attacker can identify the small number of high-degree 'hubs' that join the network together, the network can be disconnected by removing relatively few nodes. (This is less surprising when one considers the number of links removed by such an attack.)

Motter *et al.* [185] consider attacks on individual links in scale-free networks. Long-range links are responsible for the small world effect in the model of Watts and Strogatz, but the same is not true for scale-free networks: removing short-range links has a greater effect on the average distance between nodes than removing long-range links.

This result can be explained by considering the number of shortest paths that include each link. In a scale-free network, pairs of high-degree nodes tend to be separated by fewer links than average. Intuitively, this is because a high-degree node has more 'chances' than a low-degree node of having a short path to a given destination. The global result is a small, well-connected core consisting of high-degree nodes and short-range links, and a large, poorly connected periphery consisting of low-degree nodes and long-range links. Links in the core tend to be involved in a large number of shortest paths, so removing them affects a large number of paths and thus has a large effect on the average path length, even though the effect on each path is not great because of the short range of the links.

B Details of the Proposed Design

B.1 A Utility-Based Model of Reciprocation

Reciprocation between selfish nodes can be modelled using the economic concept of utility-maximising behaviour. In this model, nodes request services from their neighbours and can measure the level of service received. Each node attempts to maximise the benefit it receives from its neighbours while minimising the cost it incurs by providing services.

Every action a node can perform has an estimated cost and one or more possible outcomes. Each outcome has an estimated benefit and an estimated probability of occurring [135]. The sum of the probabilities must be 1, ie each action has exactly one outcome. Costs and benefits may be positive or negative, and they are subjective – it is never necessary to compare the estimates of one node to those of another node. Furthermore, we do not assume that costs are necessarily measured in the same units as benefits.

The *expected benefit* of an action is defined as the mean benefit of all possible outcomes, weighted by probability. An action's *expected utility* is defined as the expected benefit per unit of cost. When faced with a choice of actions, a *selfish node* is defined as one that always chooses the action with the highest expected utility, according to its own estimates.

If a node expects that providing a service to a neighbour will result in a higher level of service being provided in return, it can weigh the expected benefit of the reciprocation against the cost of providing the service.

To estimate the benefit of obtaining reciprocation from a neighbour, a node makes the assumption that all the benefit received from the neighbour so far is a result of reciprocation – in other words it assumes that the neighbour is selfish. This means that the benefit received from the neighbour can be attributed to the services provided to it. The total benefit received divided by the number of services provided tells the node the average benefit obtained by providing each service, which is the expected benefit of providing another service.

When evaluating a request in this way, a node does not know the benefit of the requested service from the neighbour's point of view – costs and benefits are subjective – but it can estimate the cost to itself of providing the service, and the benefit to itself of the resulting reciprocation. If it is possible to measure costs and benefits in the same units then the cost of obtaining reciprocation can be subtracted from the expected benefit, and a node may decide not to provide a service if the expected benefit is less than the cost. However, unlike standard models of expected utility [229, 135],

the present model does not require that costs and benefits can be compared in this way. Instead, by evaluating the expected utility of serving each request, a node can prioritise requests to maximise its own benefit by obtaining the most reciprocation per unit of cost.

Reciprocation on the basis of expected utility is rational behaviour for selfish nodes, and encourages them to contribute resources to the network. It may also help to prevent denial of service attacks, since an attacker will not be able to consume the network's resources without contributing resources in return.

B.2 Cooperation over Longer Distances

In some situations the requester and provider of a service might not be neighbours. To encourage cooperation in these situations, we can construct a chain of single-hop interactions, with reciprocation occurring at each link in the chain. Each node provides a service to its upstream neighbour without knowing whether that neighbour is the original requester, requests a service from its downstream neighbour without knowing whether that neighbour is the final provider, and reciprocates with its upstream and downstream neighbours. The final provider creates an *unforgeable acknowledgement* that can be verified by intermediate nodes as well as the requester. Each node returns the acknowledgement to its upstream neighbour, showing that the requested service has been performed. Reciprocation occurs only between immediate neighbours, but it provides an incentive to participate in multi-hop as well as single-hop interactions.

In order for unforgeable acknowledgements to operate without certified identities, requests and responses must be constructed in such a way that relays can verify that the response satisfies the request without having a security association with either the requester or the responder. For example, a request for a data block might contain the cryptographic hash of the block, allowing each relay to verify that the correct block has been returned before forwarding it.

Estimating the utility of an action is more complicated in the multi-hop case than in the single-hop case, because of three factors: reliability, indirect benefits, and indirect costs.

A selfish node attempts to maximise its own benefit without directly considering the benefit of others. Nevertheless, reciprocation gives selfish nodes an incentive to provide services, or to forward requests for services they cannot provide. The indirect benefit of returning an acknowledgement for a forwarded request is the same as the expected benefit of providing the service directly, and is measured in terms of the (direct or indirect) benefit the node can expect to receive in return.

When a request is forwarded to a downstream neighbour there is a chance that no suitable acknowledgement will be returned, in which case the cost of forwarding the request will have been paid without receiving any benefit. Nodes must therefore take the reliability of the downstream path into account when calculating the expected benefit of forwarding a request (recall that expected benefit is defined as the mean benefit of all possible outcomes, weighted by probability).

Forwarded requests have indirect costs as well as indirect benefits, because every service requested from a neighbour must be paid for with reciprocation. The indirect cost of requesting a service from a neighbour is equal to the total cost of the services provided to it, divided by the number of services requested from it.

As in the single-hop case, the expected utility of forwarding a request is the expected (direct and indirect) benefit per unit of (direct and indirect) cost.

B.3 Unforgeable Acknowledgements

This section describes an unforgeable acknowledgement scheme for multi-hop packet forwarding. The sender and recipient of the packet must share a secret authentication key, but relays do not need to hold any keys.

The sender creates a unique secret for each packet, calculates the image of the secret under a one-way function, and attaches the image to the packet. Relays store the image of the secret when they forward the packet. If the packet reaches its destination unmodified, the recipient generates the same unique secret and sends it as an acknowledgement. Relays can verify the acknowledgement by calculating its image under the one-way function and comparing the result to the stored image. The acknowledgement is forwarded back towards the sender, allowing each node to confirm that its neighbour delivered the packet as requested. However, an unforgeable acknowledgement does not prove who sent the packet, or who acknowledged it, to anyone except the endpoints. This makes it possible to measure reliability without giving up unlinkability.

Implementation using Message Authentication Codes

The function used by the sender and recipient to generate acknowledgements must have the following properties:

- No two packets should have the same acknowledgement
- The recipient must be able to detect modifications to the packet
- Acknowledgements must not leak information about the authentication key

Message authentication codes (eg HMAC [152]) have all the required properties. In this implementation, the one-way function is a cryptographic hash function (eg SHA1 [82]), the unique secret is the MAC of the packet, the hash of the MAC is attached to the packet by the sender, and the recipient sends the MAC as an acknowledgement. Note that relays do not verify the acknowledgement *as a MAC* – they simply hash it and compare the result to the stored hash. However, from the point of view of the final recipient, the hash of the MAC can also be used to authenticate the packet.

This implementation of unforgeable acknowledgements is described in more detail in the accompanying research note [219].

Implementation using a Block Cipher

It is also possible to implement unforgeable acknowledgements using a block cipher. The sender and recipient use their secret authentication key to generate a unique, single-use *acknowledgement key* for each packet. The sender hashes the packet, encrypts the hash with the acknowledgement key, and attaches the encrypted hash to the packet. Relays store the encrypted hash and the unencrypted hash of the packet. If the block cipher can resist a known plaintext attack, which is a standard requirement for block ciphers, then relays cannot learn anything about the key from the plaintext and the ciphertext.

The acknowledgement contains the hash of the packet and the unique acknowledgement key. Relays use the hash of the packet to look up the stored ciphertext, and verify the acknowledgement by encrypting the hash with the unique key and comparing the result to the stored ciphertext. (Note that the acknowledgment key is not the same key used for end-to-end encryption or authentication of the packet!)

Timeouts

Relays cannot store hashes indefinitely while waiting for acknowledgements. At some point, old hashes must be discarded to make room for new ones. Variable timeouts based on the distance from the sender would require a hop counter, undermining the sender's anonymity. On the other hand, fixed timeouts will occur at upstream nodes before downstream nodes, which is unfortunate because an acknowledgement that is returned to an upstream neighbour after the neighbour has discarded the corresponding hash will not earn any reciprocation. However, fixed timeouts also allow the downstream node to work out whether the upstream node will have discarded the hash yet, so the downstream node can avoid a wasted transmission.

Anonymous senders must not give credit for late acknowledgements, otherwise the resulting level of reciprocation could be used to test whether the previous node is the sender, since a relay would already have discarded the hash.

B.4 Evidence-Based Routing

Rather than attempting to find routes across a topology that may or may not exist, we propose an *evidence-based* approach to routing, using proof of reliability to guide forwarding decisions. From a pragmatic point of view, all that matters when deciding whether or not to forward a packet is whether the next hop is likely to deliver the packet. It is not necessary to speculate about the reasons for packet loss, and indeed attempting to do so may allow dishonest users to manipulate the system.

In evidence-based routing, routes are discovered on demand, and forwarding decisions are based on constant feedback. The following sections describe two schemes for evidence-based routing. Both schemes make use of reciprocation and unforgeable acknowledgements, as described in the preceding sections.

B.4.1 Anonymous Datagrams

The first scheme uses flooding to deliver encrypted packets that do not carry source or destination addresses, or anything else to associate them with one another. Each packet is marked with a unique, random-looking label that will be recognised by the recipient. For example, labels could be generated by encrypting a non-wrapping counter with a key shared by the sender and recipient. To avoid decrypting every label, the recipient can calculate the labels it expects to receive in advance and store them in a lookup table.

At first glance, anonymous flooding seems to give every node fair access to the network: reciprocation prevents nodes from transmitting their own packets without forwarding packets for others, while anonymity prevents nodes from selecting whose packets they forward. Unfortunately, reciprocation could have a side-effect that undermines equal access: it is in each node's interest to work out which packets are likely to be acknowledged and to give them priority, in order to obtain the most reciprocation from its neighbours.

Because packets are anonymised, the only pieces of information on which to base this decision are the packet's incoming and outgoing links. By measuring reliability separately for each outgoing link with respect to each incoming link, a selfish node may be able to find reliable pairs and increase its benefit by giving their packets priority. For example, imagine a node A with four neighbours: B, C, D and E. Neighbours B and C are communicating via A, as are D and E. By measuring the reliability of each pair of links, A can determine that packets from B are more likely to be acknowledged by C than packets from D or E, so it will give B's packets priority on the outgoing link to C.

In one sense this is a useful optimisation: nodes can devote more of their bandwidth to packets that are likely to reach their destinations, which should improve scalability when compared to simple flooding. But this local optimisation could be harmful if it leads to disproportionate resources being allocated to short-range flows.

B.4.2 Anonymous Flows

The second scheme is similar to the first, except that packets are labelled with flow identifiers. Flow identifiers are unique within the scope of a link, so the combination of incoming link and incoming flow identifier uniquely identifies any flow. Each flow's reliability is measured separately for each outgoing link.

Unlike an address, a flow identifier changes at every hop. Flow identifiers do not identify the source or destination of packets, but they enable more intelligent local forwarding by grouping packets that are likely to have similar reliability. This grouping is established by the source when it gives the packets the same flow identifier.

The first packet in a flow to traverse an outgoing link is assigned a new flow identifier, which must be unique within the scope of the outgoing link, independent from the flow identifier on the incoming link, and independent from the flow identifiers on any other outgoing links. Subsequent packets from the same flow will be given the same flow identifier, maintaining the grouping established by the source.

Route Discovery

Every packet is potentially flooded, but because of flow-specific reliability measurement a packet is more likely to be transmitted on some links than others (the reciprocation mechanism ensures that packets with higher estimated reliability are given priority). For a new flow, the reliability of each outgoing link is estimated by that link's overall reliability for new flows with the same incoming link. Once a few packets in the flow have been acknowledged, the estimated reliability of some outgoing links will increase, while others will decay. Anonymous datagrams can be viewed as a special case of anonymous flows, with one packet per flow.

Whereas DSR [138] and similar protocols have explicit route discovery and route maintenance phases, the anonymous flow scheme performs these functions as a side-effect of normal forwarding. Any packet may take any path to the destination, and if an acknowledgement is received then that path is more likely to be selected in the future. This means that an unreliable node (malicious or otherwise) will tend to be routed around when its reliability drops below the reliability of the best alternative route. Rather than defining a reliability threshold below which a route is considered 'faulty', every route is simply used to the extent that it is useful. If a more reliable route exists then the unreliable route will tend to be avoided; if there is no better alternative then the unreliable route will continue to be used.

Robustness

There is no way to verify that packets with the same flow identifier originated from the same source. An attacker along the path can add invalid packets to a flow in order to damage the flow's reliability. However, if the attack

succeeds in causing downstream nodes to drop legitimate packets then the attacker will start to appear unreliable from the point of view of upstream nodes, which will route around it if possible. If the rerouted flow arrives at a downstream node that was previously fooled by the attacker into dropping packets, the rerouted flow's reliability will be measured independently because it is arriving on a different link, and so the new route will not suffer from the old route's damaged reliability. This means it is only necessary to route around the attacker, not around all nodes fooled by the attacker.

If the attacker controls two nodes, one of them on the path and one of them off the path, the on-path node can create packets that can only be acknowledged by the off-path node and add them to a legitimate flow, in an attempt to divert the flow off the path and cause packet loss. For example, if the attacker adds two packets for every legitimate packet, the bogus route will appear 2/3 reliable while the legitimate route will only appear 1/3 reliable. However, as before, if the attack succeeds in causing legitimate packets to be dropped then the attacker will start to appear unreliable to upstream nodes and will be routed around if possible. In both cases, the attacker could achieve the same effect more easily by simply dropping some of the legitimate packets.

An attacker who does not control any nodes on the path cannot affect a flow's reliability by sending invalid packets, because the attacker's packets, arriving on a different link, will be considered to belong to a different flow. However, an attacker adjacent to the path may be able to disrupt a flow by disrupting the nodes along the path, eg by dropping its connection to a node on the path, thus causing bandwidth to be reallocated, or by storing acknowledgements and releasing them in a rush, causing a spike in the attacker's apparent reliability. Thus the anonymous flow scheme falls short of Byzantine robustness [200].

B.4.3 Estimating Reliability

Timeouts allow reliability to be estimated using an exponentially weighted moving average (EWMA): whenever a packet times out the reliability estimate is multiplied by α, and whenever a packet is acknowledged the estimate is multiplied by α and increased by $(1 - \alpha)$, for some decay constant $0 < \alpha < 1$ that determines how much emphasis is given to recent measurements. More sophisticated reliability estimators such as aged EWMAs and Kalman filters [37, 263] are left for future work.

The anonymous datagram scheme requires one estimator per pair of neighbours, plus one estimator per neighbour for the node's own packets. The anonymous flow scheme requires an additional estimator for each flow. However, the amount of space needed to store the state of each EWMA or Kalman filter is very modest.

To combat the flow truncation attack described in Section 5.9, it may be necessary to take the length of flows into account when estimating reliability. This can be done in a reasonably compact way by dividing flow lengths into bins of exponentially increasing size, and keeping a separate moving average for each bin. However, this raises the question of how to combine an estimate that is based on flow length with an estimate based on the history of one particular flow. One possibility would be to take a weighted average of the two estimates, with the weights being determined by the relative accuracy of the estimators for previous packets.

C Anonymity: Attacks and Defences

This section describes several attacks against anonymity and unlinkability in the proposed design. The attacks will require realistic simulations to determine their feasibility and effectiveness. A detailed specification of how these attacks should be tested can only be drawn up when the experiments described in Section 5 are complete and more is known about the dynamics of evidence-based routing.

C.1 Surrounded Node Attack

Threat model: Internal, passive attacker; local eavesdropper.

This attack is possible when an attacker determines that all the currently active neighbours of a node are controlled by the attacker. The surrounded node must therefore be the source of any packets and acknowledgements it sends apart from those it forwards for the attacker. If this occurs simultaneously for the source and destination of any packet or acknowledgement, the attacker can link the communicating parties.

Defence: Nodes should not send or acknowledge packets if only one neighbour is active, and users should ensure that all their neighbouring nodes are owned by different individuals. Nevertheless, colluding neighbours can surround a node.

C.2 Acknowledgement Timing Attack

Threat model: Internal, active attacker.

By measuring the delay between sending a packet and receiving an acknowledgement, an attacker may be able to guess whether its neighbour is the destination of the packet, especially if the attacker can estimate the latency of the neighbour's other links.

Defence: Nodes can make this attack more difficult by delaying their acknowledgements by an amount that corresponds to the round-trip time on a simulated link (eg an exponentially distributed delay in each direction to simulate queueing time plus a constant delay to simulate transmission time). However, the round-trip time measured by the attacker will always increase with the number of hops to the destination, so simulated delays will not fool an attacker who has a good estimate of typical link latencies. Random delays will be even less effective.

Perhaps the most that can be said against this attack is that it mainly threatens the attacker's neighbours, who have chosen to trust the attacker.

C.3 Traffic Confirmation Attack

Threat model: Internal, active attacker.

A relay can test whether a particular user is on the path of a particular flow by sending packets to the user as part of the existing flow instead of creating a new flow. If the user acknowledges the packets, he or she is probably on the path of the existing flow. Similarly, a relay can test whether two flows have the same destination by modifying the flow identifiers of packets to move them from one flow to the other.

Defence: Recipients must not acknowledge packets that do not belong to the flow in which they arrived, even if the packets are valid and come from a trusted sender.

C.4 Intersection Attack Against Flow Identifiers

Threat model: Internal, passive attacker; global eavesdropper.

An attacker can note the lifetimes of the flow identifiers passing through its node, and intersect them with the lifetimes of nodes and links obtained by global eavesdropping. Since flow identifiers change whenever the route changes, the source must be connected to the attacker by a path consisting of nodes and links that have been active for the entire lifetime of the flow identifier. This will allow the attacker to rule out some sources, and in some cases to identify the source with certainty.

Defence: Sources should regularly change the identifiers of long-lived flows, even though this will mean incurring the cost of rediscovering the route. If this attack turns out to be a major threat to unlinkability, it may be necessary to abandon the use of flow identifiers.

C.5 Intersection Attack Against Stable Flows

Threat model: Internal, passive attacker; global eavesdropper.

Even without flow identifiers, an attacker may be able to infer the existence of a stable route if it receives a steady stream of acknowledgements for packets with the same previous and next hops. As before, the attacker can intersect the (approximate) lifetime of the flow with the lifetimes of nodes and links obtained by global eavesdropping, this time to identify the destination as well as the source.

Defence: The source could regularly change the first hop of the route in order to force a new route to be learned, but local adaptation might cause the routes to merge downstream before reaching the attacker. The best defence against this attack seems to be to communicate for short periods.

C.6 Intersection Attack Against Pseudonyms

Threat model: Internal, active attacker; global eavesdropper.

This is the same attack that is used against mix networks. If a user communicates pseudonymously with a global eavesdropper over a period of time, the attacker can intersect the sets of nodes that are active each time the pseudonym is active, eventually identifying the user's node.

Defence: This attack is easier to resist in peer-to-peer networks than in client-server networks, because clients are only active when they are communicating, whereas peers can act as relays even when they have no traffic of their own to send. Users should keep their nodes running when they are not communicating, which will protect their own anonymity and the anonymity of other users by increasing the degree of overlap between the sets of nodes that are active at different times. This will also help to prevent intersection attacks against long-lived flows.

References

[1] K. Aberer, P. Cudre-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, R. Schmidt, and J. Wu. Advanced peer-to-peer networking: The P-Grid system and its applications. *PIK Journal - Praxis der Informationsvevarbeitung und Kommunikation*, Special Issue on P2P Systems, 2003.

[2] T. Ackemann, R. Gold, C. Mascolo, and W. Emmerich. Incentives in peer-to-peer and grid networking. Technical report, Department of Computer Science, University College London, February 2004.

[3] L.A. Adamic. Zipf, power-laws, and Pareto - a ranking tutorial. Available from `http://www.hpl.hp.com/research/idl/papers/ranking/ranking.html`.

[4] E. Adar and B. Huberman. Free riding on Gnutella. *First Monday*, 5(10), October 2000.

[5] A. Adya, W.J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J.R. Douceur, J. Howell, J.R. Lorch, M. Theimer, and R.P. Wattenhofer. FARSITE: Federated, available, and reliable storage for an incompletely trusted environment. In *Proceedings of the 5th USENIX Symposium on Operating Systems Design and Implementation, Boston, MA, USA*, pages 1–14, December 2002.

[6] Aimster website, available from `http://web.archive.org/web/20010801151157/aimster.com/index.phtml`.

[7] R. Albert, H. Jeong, and A.L. Barabási. Error and attack tolerance of complex networks. *Nature*, 406(6794):387–482, 2000.

[8] K.G. Anagnostakis and M.B. Greenwald. Exchange-based incentive mechanisms for peer-to-peer file sharing. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS 2004), Tokyo, Japan, March 2004*, pages 524–533. IEEE Computer Society, 2004.

[9] L. Anderegg and S. Eidenbenz. Ad hoc VCG: A truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. In *ACM Mobicom*, 2003.

[10] R. Anderson. The Eternity Service. In *Proceedings of the 1st International Conference on the Theory and Applications of Cryptology, Prague, Czech Republic*, pages 242–252, October 1996.

[11] Temporary injunction in the anonymous remailer case, September 1996. Anon.penet.fi press release, available from `http://www.eff.org/Privacy/Anonymity/960923_penet_injunction.announce`.

[12] AntsP2P website, `http://antsp2p.sourceforge.net/`.

[13] J. Aspnes and G. Shah. Skip graphs. In *14th Annual ACM-SIAM Symposium on Discrete Algorithms*, January 2003.

[14] I. Avramopoulos, H. Kobayashi, R. Wang, and A. Krishnamurthy. Amendment to: Highly secure and efficient routing. Available from `http://www.cs.princeton.edu/~rywang/papers/infocom04b/amendment.pdf`.

[15] I. Avramopoulos, H. Kobayashi, R. Wang, and A. Krishnamurthy. Highly secure and efficient routing. In *IEEE Infocom, Hong Kong*, March 2004.

[16] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens. An on-demand secure routing protocol resilient to Byzantine failures. In *Proceedings of the ACM Workshop on Wireless Security (WiSe'02), Atlanta, GA, USA*, pages 21–30, September 2002.

[17] R. Axelrod. *The Evolution of Cooperation*. New York: Basic Books, 1984.

[18] Azureus support for distributed tracking. Available from `http://azureus.aelitis.com/wiki/index.php/Distributed_tracker`.

[19] A. Back, U. Moller, and A. Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In *Proceedings of the 4th International Workshop on Information Hiding (IH 2001), Pittsburgh, PA, USA, April 2001*, volume 2137 of *Lecture Notes in Computer Science*, pages 245–257. Springer, 2001.

[20] S. Balazs. Samizdat and the internet - a comparison, 2000. Available from `http://www.slis.ualberta.ca/issues/sbalazs/samizdat.htm`.

[21] N. Bansod, A. Malgi, B. Choi, and J. Mayo. MuON: Epidemic based mutual anonymity. In *13th International Conference on Network Protocols (ICNP 2005), Boston, MA, USA*, November 2005.

[22] A.L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[23] K. Bennett and C. Grothoff. GAP - practical anonymous networking. In R. Dingledine, editor, *Proceedings of the 3rd International Workshop on Privacy Enhancing Technologies (PET 2003), Dresden, Germany, March 2003*, volume 2760 of *Lecture Notes in Computer Science*, pages 141–160. Springer, 2003.

[24] O. Berthold, A. Pfitzmann, and R. Standtke. The disadvantages of free mix routes and how to overcome them. In H. Federrath, editor, *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 2000*, volume 2009 of *Lecture Notes in Computer Science*, pages 30–45. Springer, 2001.

[25] P. Biddle, P. England, M. Peinado, and B. Willman. The darknet and the future of content protection. In *Proceedings of the 2nd International Workshop on Digital Rights Management (DRM 2002), Washington, DC, USA, November 2002*, volume 2696 of *Lecture Notes in Computer Science*, pages 155–176. Springer, 2003.

[26] A. Blanc, Y.K. Liu, and A. Vahdat. Designing incentives for peer-to-peer routing. In *2nd Workshop on Economics of Peer-to-Peer Systems, Cambridge, MA, USA*, June 2004.

[27] B.H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[28] B. Bollobás. *Random Graphs*. Cambridge University Press, 2nd edition, 2001.

[29] D. Bricklin. Friend-to-friend networks, August 2000. Available from `http://www.bricklin.com/f2f.htm`.

[30] J. Broch, D.A. Maltz, D.B. Johnson, Y.C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th International Conference on Mobile Computing and Networking (MobiCom), Dallas, TX, USA*, pages 85–97, October 1998.

[31] D. Brookshier, D. Govoni, and N. Krishnan. *JXTA: Java P2P Programming*. Sams, 2002.

[32] Z. Brown. Cebolla: Pragmatic IP anonymity. In *Ottawa Linux Symposium, Ottawa, Canada*, June 2002.

[33] S. Buchegger and J.Y. Le Boudec. The effect of rumor spreading in reputation systems for mobile ad hoc networks. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt '03), Sophia-Antipolis, France*, March 2003.

[34] S. Buchegger and J.Y. Le Boudec. A robust reputation system for P2P and mobile ad hoc networks. In *2nd Workshop on Economics of Peer-to-Peer Systems, Cambridge, MA, USA*, June 2004.

[35] F.E. Bustamante and Y. Qiao. Friendships that last: Peer lifespan and its role in P2P protocols. In *8th International Workshop on Web Content Caching and Distribution, Hawthorne, NY, USA*, September-October 2003.

[36] L. Buttyán and J.P. Hubaux. Nuglets: A virtual currency to stimulate cooperation in self-organized mobile ad hoc networks. Technical report, Swiss Federal Institute of Technology, January 2001.

[37] L. Capra and M. Musolesi. Autonomic trust prediction for pervasive systems. In *International Workshop on Trusted and Autonomic Computing Systems (TACS-06), Vienna, Austria*, April 2006.

[38] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D.S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *5th Symposium on Operating Systems Design and Implementation, Boston, MA, USA*, December 2002.

[39] M. Castro, P. Druschel, Y.C. Hu, and A. Rowstron. Topology-aware routing in structured peer-to-peer overlay networks. Technical Report MSR-TR-2002-82, Microsoft Research, 2002.

[40] M. Castro and B. Liskov. Practical Byzantine fault tolerance. In *3rd Symposium on Operating Systems Design and Implementation, New Orleans, LA, USA*, February 1999.

[41] J. Cederlöf. Web of trust statistics and pathfinder. Available from `http://www.lysator.liu.se/~jc/wotsap/`.

[42] D.G. Champernowne. A model of income distribution. *Economic Journal*, 63(250):318–351, June 1953.

[43] S. Chandan and C. Hogendorn. Pricing and network externalities in peer-to-peer communications networks. In *29th Research Conference on Communication, Information and Internet Policy, Alexandria, VA, USA*, October 2001.

[44] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2), February 1981.

[45] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[46] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like P2P systems scalable. In *SIGCOMM 2003, Karlsruhe, Germany*, August 2003.

[47] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *7th International Conference on Mobile Computing and Networking (MobiCom)*, July 2001.

[48] B. Chen and R. Morris. L+: Scalable landmark routing and address lookup for multi-hop wireless networks. Technical Report LCS 837, MIT, March 2002.

[49] S. Cheung. An efficient message authentication scheme for link state routing. In *Proceedings of the 13th Annual Computer Security Applications Conference (ACSAC '97), San Diego, CA, USA*, pages 90–98, December 1997.

[50] I. Clarke. Freenet's next generation routing protocol. Available from `http://freenet.sourceforge.net/index.php?page=ngrouting`.

[51] I. Clarke, S.G. Miller, T.W. Hong, O. Sandberg, and B. Wiley. Protecting free expression online with Freenet. *IEEE Internet Computing*, 6(1), January 2002.

[52] I. Clarke and O. Sandberg. Routing in the dark: Scalable searches in dark P2P networks. In *DefCon 13, Las Vegas, NV, USA*, July 2005.

[53] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In H. Federrath, editor, *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 2000*, volume 2009 of *Lecture Notes in Computer Science*, pages 46–66. Springer, 2001.

[54] B. Cohen. Incentives build robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA*, June 2003.

[55] B.F. Cooper and H. Garcia-Molina. Bidding for storage space in a peer-to-peer data preservation system. In *22nd International Conference on Distributed Computing Systems, Vienna, Austria*, July 2002.

[56] L. Cottrell. Mixmaster and remailer attacks. Available from `http://www.obscura.com/~loki/remailer/remailer-essay.html`.

[57] L.P. Cox and B.D. Noble. Samsara: Honor among thieves in peer-to-peer storage. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles, Bolton Landing, NY, USA*, pages 120–132, 2003.

[58] J. Crowcroft, R. Gibbens, F. Kelly, and S. Ostring. Modelling incentives for collaboration in mobile ad hoc networks. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt '03), Sophia-Antipolis, France*, March 2003.

[59] Cypherpunks mailing list, `http://www.cypherpunks.to/list/`.

[60] F. Dabek, M.F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *18th ACM Symposium on Operating Systems Principles, Banff, Canada*, October 2001.

[61] W. Dai. PipeNet 1.1, January 1998. Usenet post, available from `http://www.eskimo.com/~weidai/pipenet.txt`.

[62] W. Dai. A practical attack against ZKS Freedom, May 1999. Usenet post, available from `http://www.eskimo.com/~weidai/freedom-attacks.txt`.

[63] G. Danezis. Mix-networks with restricted routes. In R. Dingledine, editor, *Proceedings of the 3rd International Workshop on Privacy Enhancing Technologies (PET 2003), Dresden, Germany, March 2003*, volume 2760 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2003.

[64] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, USA*, pages 2–15, May 2003.

[65] B. Das and V. Bharghavan. Routing in ad hoc networks using minimum connected dominating sets. In *IEEE International Conference on Communications*, June 1997.

[66] R.M. Dawes. Social dilemmas. *Annual Review of Psychology*, 31:169–193, January 1980.

[67] Z. Despotovic and K. Aberer. Maximum likelihood estimation of peers' performance in P2P networks. In *2nd Workshop on Economics of Peer-to-Peer Systems, Cambridge, MA, USA*, June 2004.

[68] R. Dhamija. A security analysis of Freenet, December 2000. Available from `http://www.sims.berkeley.edu/~rachna/courses/cs261/paper.html`.

[69] C. Díaz and A. Serjantov. Generalising mixes. In R. Dingledine, editor, *Proceedings of the 3rd International Workshop on Privacy Enhancing Technologies (PET 2003), Dresden, Germany, March 2003*, volume 2760 of *Lecture Notes in Computer Science*, pages 18–32. Springer, 2003.

[70] C. Díaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In R. Dingledine and P. Syverson, editors, *Proceedings of the 2nd International Workshop on Privacy Enhancing Technologies (PET 2002), San Francisco, CA, USA, April 2002*, volume 2482 of *Lecture Notes in Computer Science*, pages 54–68. Springer, 2003.

[71] T. Dierks and C. Allen. RFC 2246: The TLS protocol, January 1999.

[72] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

[73] R. Dingledine and N. Mathewson. Anonymity loves company: Usability and the network effect. Draft chapter, available from `http://freehaven.net/doc/wupss04/usability.pdf`.

[74] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *13th USENIX Security Symposium, San Diego, CA, USA*, August 2004.

[75] R. Dingledine, V. Shmatikov, and P. Syverson. Synchronous batching: From cascades to free routes. In *4th International Workshop on Privacy Enhancing Technologies (PET 2004), Toronto, Canada*, May 2004.

[76] P.S. Dodds, R. Muhamad, and D.J. Watts. An experimental study of search in global social networks. *Science*, 301(5634):827–829, August 2003.

[77] S. Dolev and R. Ostrovsky. Xor-trees for efficient anonymous multicast and reception. Technical Report DIMACS 98-54, Rutgers University, 1998.

[78] J.R. Douceur. The Sybil attack. In P. Druschel, F. Kaashoek, and A. Rowstron, editors, *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, USA, March 2002*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer, 2002.

[79] P. Druschel and A. Rowstron. PAST: A large-scale, persistent peer-to-peer storage utility. In *8th Workshop on Hot Topics in Operating Systems, Elmau, Germany*, May 2001.

[80] P. Druschel and A. Rowstron. Pastry: Scalable distributed object location and routing for large-scale peer-to-peer systems. In *18th IFIP/ACM Conference on Distributed Systems Platforms (Middleware 2001), Heidelberg, Germany*, November 2001.

[81] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Proceedings of the 12th Annual International Cryptology Conference (CRYPTO '92), Santa Barbara, CA, USA, August 1992*, volume 740 of *Lecture Notes in Computer Science*, pages 139–147. Springer, 1992.

[82] D. Eastlake and P. Jones. RFC 3174: US secure hash algorithm 1 (SHA1), September 2001.

[83] eDonkey2000 website, `http://overnet.com/`.

[84] M. Ek, F. Hultin, and J. Lindblom. WASTE peer-to-peer protocol, March 2005. Reverse-engineered protocol documentation, available from `http://prdownloads.sourceforge.net/j-waste/waste_documentation-1.1.pdf?download`.

[85] eMule website, `http://www.emule-project.net/`.

[86] J. Feigenbaum and S. Shenker. Distributed algorithmic mechanism design: Recent results and future directions. In *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 1–13, 2002.

[87] M. Feldman and J. Chuang. Hidden-action in multi-hop routing. In *2nd Workshop on Economics of Peer-to-Peer Systems, Cambridge, MA, USA*, June 2004.

[88] M. Félegyházi, J.P. Hubaux, and L. Buttyán. Nash equilibria of packet forwarding strategies in wireless ad hoc networks. To appear in IEEE Transactions on Mobile Computing.

[89] M. Félegyházi, J.P. Hubaux, and L. Buttyán. Equilibrium analysis of packet forwarding strategies in wireless ad hoc networks - the dynamic case. Technical Report IC/2003/68, EPFL, November 2003.

[90] R. Ferreira, M. Ramanathan, A. Awan, A. Grama, and S. Jagannathan. Search with probabilistic guarantees in unstructured peer-to-peer networks. In *5th IEEE International Conference on Peer-to-Peer Computing, Konstanz, Germany*, August-September 2005.

[91] D. Figueiredo, J. Shapiro, and D. Towsley. Incentives to promote availability in peer-to-peer anonymity systems. In *13th International Conference on Network Protocols (ICNP 2005), Boston, MA, USA*, November 2005.

[92] M. Fischer. The consensus problem in unreliable distributed systems (a brief survey). Technical Report YALEU/DCS/RR-273, Yale University, June 1983.

[93] A. Fisk. Dynamic query protocol. Available from `http://www.the-gdf.org/wiki/index.php?title=Dynamic_Query_Protocol`.

[94] M. Franceschetti and R. Meester. Navigation in small world networks, a scale-free continuum model. Technical Report UCB/ERL M03/33, EECS Department, UC Berkeley, 2003.

[95] M. Freedman and R. Vingralek. Efficient peer-to-peer lookup based on a distributed trie. In P. Druschel, F. Kaashoek, and A. Rowstron, editors, *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, USA, March 2002*, volume 2429 of *Lecture Notes in Computer Science*, pages 66–75. Springer, 2002.

[96] M.J. Freedman. A peer-to-peer anonymizing network layer. Master's thesis, Massachusetts Institute of Technology, May 2002.

[97] M.J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *9th ACM Conference on Computer and Communications Security, Washington, DC, USA*, November 2002.

[98] X. Gabaix. Zipf's Law for cities: An explanation. *Quarterly Journal of Economics*, 113(3):739–767, August 1999.

[99] P. Gauthier, B. Bershad, and S.D. Gribble. Dealing with cheaters in anonymous peer-to-peer networks. Technical Report 04-01-03, University of Washington, January 2004.

[100] The Gnutella Developers' Forum, `http://www.the-gdf.org/`.

[101] R. Gibbons. *Game Theory for Applied Economists*. Princeton University Press, 1992.

[102] GNUnet website, `http://gnunet.org/papers.php3`.

[103] The Gnutella protocol specification v0.4. Available from `http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf`.

[104] S. Goel, M. Robson, M. Polte, and E.G. Sirer. Herbivore: A scalable and efficient protocol for anonymous communication. Technical Report 2003-1890, Cornell University, February 2003.

[105] I. Goldberg and D. Wagner. TAZ servers and the rewebber network: Enabling anonymous publishing on the world wide web. *First Monday*, 3(4), April 1998.

[106] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private internet connections. *Communications of the ACM*, 42(2):39–41, February 1999.

[107] P. Golle, K. Leyton-Brown, I. Mironov, and M. Lillibridge. Incentives for sharing in peer-to-peer networks. In *Proceedings of the 2nd International Workshop on Electronic Commerce (WELCOM 2001), Heidelberg, Germany, November 2001*, volume 2232 of *Lecture Notes in Computer Science*, pages 75–86. Springer, 2001.

[108] L. Gong. JXTA: A network programming environment. *IEEE Internet Computing*, 5:88–95, 2001.

[109] L. Gonze. Friendnet, December 2002. Available from `http://www.oreillynet.com/pub/wlg/2428`.

[110] Groove Networks website, `http://www.groove.net/`.

[111] C. Grothoff. An excess-based economic model for resource allocation in peer-to-peer networks. *Wirtschaftsinformatik*, 45(3):285–292, June 2003.

[112] Grouper website, `http://grouper.com/`.

[113] M. Ham and G. Agha. ARA: A robust audit to prevent free-riding in P2P networks. In *5th IEEE International Conference on Peer-to-Peer Computing, Konstanz, Germany*, August-September 2005.

[114] G. Hardin. The tragedy of the commons. *Science*, 162:1243–1248, 1968.

[115] T. Hargreaves. The FastTrack protocol, July 2004. Reverse-engineered protocol documentation, available from `http://cvs.berlios.de/cgi-bin/viewcvs.cgi/*checkout*/gift-fasttrack/giFT-FastTrack/PROTOCOL`.

[116] N.J.A. Harvey, M.B. Jones, S. Saroiu, M. Theimer, and A. Wolman. Skipnet: A scalable overlay network with practical locality properties. In *4th USENIX Symposium on Internet Technologies and Systems, Seattle, WA, USA*, March 2003.

[117] R. Hauser, T. Przygienda, and G. Tsudik. Reducing the cost of security in link-state routing. In *ISOC Symposium on Network and Distributed System Security, San Diego, CA, USA*, February 1997.

[118] S. Hazel and B. Wiley. Achord: A variant of the Chord lookup service for use in censorship resistant peer-to-peer publishing systems. In *1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, USA*, March 2002.

[119] A. Hintz. Fingerprinting websites using traffic analysis. In R. Dingledine and P. Syverson, editors, *Proceedings of the 2nd International Workshop on Privacy Enhancing Technologies (PET 2002), San Francisco, CA, USA, April 2002*, volume 2482 of *Lecture Notes in Computer Science*, pages 171–178. Springer, 2003.

[120] HiveCache website, `http://www.hivecache.com/`.

[121] Know your enemy: Tracking botnets. Technical report, The Honeynet Project and Research Alliance, March 2005. Available from `http://www.honeynet.org/papers/bots/`.

[122] X. Hong, K. Xu, and M. Gerla. Scalable routing protocols for mobile ad hoc networks. *IEEE Network*, 16(4):11–21, July 2002.

[123] M.R. Horton. RFC 850: Standard for interchange of USENET messages, June 1983.

[124] Y.C. Hu, D.B. Johnson, and A. Perrig. SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks. In *4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '02)*, June 2002.

[125] Y.C. Hu, A. Perrig, and D.B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *8th International Conference on Mobile Computing and Networking (MobiCom)*, September 2002.

[126] E. Huang, J. Crowcroft, and I. Wassell. Rethinking incentives for mobile ad hoc networks. In *SIGCOMM 2004 Workshop on Incentives and Game Theory in Networked Systems, Portland, OR, USA*, August-September 2004.

[127] B.A. Huberman and N.S. Glance. Evolutionary games and computer simulations. *Proceedings of the National Academy of Sciences USA*, 90:7716–7718, August 1993.

[128] I2P website, `http://www.i2p.net/`.

[129] A. Iamnitchi, M. Ripeanu, and I. Foster. Small-world file-sharing communities. In *IEEE Infocom, Hong Kong*, March 2004.

[130] IEEE 802.11 Wireless LAN Standards, available from `http://standards.ieee.org/getieee802/802.11.html`.

[131] Invisible IRC Project website, `http://www.invisiblenet.net/iip/docsOfficial.php`.

[132] Clip2.com Inc. Reflector white paper, October 2000. Available from `http://web.archive.org/web/20010804091313/http://dss.clip2.com/reflector_wp.html`.

[133] J. Ioannidis, S. Ioannidis, A.D. Keromytis, and V. Prevelakis. Fileteller: Paying and getting paid for file storage. In *Proceedings of the 6th International Financial Cryptography Conference, Southampton, Bermuda*, pages 282–299, March 2002.

[134] M. Jakobsson. Flash mixing. In *Proceedings of the 18th Annual ACM Symposium on Principles of Distributed Computing, Atlanta, GA, USA*, pages 83–89, May 1999.

[135] R.C. Jeffrey. *The Logic of Decision*. University of Chicago Press, 2nd edition, 1983.

[136] H. Jiang and S. Jin. Exploiting dynamic querying like flooding techniques in unstructured peer-to-peer networks. In *13th International Conference on Network Protocols (ICNP 2005), Boston, MA, USA*, November 2005.

[137] S. Jiang, L. Guo, and X. Zhang. LightFlood: An efficient flooding scheme for file search in unstructured peer-to-peer systems. In *Proceedings of the 32nd International Conference on Parallel Processing (ICPP '03), Kaohsiung, Taiwan*, pages 627–635, October 2003.

[138] D.B. Johnson, D.A. Maltz, and J. Broch. DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In C.E. Perkins, editor, *Ad Hoc Networking*, chapter 5, pages 139–172. Addison-Wesley, 2001.

[139] JXTA website, `http://www.jxta.org/`.

[140] S.D. Kamvar, M.T. Schosser, and H. Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *12th International World Wide Web Conference, Budapest, Hungary*, May 2003.

[141] B. Kantor and P. Lapsley. RFC 977: Network news transfer protocol, February 1986.

[142] B. Karp and H.T. Kung. Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th International Conference on Mobile Computing and Networking (MobiCom), Boston, MA, USA*, pages 243–254, August 2000.

[143] S. Kent, C. Lynn, J. Mikkelson, and K. Seo. Secure border gateway protocol (S-BGP) - real world performance and deployment issues. In *ISOC Symposium on Network and Distributed System Security, San Diego, CA, USA*, February 2000.

[144] D. Kesdogan, J. Egner, and R. Buschkes. Stop-and-go-mixes: Providing probabilistic anonymity in an open system. In *Proceedings of the 2nd International Workshop on Information Hiding (IH '98), Portland, OR, USA, April 1998*, volume 1525 of *Lecture Notes in Computer Science*, pages 83–98. Springer, 1998.

[145] M. Kinateder, R. Terdic, and K. Rothermel. Strong pseudonymous communication for peer-to-peer reputation systems. In *Proceedings of the 20th Annual ACM Symposium on Applied Computing, Santa Fe, NM, USA*, pages 1570–1576, March 2005.

[146] J.M. Kleinberg. Navigation in a small world. *Nature*, 406(6798):845, August 2000.

[147] J.M. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *32nd ACM Symposium on Theory of Computing, Portland, OR, USA*, May 2000.

[148] J.M. Kleinberg. Small-world phenomena and the dynamics of information. *Advances in Neural Information Processing Systems*, 14, 2001.

[149] J.S. Kleinfeld. Could it be a big world after all? The 'six degrees of separation' myth. *Society*, 2002.

[150] Y. Ko and N. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *4th International Conference on Mobile Computing and Networking (MobiCom), Dallas, TX, USA*, October 1998.

[151] Konspire2b website, `http://konspire.sourceforge.net/`.

[152] H. Krawczyk, M. Bellare, and R. Canetti. RFC 2104: HMAC: Keyed-hashing for message authentication, February 1997.

[153] A.Z. Kronfol. *FASD: A Fault-Tolerant, Adaptive, Scalable, Distributed Search Engine*. PhD thesis, Princeton University, May 2002.

[154] D. Kugler. An analysis of GNUnet and the implications for anonymous, censorship-resistant networks. In R. Dingledine, editor, *Proceedings of the 3rd International Workshop on Privacy Enhancing Technologies (PET 2003), Dresden, Germany, March 2003*, volume 2760 of *Lecture Notes in Computer Science*, pages 161–176. Springer, 2003.

[155] S. Kuhn. Prisoner's dilemma. In E.N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. 2003. Available from `http://plato.stanford.edu/archives/fall2003/entries/prisoner-dilemma/`.

[156] Y. Kulbak and D. Bickson. The eMule protocol specification. Technical report, School of Computer Science and Engineering, Hebrew University of Jerusalem, January 2005.

[157] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[158] A. Langley. Mixmaster remailers. In A. Oram, editor, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, chapter 7. O'Reilly, March 2001.

[159] C. Law and K.Y. Siu. Distributed construction of random expander networks. In *IEEE Infocom, San Francisco, CA, USA*, March-April 2003.

[160] J. Ledlie, J. Taylor, L. Serban, and M. Seltzer. Self-organization in peer-to-peer systems. In *10th ACM SIGOPS European Workshop, Saint-Emilion, France*, September 2002.

[161] S.J. Lee, M. Gerla, and C.K. Toh. A simulation study of table-driven and on-demand routing protocols for mobile ad hoc networks. *IEEE Network*, 13(4):48–54, July 1999.

[162] J. Leino. Applications of game theory in ad hoc networks. Master's thesis, Department of Engineering, Physics and Mathematics, Helsinki University of Technology, October 2003.

[163] B. Levine and C. Shields. Hordes: A multicast based protocol for anonymity. *Journal of Computer Security*, 10(3):213–240, 2002.

[164] B.N. Levine, M.K. Reiter, C. Wang, and M. Wright. Timing attacks in low-latency mix systems (extended abstract). In A. Juels, editor, *Proceedings of the 8th International Financial Cryptography Conference (FC 2004), Key West, FL, USA, February 2004*, volume 3110 of *Lecture Notes in Computer Science*, pages 251–265. Springer, 2004.

[165] C. Li, B. Yu, and K. Sycara. An incentive mechanism for message relaying in peer-to-peer discovery. In *2nd Workshop on Economics of Peer-to-Peer Systems, Cambridge, MA, USA*, June 2004.

[166] S.J. Liebowitz and S.E. Margolis. Network externalities (effects). In *In New Palgrave Dictionary of Economics and the Law, MacMillan*, 1998.

[167] Limewire host count, available from `http://www.limewire.com/english/content/netsize.shtml`.

[168] E. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7(2), 2005.

[169] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *16th International Conference on Supercomputing, New York, NY, USA*, June 2002.

[170] Q. Lv, S. Ratnasamy, and S. Shenker. Can heterogeneity make Gnutella scalable? In P. Druschel, F. Kaashoek, and A. Rowstron, editors, *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, USA, March 2002*, volume 2429 of *Lecture Notes in Computer Science*, pages 94–103. Springer, 2002.

[171] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Sustaining cooperation in multi-hop wireless networks. In *2nd Symposium on Networked System Design and Implementation, Boston, MA, USA*, May 2005.

[172] Mobile ad-hoc networks (MANET) charter. Technical report, Internet Engineering Task Force, November 2005. Available from `http://www.ietf.org/html.charters/manet-charter.html`.

[173] G.S. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed hashing in a small world. In *4th USENIX Symposium on Internet Technologies and Systems, Seattle, WA, USA*, March 2003.

[174] S. Marti, T.J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th International Conference on Mobile Computing and Networking (MobiCom), Boston, MA, USA*, pages 255–265, August 2000.

[175] N. Mathewson and R. Dingledine. Practical traffic analysis: Extending and resisting statistical disclosure, May 2004. Available from `http://www.freehaven.net/doc/e2e-traffic/e2e-traffic.pdf`.

[176] P. Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In P. Druschel, F. Kaashoek, and A. Rowstron, editors, *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, USA, March 2002*, volume 2429 of *Lecture Notes in Computer Science*, pages 53–65. Springer, 2002.

[177] M. McLuhan. *Understanding Media: The Extensions of Man*. Routledge, 1964.

[178] Y. Medynskiy and J. Kaye. Characterizing livejournal: SCCs, LiveRank, and six degrees, May 2005. Available from `http://cemcom.infosci.cornell.edu/papers/characterizing-livejournal-medynskiy-kaye-draft03.pdf`.

[179] T. Miconi. I jumped in the GnutellaNet and what did i see? lessons from a simple Gnutella network simulation, October 2002. Available from `http://thomas.miconi.free.fr/TSN3.pdf`.

[180] S. Milgram. The small world problem. *Psychology Today*, 1:61–67, May 1967.

[181] M. Mitzenmacher. A brief history of generative models for power law and lognormal distributions. Available from `http://www.eecs.harvard.edu/~michaelm/NEWWORK/postscripts/lognorm.ps`.

[182] Mnet website, `http://mnet.sourceforge.net/doc.php`.

[183] R. Morselli, B. Bhattacharjee, M.A. Marsh, and A. Srinivasan. Efficient lookup on unstructured topologies. In *24th Annual ACM Symposium on Principles of Distributed Computing, Las Vegas, NV, USA*, July 2005.

[184] A.E. Motter and Y.C. Lai. Cascade-based attacks on complex networks. *Physical Review E*, 66(065102), 2002.

[185] A.E. Motter, T. Nishikawa, and Y.C. Lai. Range-based attacks on links in scale-free networks: Are long-range links responsible for the small-world phenomenon? *Physical Review E*, 66(065103), 2002.

[186] S.J. Murdoch and G. Danezis. Low-cost traffic analysis of Tor. In *Proceedings of the IEEE Symposium on Security and Privacy, Berkeley, CA, USA*, pages 183–195, May 2005.

[187] MUTE website, `http://mute-net.sourceforge.net/`.

[188] M.E.J. Newman, D.J. Watts, and S.H. Strogatz. Random graph models of social networks. In *Proceedings of the National Academy of Sciences USA*, volume 99 (Supplement 1), pages 2566–2572, February 2002.

[189] T.W. Ngan, D.S. Wallach, and P. Druschel. Enforcing fair sharing of peer-to-peer resources. In F. Kaashoek and I. Stoica, editors, *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), Berkeley, CA, USA, February 2003*, volume 2735 of *Lecture Notes in Computer Science*, pages 149–159. Springer, 2003.

[190] S. Nielson, S. Crosby, and D. Wallach. A taxonomy of rational attacks. In M. Castro and R. Renesse, editors, *Proceedings of the 4th International Workshop on Peer-to-Peer Systems (IPTPS '05), Ithaca, NY, USA, February 2005*, volume 3640 of *Lecture Notes in Computer Science*, pages 36–46. Springer, 2005.

[191] C. O'Donnell and V. Vaikuntanathan. Information leak in the Chord lookup protocol. In G. Caronni, N. Weiler, and N. Shahmerhi, editors, *Proceedings of the 4th International Conference on Peer-to-Peer Computing (P2P 2004), Zurich, Switzerland*, pages 28–35. IEEE Computer Society Press, August 2004.

[192] J. Oikarinen and D. Reed. RFC 1459: Internet relay chat protocol, May 1993.

[193] OMNet++ website, `http://www.omnetpp.org/`.

[194] P. Papadimitratos and Z.J. Haas. Secure routing for mobile ad hoc networks. In *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), San Antonio, TX, USA*, January 2002.

[195] P. Papadimitratos and Z.J. Haas. Secure link state routing for mobile ad hoc networks. In *IEEE Workshop on Security and Assurance in Ad Hoc Networks, Orlando, FL, USA*, January 2003.

[196] A. Parker. The true picture of peer-to-peer filesharing, 2005. Commercial presentation, available from `http://www.cachelogic.com/research/slide1.php`.

[197] C. Partridge, D. Cousins, A.W. Jackson, R. Krishnan, T. Saxena, and W.T. Strayer. Using signal processing to analyze wireless data traffic. In *Technical Memorandum No. 1321, BBN Technologies, Cambridge, MA, USA*, May 2002.

[198] W. Peng and X.C. Lu. On the reduction of broadcast redundancy in mobile ad hoc networks. In *Proceedings of the 1st ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Boston, MA, USA*, pages 129–130, August 2000.

[199] C. Perkins, editor. *Ad Hoc Networking*. Addison-Wesley, 2000.

[200] R. Perlman. *Network Layer Protocols with Byzantine Robustness*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, August 1988.

[201] A. Perrig, R. Canneti, J.D. Tygar, and D. Song. The TESLA broadcast authentication protocol. *CryptoBytes*, 5(2):2–13, 2002.

[202] A. Pfitzmann and M. Köhntopp. Anonymity, unobservability, and pseudonymity - a proposal for terminology. In H. Federrath, editor, *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 2000*, volume 2009 of *Lecture Notes in Computer Science*, pages 1–9. Springer, 2001.

[203] A. Pfitzmann and M. Waidner. Networks without user observability. *Computers and Security*, 2(6):158–166, 1987.

[204] C. Plaxton, R. Rajaram, and A. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 311–320, June 1997.

[205] B.C. Popescu, B. Crispo, and A.S. Tanenbaum. Safe and private data sharing with Turtle: Friends team-up and beat the system. In *12th International Workshop on Security Protocols, Cambridge, UK*, April 2004.

[206] J.B. Postel. RFC 821: Simple mail transfer protocol, August 1982.

[207] M.T. Prinkey. An efficient scheme for query processing on peer-to-peer networks, 2001. Available from `http://aeolusres.homestead.com/files/index.html`.

[208] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint relaying: An efficient technique for flooding in mobile wireless networks. Technical Report RR-3898, INRIA, February 2000.

[209] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *SIGCOMM 2001, San Diego, CA, USA*, August 2001.

[210] J.F. Raymond. Traffic analysis: Protocols, attacks, design issues and open problems. In H. Federrath, editor, *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 2000*, volume 2009 of *Lecture Notes in Computer Science*, pages 10–29. Springer, 2001.

[211] M.K. Reiter and A.D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and Systems Security*, 1(1):66–92, 1998.

[212] M. Rennhard and B. Plattner. Introducing MorphMix: Peer-to-peer based anonymous internet usage with collusion detection. In *Proceedings of the Workshop on Privacy in the Electronic Society, in association with the 9th ACM Conference on Computer and Communications Security, Washington, DC, USA*, pages 91–102, November 2002.

[213] M. Rennhard and B. Plattner. Practical anonymity for the masses with MorphMix. In A. Juels, editor, *Proceedings of the 8th International Financial Cryptography Conference (FC 2004), Key West, FL, USA, February 2004*, volume 3110 of *Lecture Notes in Computer Science*, pages 233–250. Springer, 2004.

[214] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwahara. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.

[215] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal*, 6(1), 2002.

[216] J. Risson and T. Moors. Survey of research towards robust peer-to-peer networks: Search methods. Technical Report UNSW-EE-P2P-1-1, University of New South Wales, September 2004.

[217] J. Ritter. Why Gnutella can't scale. no, really, February 2001. Available from `http://www.darkridge.com/~jpr5/doc/gnutella.html`.

[218] G. Roberts and T.N. Sherratt. Development of cooperative relationships through increasing investment. *Nature*, 394(6689):175–179, July 1998.

[219] M. Rogers and S. Bhatti. Unforgeable acknowledgements for unlinkable communication. Technical Report RN/06/05, Department of Computer Science, University College London, March 2006.

[220] C. Rohrs. Query routing for the Gnutella network, May 2002. Available from `http://www.limewire.com/developer/query_routing/keyword%20routing.htm`.

[221] M. Roussopoulos and M. Baker. CUP: Controlled update propagation in peer-to-peer networks. In *USENIX Annual Technical Conference, San Antonio, TX, USA*, June 2003.

[222] M. Roussopoulos and M. Baker. Incentive-based propagation of metadata updates in peer-to-peer networks. *P2P Journal*, pages 1–6, September 2003.

[223] J.H. Saltzer, D.P. Reed, and D.D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, November 1984.

[224] O. Sandberg. Distributed routing in small-world networks, December 2005. Available from `http://www.math.chalmers.se/~ossa/swroute.pdf`.

[225] K. Sanzgiri, B. Dahill, B.N. Levine, C. Shields, and E.M. Belding-Royer. A secure routing protocol for ad hoc networks. In *IEEE International Conference on Network Protocols (ICNP)*, November 2002.

[226] S. Saroiu, P. Krishna Gummadi, and S.D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Multimedia Computing and Networking (MMCN '02)*, January 2002.

[227] N. Sarshar, P. Boykin, and V. Roychowdhury. Percolation search in power law networks: Making unstructured peer-to-peer networks scalable. In G. Caronni, N. Weiler, and N. Shahmerhi, editors, *Proceedings of the 4th International Conference on Peer-to-Peer Computing (P2P 2004), Zurich, Switzerland*, pages 2–9. IEEE Computer Society Press, August 2004.

[228] N. Sarshar and V. Roychowdhury. Scale-free and stable structures in complex ad hoc networks. *Physical Review E*, 69(026101), 2004.

[229] L.J. Savage. *The Foundations of Statistics*. Wiley, 1954.

[230] V. Scarlata, B.N. Levine, and C. Shields. Responder anonymity and anonymous peer-to-peer file sharing. In *Proceedings of the 9th International Conference on Network Protocols (ICNP '01), Riverside, CA, USA*, pages 272–280, November 2001.

[231] A. Serjantov. Anonymizing censorship resistant systems. In P. Druschel, F. Kaashoek, and A. Rowstron, editors, *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, USA, March 2002*, volume 2429 of *Lecture Notes in Computer Science*, pages 111–120. Springer, 2002.

[232] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In R. Dingledine and P. Syverson, editors, *Proceedings of the 2nd International Workshop on Privacy Enhancing Technologies (PET 2002), San Francisco, CA, USA, April 2002*, volume 2482 of *Lecture Notes in Computer Science*, pages 41–53. Springer, 2003.

[233] A. Serjantov, R. Dingledine, and P. Syverson. From a trickle to a flood: Active attacks on several mix types. In F.A.P. Petitcolas, editor, *Proceedings of the 5th International Workshop on Information Hiding (IH 2002), Noordwijkerhout, The Netherlands, October 2002*, volume 2578 of *Lecture Notes in Computer Science*, pages 36–52. Springer, 2002.

[234] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, November 1979.

[235] R. Sherwood, B. Bhattacharjee, and A. Srinivasan. P5: A protocol for scalable anonymous communication. In *IEEE Symposium on Security and Privacy, Berkeley, CA, USA*, May 2002.

[236] Shinkuro website, `http://shinkuro.com/`.

[237] J. Shneidman and D.C. Parkes. Rationality and self-interest in peer to peer networks. In F. Kaashoek and I. Stoica, editors, *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), Berkeley, CA, USA, February 2003*, volume 2735 of *Lecture Notes in Computer Science*, pages 139–148. Springer, 2003.

[238] M.V. Simkin and V.P. Roychowdhury. Copied citations create renowned papers? *Annals of Improbable Research*, 11(1):24–27, 2005.

[239] E.G. Sirer, M. Polte, and M. Robson. CliqueNet: A self-organizing, scalable, peer-to-peer anonymous communication substrate, December 2001. Available from `http://www.cs.cornell.edu/People/egs/papers/cliquenet-iptp.pdf`.

[240] B. Smith, S. Murthy, and J.J. Garcia-Luna-Aceves. Securing distance-vector routing protocols. In *ISOC Symposium on Network and Distributed System Security, San Diego, CA, USA*, February 1997.

[241] J. Maynard Smith. *Evolution and the Theory of Games*. Cambridge University Press, 1982.

[242] A. Solow. Power laws without complexity. *Ecology Letters*, 8(4):361–363, April 2005.

[243] D.X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, USA*, pages 44–55, May 2000.

[244] K. Sripanidkulchai. The popularity of Gnutella queries and its implications on scalability, February 2001. Available from `http://www.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html`.

[245] M. Stiegler. An introduction to petname systems, February 2005. Available from `http://www.skyhunter.com/marcs/petnames/IntroPetNames.html`.

[246] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM 2001, San Diego, CA, USA*, August 2001.

[247] T. Strufe and D. Reschke. Efficient content distribution in semi-decentralized peer-to-peer networks. In *Proceedings of the 8th International Netties Conference, Ilmenau, Germany*, pages 33–38, September-October 2002.

[248] D. Stutzbach, R. Rejaie, and S. Sen. Characterizing unstructured overlay topologies in modern P2P file-sharing systems. In *Internet Measurement Conference, Berkeley, CA, USA*, October 2005.

[249] Q. Sun and H. Garcia-Molina. SLIC: A selfish link-based incentive mechanism for unstructured peer-to-peer networks. In *24th International Conference on Distributed Computing Systems*, 2004.

[250] K. Tamilmani, V. Pai, and A.E. Mohr. SWIFT: A system with incentives for trading. In *2nd Workshop on Economics of Peer-to-Peer Systems, Cambridge, MA, USA*, June 2004.

[251] J. Telesin. Inside 'samizdat'. *Encounter*, 40(2):25–33, February 1973.

[252] D. Tsoumakos and N. Roussopolous. Probabilistic knowledge discovery and management for P2P networks. *P2P Journal*, pages 15–20, November 2003.

[253] J. Udell, N. Asthagiri, and W. Tuvell. Security. In A. Oram, editor, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, chapter 18. O'Reilly, March 2001. This chapter describes Groove (`http://www.groove.net/`).

[254] A. Urpi, M.A. Bonuccelli, and S. Giordano. Modelling cooperation in mobile ad hoc networks: a formal description of selfishness. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt '03), Sophia-Antipolis, France*, March 2003.

[255] V. Vishnumurthy, S. Chandrakumar, and E.G. Sirer. KARMA: A secure economic framework for p2p resource sharing. In *Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA*, June 2003.

[256] L.M. Wahl and M.A. Nowak. The continuous prisoner's dilemma: I. linear reactive strategies. *Journal of Theoretical Biology*, 200:307–321, 1999.

[257] M. Waldman, L.F. Cranor, and A. Rubin. Publius. In A. Oram, editor, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, chapter 11. O'Reilly, March 2001.

[258] M. Waldman and D. Mazières. Tangler: A censorship-resistant publishing system based on document entanglements. In *Proceedings of the 8th ACM Conference on Computer and Communications Security, Philadelphia, PA, USA*, pages 126–135, November 2001.

[259] M. Waldman, A. Rubin, and L.F. Cranor. Publius: A robust, tamper-evident, censorship-resistant web publishing system. In *Proceedings of the 9th USENIX Security Symposium*, pages 59–72, August 2000.

[260] WASTE website, `http://waste.sourceforge.net/`.

[261] D.J. Watts. *Small Worlds: The Dynamics of Networks Between Order and Randomness*. Princeton University Press, 1999.

[262] D.J. Watts and S.H. Strogatz. Collective dynamics of small world networks. *Nature*, 393(6684):440–442, June 1998.

[263] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Conference on Embedded Networked Sensor Systems (SenSys '03), Los Angeles, CA, USA*, November 2003.

[264] M. Wright, M. Adler, B. Levine, and C. Shields. An analysis of the degradation of anonymous protocols. In *ISOC Symposium on Network and Distributed System Security, San Diego, CA, USA*, February 2002.

[265] J. Wu and H. Li. On calculating connected dominating sets for efficient routing in ad hoc wireless networks. In *3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, Seattle, WA, USA*, August 1999.

[266] Y. Xue, B. Li, and K. Nahrstedt. A scalable location management scheme in mobile ad hoc networks. In *Proceedings of the 26th Annual IEEE Conference on Local Computer Networks, Washington, DC, USA*, page 102, 2001.

[267] Y. Xue, B. Li, and K. Nahrstedt. Price-based resource allocation in wireless ad hoc networks. In *Proceedings of the 11th International Workshop on Quality of Service (IWQoS 2003), Berkeley, CA, USA, June 2003*, volume 2707 of *Lecture Notes in Computer Science*, pages 79–96. Springer, 2003.

[268] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer networks. In *22nd International Conference on Distributed Computing Systems, Vienna, Austria*, July 2002.

[269] B. Yang and H. Garcia-Molina. Designing a super-peer network. In *IEEE International Conference on Data Engineering*, March 2003.

[270] P. Zakharov. Thermodynamic approach for community discovering within the complex networks: LiveJournal study, February 2006. arXiv preprint, available from `http://arxiv.org/abs/physics/0602063`.

[271] M.G. Zapata. Secure ad hoc on demand distance vector (SAODV) routing, October 2001. Available from `ftp://manet.itd.nrl.navy.mil/pub/manet/2001-10.mail`.

[272] M.G. Zapata and N. Asokan. Securing ad hoc routing protocols. In *Proceedings of the ACM Workshop on Wireless Security (WiSe'02), Atlanta, GA, USA*, pages 1–10, September 2002.

[273] Zeroconf website, `http://www.zeroconf.org/`.

[274] H. Zhang, A. Goel, and R. Govindan. Using the small-world model to improve Freenet performance. In *IEEE Infocom, New York, NY, USA*, June 2002.

[275] Y. Zhang and V. Paxson. Detecting stepping stones. In *Proceedings of the 9th USENIX Security Symposium*, pages 171–184, August 2000.

[276] B.Y. Zhao, J.D. Kubiatowicz, and A.D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, EECS Department, UC Berkeley, April 2001.

[277] L. Zhou and Z.J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, November 1999.