An Adaptive Routing Protocol for Censorship-Resistant Communication

Michael Rogers
University College London, UK
m.rogers@cs.ucl.ac.uk

Saleem Bhatti
University of St Andrews, UK
saleem@cs.st-andrews.ac.uk

Abstract

In open-membership networks such as peer-to-peer overlays and mobile *ad hoc* networks, messages must be routed across an unknown and changing topology where it may not be possible to establish the identities or trustworthiness of all the nodes involved in routing. This paper describes a decentralised, adaptive routing protocol in which nodes use feedback in the form of unforgeable acknowledgements (U-ACKs) to discover dependable routes without knowing the identities of the endpoints or the structure of the network beyond their immediate neighbours. Our protocol is designed to survive faulty or misbehaving nodes and reveal minimal information about the communicating parties, making it suitable for use in censorship-resistant communication.

1 INTRODUCTION

Internet pioneer John Gilmore famously claimed that ”The Net interprets censorship as damage and routes around it” [13]. Unfortunately, at a technical level this statement is becoming less accurate every year: censorship of the internet is becoming more widespread and sophisticated, and dozens of governments now filter the information available to their citizens [21]. However, at a social level Gilmore’s statement remains a maxim for activists and researchers working on censorship-resistant communication.

In this paper we examine the problem of routing messages across an untrusted, open-membership network, where a node cannot establish the identities or trustworthiness of any nodes other than its immediate neighbours. Each communication exchange involves a series of messages sent from an *originator* to a *destination* by relying on the forwarding behaviour of intermediate *relays*. We assume that any node in the network can function as an originator, destination or relay. Examples of such networks include peer-to-peer overlays and mobile *ad hoc* networks.

In an untrusted network, messages may be lost, reordered, or modified for any number of reasons, and it may not be possible to determine whether such events are accidental or deliberate in nature. Rather than trying to identify the node or link responsible for each failure, we take the pragmatic approach of measuring dependability without attempting to distinguish between deliberate and accidental failures. We show that by observing end-to-end *unforgeable acknowledgements* (U-ACKs), relays can adaptively discover dependable routes without knowing the origins or destinations of the messages and acknowledgements they forward. Lightweight *flow identifiers* can be used to improve the efficiency of adaptive routing.

The next section discusses previous work in this area. Section 3 gives an overview of our adaptive routing protocol. In Section 4 we describe simulations to evaluate the protocol’s efficiency and scalability. Section 5 discusses the results of the simulations and considers possible applications of our protocol. We conclude the paper with some ideas for future work.

2 BACKGROUND AND RELATED WORK

Many routing protocols use feedback from the destination to guide forwarding decisions at relay nodes. This adaptive approach has the advantage of propagating information about the state of the network only to those nodes to which the information is relevant; however, the absence of information about unused routes can make the cost of initial route discovery relatively high.

Q-routing [4] uses reinforcement learning to find the quickest route to a destination. Each node updates its estimate of the time it will take a message to reach the destination, including time spent in the node’s own queue, based on

immediate feedback in the form of the next node's estimate of the delivery time. Information is thus passed back from the destination towards the source, taking into account congestion and any other causes of delay.

AntNet [10] and AntHocNet [11] are routing protocols inspired by the collective foraging behaviour of ants, which use chemical markers to discover short paths between food sources and their nest. AntNet uses routing messages known as forward and backward ants. Each node periodically dispatches a forward ant to a destination chosen probabilistically from its routing table. The ant records the time at which it enters and leaves each node. When it reaches its destination it returns to the source as a backward ant; information gathered on the forward journey about link states and latencies is left at each node along the path, where it is incorporated probabilistically into the routing table.

In both AntNet and Q-routing, nodes base their routing decisions on information about network paths that is supplied by other nodes. This approach is unsuitable for untrusted networks, where path information could be corrupted by malicious or faulty nodes.

A different kind of ant-inspired routing is used in the MUTE file sharing network [20]. MUTE is a peer-to-peer overlay in which each node adopts a random overlay address for sending and receiving messages. Messages are routed across the overlay using a probabilistic reverse-path forwarding protocol: every message carries the overlay addresses of its originator and destination, allowing relays to learn the reverse path to the originator before forwarding the message towards the destination. If no path to the destination is known, the message is broadcast; if several paths are known, a path is chosen at random with probability proportional to the number of messages received from the destination along that path. Through this simple process of local adaptation, it is possible to discover routes across the network without knowing its membership or structure. However, this form of reverse-path forwarding is vulnerable to address spoofing: an attacker could divert messages addressed to a victim by sending messages using the victim's overlay address.

Address spoofing raises an issue that is central to routing protocols: identity. Most routing protocols use a globally unique name or address to identify each node; it is usually assumed that accidental address collisions can be resolved (for example by choosing a new address when a collision is detected [6]) and deliberate collisions can be prevented (for example through the use of signed route advertisements [23, 18], or by using certified identities for all nodes [1, 2]). However, these assumptions may not hold in an untrusted network without a centralised public key infrastructure. Routing protocols for such networks must therefore be designed to cope with attackers who may deliberately use the identities of other nodes [7, 8], use more than one identity at once [12], or change identities to escape the consequences of past behaviour [15]. Open-membership networks must also cope with the more mundane aspects of identity management, such as scalability and churn: it may not be practical for every node to be aware of the structure and membership of a constantly changing network, even if no attack is taking place.

Open networks also present new challenges to privacy: anyone who can participate in a network can gather information about the activities of other users. Even if all traffic is encrypted, traffic analysis can reveal a great deal of information about who communicates with whom, when, and for how long [9, 22, 17]. We believe that protocols for open networks should aim to protect the privacy of users by minimising the information revealed to eavesdroppers, so our adaptive routing protocol is designed to preserve *unlinkability* between originators and destinations [24].

3 ADAPTIVE ROUTING IN THE DARK

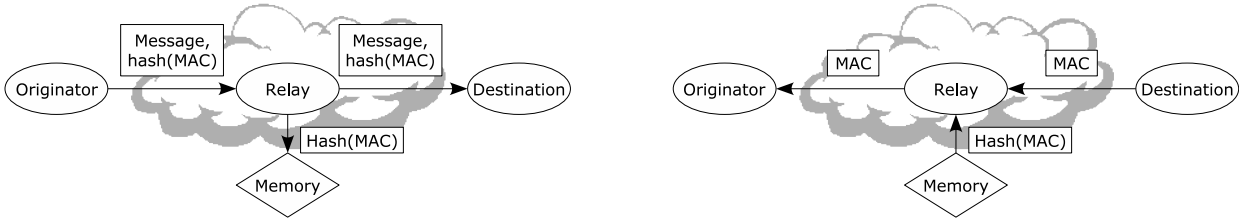
Because of the difficulties surrounding identity and privacy in open-membership networks, we are interested in the question of whether routing can operate succesfully when nodes have only minimal knowledge of the network. In this section we describe an adaptive routing protocol that uses end-to-end feedback to guide routing decisions without identifying the endpoints to the relays.

3.1 Unforgeable acknowledgements

Our adaptive routing protocol uses end-to-end (originator to destination) *unforgeable acknowledgements* (U-ACKs) that can be verified by relays without establishing a security association with either of the endpoints. Unlike a digital signature scheme, relays do not need to share any keys with the originator or destination, or to know their identities.

A full description of the U-ACK mechanism can be found in [14]. Unforgeable acknowledgements make use of two standard cryptographic primitives: *message authentication codes* (MACs) and *collision-resistant hashing*. Before transmitting a message, the originator computes a MAC over the message using a secret key shared with the destination. (Any standard key agreement mechanism appropriate to the application can be used to establish the shared key.)

Figure 1: The U-ACK protocol: relays use the hash attached to the message to verify the acknowledgement.



Instead of attaching the MAC to the message, the originator attaches the *hash of the MAC* to the message. Relays store a copy of the hash when they forward the message. If the message reaches its destination, the destination computes a MAC over the received message using the secret key shared with the originator. If the hash of this MAC matches the hash received with the message, then the destination has validated the message, and it sends the *MAC as an acknowledgement*. The acknowledgement is forwarded back along the path taken by the message. Relays can verify that the acknowledgement hashes to the same value that was attached to the message, but they cannot forge acknowledgements for undelivered messages – they lack the secret key to compute the correct MAC, and the hash function is collision resistant. Thus a U-ACK proves to the originator and relays that the message was delivered unmodified to its intended destination, without revealing the destination's identity to the relays.

In the above discussion we have assumed that all links between neighbours are bidirectional – if a message can be sent in one direction, an acknowledgement can be sent in the opposite direction. For the purposes of our protocol, nodes that are only connected by unidirectional links are not considered to be neighbours, and the protocol cannot discover routes that contain unidirectional links. This issue is discussed in more detail in [14].

3.2 Local adaptation

Nodes in our protocol require only minimal knowledge of the network; in fact we assume that each node knows nothing about the network beyond its immediate neighbours. Each node must be able to identify its neighbours for the purposes of forwarding, but these identities need not be cryptographically verifiable, and a node is free to use a different identity when dealing with each neighbour. *Our protocol does not use end-to-end addresses.* U-ACKs allow relays to discover which messages have reached their destinations without identifying those destinations; using this information, nodes can attempt to learn which messages should be forwarded to which neighbours in order to maximise the number of messages delivered. As with Q-routing, AntNet and MUTE, this process of local adaptation can lead to globally efficient routing.

We use the following general approach for discovering dependable routes:

- Each node keeps a small pool of messages that are waiting to be sent
- For each message in the pool and each potential next hop, the node estimates the probability that an acknowledgement will be received if the message is sent to the next hop
- The message and next hop with the highest probability are chosen
- The next hop is removed from the message's list of potential next hops, and the message is sent to the next hop
- Information about the message and the next hop is recorded in the message table (see Section 3.9)
- When the message is acknowledged or times out (see Section 3.7), the information in the message table is used to update the node's dependability estimators

The size of the message pool is limited; in the simulations described in Section 4, the pool can hold five messages. Messages that have been sent to all potential next hops are removed from the pool. When a new message is added to the pool and the pool is already full, the message with the lowest remaining probability of being acknowledged (which may be the new message) is discarded.

3.3 Dependability estimators

Within the general framework described above there are many possible ways to evaluate a message's dependability, but to achieve the best results, any information that may indicate the message's relationship to earlier messages should be taken into consideration. Even without end-to-end addresses, the identities of the previous and next hops provide some information that can be used to distinguish between messages. Timing is also significant: changes in the network topology and traffic levels, even if they are not directly visible to the node, make new information more relevant than old information when estimating dependability. Thus our first attempt at a dependability estimator is a simple exponentially weighted moving average for each pair of neighbours (previous hop and next hop). The moving average is adjusted upwards whenever a message is acknowledged (equation 1), and downwards whenever a message times out (equation 2):

$$x_{i+1} = \alpha x_i + (1-\alpha) \quad (1)$$

$$x_{i+1} = \alpha x_i \quad (2)$$

where x_i is the estimate before updating the moving average and x_{i+1} is the estimate afterwards. The parameter α determines the sensitivity of the estimator; in our simulations the value of α was 0.9. We will see in Section 5 that even this simple per-pair estimator provides a considerable improvement in efficiency when compared with flooding; further improvements may be possible by designing more sophisticated estimators.

3.4 Flow identifiers

In addition to discovering implicit relationships between messages, the efficiency of adaptive routing can be improved if related messages are explicitly grouped together. We define a *flow* as any sequence of messages that have the same origin and destination and that are semantically related in some way, such as the sequence of messages that make up a single file transfer. To indicate the existence of a flow, the originator marks all messages in the flow with an arbitrary *flow identifier*. The contents of the flow identifier are not significant – it is just a label, and it is not covered by the message authentication code. All messages in a flow are marked with the same flow identifier.

Flow identifiers have local scope: as a flow travels across the network, it may be assigned a different identifier on each link it traverses. However, messages belonging to the same flow should have matching identifiers on any given link. Each flow traversing a link must be assigned an identifier that distinguishes it from any other flows currently traversing the same link; in particular, flows arriving at a node from different upstream neighbours must be assigned distinct identifiers on any downstream link, even if they happen to have matching identifiers on their respective upstream links.

The use of flow identifiers with local scope is similar to the use of label-swapping in virtual circuits or multi-protocol label switching, but there is no requirement to establish state in the relays before data transfer begins – identifiers can be assigned to new flows on the fly.

Although they do not identify the endpoints, flow identifiers enable fine-grained dependability measurement: messages arriving from the same previous hop with the same flow identifier are likely to have the same (unknown) origin and destination, so the dependability of earlier messages in the flow can be used to estimate the dependability of later messages.

Nodes can make use of this information by keeping a separate dependability estimator for each active flow. As with the simple per-pair estimators described above, new information is more likely to be relevant than old information, so an exponentially weighted moving average is again appropriate.

To estimate the dependability of new flows, nodes also keep per-pair estimators that are only updated by the first message in each flow. This provides an estimate of the dependability of a message *given that it is the first message in a new flow* – a per-pair estimator updated by every message would tend to overestimate the dependability of new flows. The per-pair estimators are used to initialise per-flow estimators, which are thereafter updated independently.

3.5 Locally generated messages

In the preceding discussion we assumed that every message has a previous hop, but in fact any node may originate messages as well as forwarding them.

It would be inefficient to add every local message to the pool before the first copy is sent, so nodes keep a queue of messages for each locally generated flow (using a separate queue for each flow prevents head-of-line blocking). When

choosing a message and a next hop, the node considers the first message in each local queue as well as the messages in the pool. If a local message is chosen, it is removed from the queue and added to the pool, since additional copies may later be sent to other neighbours.

The dependability estimators for locally generated messages are similar to those described above for forwarded messages.

3.6 Duplicate detection

Duplicate messages should be detected and discarded to prevent routing loops and reduce redundancy. However, before discarding a duplicate message, the previous hop is added to the corresponding record in the message table (see Section 3.9). If a U-ACK for the message is received, a copy of the U-ACK is returned to every previous hop listed in the record. This proves to all the previous hops that the message was delivered, providing a relatively lightweight way for nodes to maintain information about alternative routes in case the existing route fails.

3.7 Timeouts

Information about outstanding messages cannot be stored indefinitely, and it is important to update dependability estimators in a timely fashion. Therefore a node must at some point conclude that an outstanding message is not going to be acknowledged, decrease the dependability estimator, and remove the corresponding record from the message table.

A relay that receives an acknowledgement after discarding the corresponding record cannot verify or forward the acknowledgement, so there is no reason for a relay to keep records for longer than its upstream or downstream neighbours. Fixed timeouts are a simple way to ensure that adjacent relays discard their records at approximately the same time, minimising wasted storage; the choice of an appropriate timeout is discussed in [14].

3.8 Aging and discounting

From the description given in Section 3.2 it might appear that adaptive routing is likely to produce a large number of redundant messages. Aging and discounting are two techniques designed to improve the efficiency of routing by reducing the likelihood of sending redundant messages.

The technique of *aging* is based on the observation that an acknowledgement arriving after the timeout will not be recognised, since the corresponding record will have been discarded. Similarly, an acknowledgement arriving near the timeout is likely to miss the timeout at the next node. Thus the probability of an acknowledgement reaching the originator decreases for as long as the message is held in the pool, reaching zero at the timeout. The dependability of each message in the pool should ideally be aged using the expected arrival time of the acknowledgement, but that would require relays to keep an estimate of the round-trip time for each flow. A simpler alternative is to use the current time, for example by decreasing a message's dependability linearly from the time it enters the pool to the time it expires.

The second technique, *discounting*, is based on the observation that each additional copy of a message that is sent is increasingly likely to be redundant. The higher the dependability of the copies sent so far, the more likely it is that an additional copy will be redundant, so an additional copy should be sent if and only if the dependability of the copies sent so far is low. (This is also desirable for the network as a whole, because it will lead to more exploration on new or damaged routes, and less exploration on well-established routes.)

Ideally we would like to calculate the conditional probability of an additional copy of the message being acknowledged, given that none of the previous copies is acknowledged first. However, it would be impractical to store all the information needed to estimate the conditional probability for each neighbour given any possible combination of previous neighbours, so in practice it is necessary to treat the probabilities as independent.

Let x_i denote the probability of the i^{th} copy of the message being acknowledged, and let y_i denote the conditional probability of the i^{th} copy being acknowledged, given that none of the previous copies is acknowledged first. Then, under the simplifying assumption of independence:

$$\begin{aligned} y_1 &= x_1 \\ y_i &= (1 - \sum_{j=1}^{i-1} y_j) x_i \end{aligned} \tag{3}$$

where x_i is the estimate before discounting and y_i is the discounted estimate. As the sum of the estimates for previous copies approaches one, the estimate for an additional copy will approach zero. The calculation can be made more efficient by keeping a running total.

3.9 Storage overhead

Our protocol has modest bandwidth and computation overheads: each acknowledgement is the size of a message authentication code (typically around 20 bytes), and only one hash computation is required to verify an acknowledgement. However, the storage overhead may be more significant. We offer some rough calculations below; the exact figures will depend on implementation decisions such as the choice of hash function, and application characteristics such as the link speed and message size.

Nodes store information about outstanding messages in the *message table*. Each record includes the hash of the expected acknowledgement, the previous hops of all received copies of the message, the next hops of all sent copies, and pointers to any dependability estimators that will need to be updated when the message is acknowledged or times out.

The size of a node's message table depends primarily on its outgoing bandwidth. If we assume a timeout of 60 seconds and a typical message size of 1000 bytes, a node may have up to 60 messages outstanding for every kB/s of outgoing bandwidth. If each record occupies 100 bytes, a typical peer-to-peer node with 32 kB/s of outgoing bandwidth would need to allocate 192 kB of storage for its message table.

Flow identifiers introduce a second source of storage overhead: the *flow table*. For each flow a node is currently forwarding, the node must record the mapping between the flow identifier on the incoming link and the flow identifier on the outgoing link, together with a per-flow dependability estimator. The number of active mappings is limited by the node's outgoing bandwidth, since each outgoing message reactivates one mapping. If we assume that mappings are discarded after 60 seconds of inactivity and each mapping occupies 100 bytes, the node described above would need a further 192 kB of storage for its flow table.

All the information in the flow table is soft state: it does not need to survive across restarts, and information about inactive flows can be discarded to reclaim space. When information about a flow is discarded, the flow is not cut off, but the route will need to be rediscovered if the flow later becomes active again.

4 SIMULATIONS

In this section we describe simulations designed to test the feasibility of adaptive routing without end-to-end addresses. The experiments were conducted using a discrete event-based simulator written in Java; the simulation code is available from the authors on request.

Each data point was based on five independent runs of the simulator with different random seeds – the error bars in the figures show the maximum and minimum values obtained in any run. Each run lasted for two hours of simulated time, and the measurements were taken over the course of the second hour.

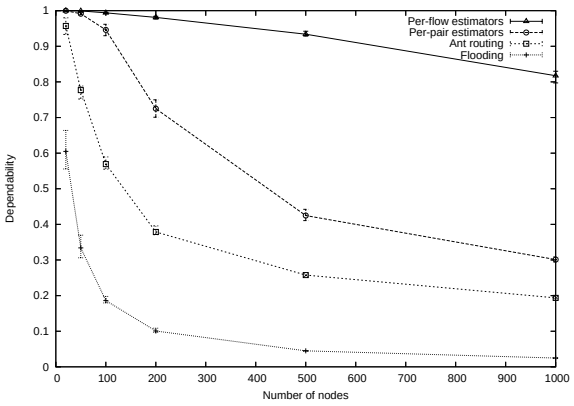
The network topologies were classical Erdös-Rényi random graphs with an average degree of 10; we obtained similar results for scale-free graphs. The number of nodes was varied from 20 to 1000.

Throughout each run, churn was simulated by removing nodes from the overlay at random and replacing them with new nodes. Node lifetimes were exponentially distributed with a mean of two hours.

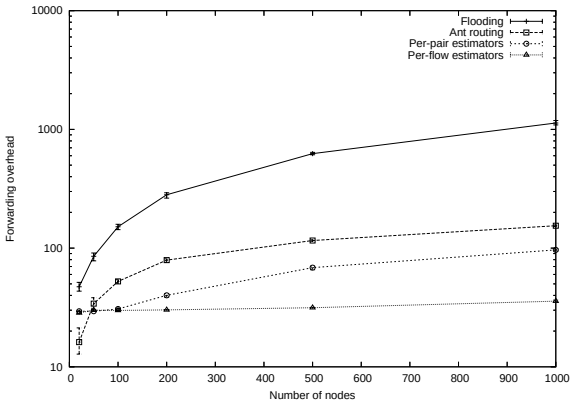
Messages were sent from randomly chosen originators to randomly chosen destinations; no node was the destination of its own messages. Messages were organised into flows of exponentially distributed length, with an average of 1000 messages per flow (the effect of varying the flow length is examined in Section 4.3). We obtained similar results for flows with constant and exponentially distributed inter-message delays; the results presented here are for constant delays. All messages were 1000 bytes in size and acknowledgements were 50 bytes. Each node had 32 kB/s of outgoing bandwidth and unlimited incoming bandwidth – these values are meant to represent the approximate capacity of nodes in current peer-to-peer networks. Each node was the originator of one flow at a time on average, for an average offered load of 1 kB/s per node.

In each run we measured the *dependability*, defined as the fraction of messages that were successfully delivered, and the *forwarding overhead*, defined as the number of messages forwarded divided by the number of messages successfully delivered.

Figure 2: Comparing the scalability of adaptive routing, ant routing, and flooding.



(a) Dependability as a function of the network size.



(b) Forwarding overhead as a function of the network size (note the logarithmic scale on the vertical axis).

4.1 Scalability

The first experiment compared the scalability of adaptive routing to two existing protocols: flooding and ant routing. We chose these protocols for comparison because, unlike most other routing protocols but in common with our protocol, they only require local knowledge of the network. The flooding implementation used a drop-tail queue with a capacity of 20 messages. The ant routing implementation was similar to that used by MUTE, as described in Section 2, with a 20-message drop-tail queue. Varying the queue size did not have a significant impact on the results.

Two variants of adaptive routing were tested. The first variant, per-pair estimators, maintained one dependability estimator per pair of neighbours. The second variant, per-flow estimators, used flow identifiers as described in Section 3.4 and maintained one dependability estimator per flow, plus one estimator per pair of neighbours to initialise the estimators for new flows. Both variants selected the message and next hop with the highest probability of receiving an acknowledgement. The pool had a capacity of five messages; varying the pool size did not have a significant impact on the results.

The results of the scalability experiment are show in Figures 3(a) and 3(b). Flooding cannot support dependable communication even in small networks, due to the large number of redundant messages it produces. Ant routing works well in small networks but does not scale. When compared with flooding, per-pair estimators clearly improve dependability and reduce forwarding overhead at all network sizes. Per-flow estimators perform even better, with the result that adaptive routing in a 1000-node network has higher dependability and lower overhead than flooding in a 20-node network. However, even with per-flow estimators, dependability is only 82% in a network of 1000 nodes, suggesting that adaptive routing may not scale to very large networks.

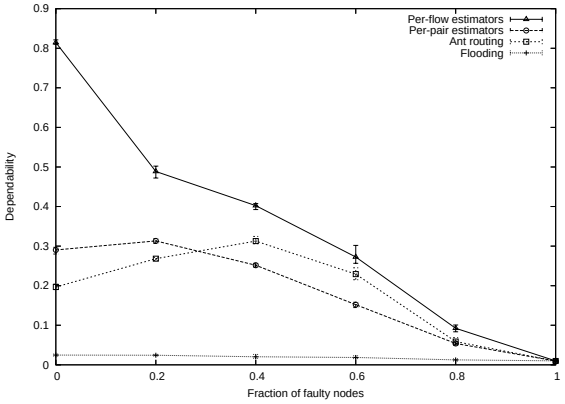
4.2 Resilience to faulty nodes

The second experiment examined the impact of faulty nodes, which do not forward messages for other nodes, but continue to place load on the network by originating and acknowledging messages. Routes passing through faulty nodes are unusable, but the neighbours of a faulty node cannot trivially detect that it is faulty, because it will still acknowledge messages for which it is the destination; a relay does not know whether a node that returns acknowledgements is the destination or just another relay. To achieve our goal of censorship resistance, it is important to know how our protocol is affected by these faulty nodes.

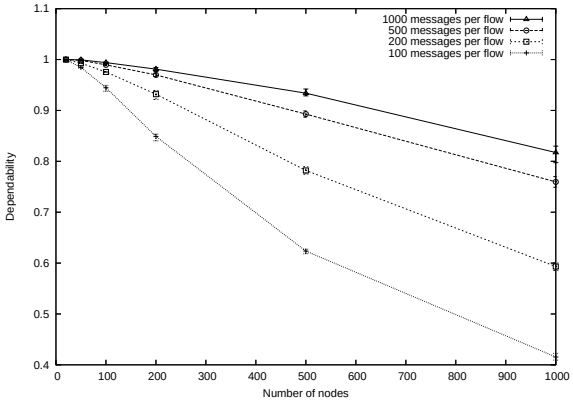
Figure 4(a) shows the impact of faulty nodes on the four routing methods described in the previous section: flooding, ant routing, per-pair estimators and per-flow estimators. The performance of flooding is largely unaffected by the presence of faulty nodes. Interestingly, the performance of ant routing actually increases when up to 40% of the nodes are faulty, and the same is true of per-pair estimators with up to 20% faulty nodes. This is accompanied by a decrease in the forwarding overhead (not shown here due to space restrictions), suggesting that the improvement might be due to a reduction in the number of redundant messages.

Increasing the number of faulty nodes eventually reduces the performance of ant routing and adaptive routing to

Figure 3: The effect of varying the number of faulty nodes (left) and the flow length (right).



(a) Dependability as a function of the fraction of faulty nodes, in a network of 1000 nodes.



(b) Dependability of per-flow estimators as a function of the network size, for various flow lengths.

the same level as flooding. Dependability does not reach zero even when all nodes are faulty, because communication can still succeed when the originator and destination are neighbours, even though no forwarding is taking place. It would be interesting to compare these results with the effect of simply removing the faulty nodes from the network.

4.3 Flow length

All of the simulations so far have involved an average of 1000 messages per flow. Figure 4(b) shows that as the average flow length decreases, the dependability of adaptive routing with per-flow estimators also decreases. This suggests that our protocol would be most suitable for applications that involve long flows, such as video conferencing, file transfer or instant messaging. (Note that it is the number of messages sent between the same endpoints, rather than the amount of data transferred, that determines the suitability of our protocol.)

5 DISCUSSION

The simulation results clearly show that local adaptation without knowledge of the endpoints can outperform flooding. This is not much of a boast – flooding is known to perform poorly in large networks – but it shows that knowledge of the network topology is not a precondition for making intelligent routing decisions. Our protocol also outperforms ant routing without relying on end-to-end overlay addresses that might be vulnerable to eavesdropping or spoofing.

Adaptive routing works by identifying the implicit and explicit relationships between messages – these include timing, previous and next hops, and flow identifiers. The performance of the protocol therefore depends on the number and strength of those relationships. Long-lived flows give the network time to identify dependable routes, and the initial cost of route discovery can be amortised over the lifetime of the flow (the same is true of routing protocols with an explicit route discovery phase, such as DSR [16]). We therefore believe the overall performance of our protocol would be improved by a more scalable method of route discovery.

5.1 Applications

Adaptive routing could be useful in private peer-to-peer overlays or *darknets*, where information about the structure and membership of the network is intentionally withheld for reasons of security and privacy. The protocol might also be applicable to mobile *ad hoc* networks, where accurate information about the topology may be unavailable due to node mobility, variable signal conditions, and the previously mentioned problems of identity and trust common to all open-membership networks.

In terms of traffic analysis, flow identifiers reveal less information than end-to-end addresses: an eavesdropping relay can determine how much information is being sent, and when, but not from whom or to whom. The round-trip time between sending a message and receiving an acknowledgement might reveal the network distance to the

destination, but the network distance to the originator would still be unknown – an attacker would need to observe multiple nodes or network links to trace a flow from its origin to its destination. An attacker with knowledge of the network topology would be able to assign higher probability to some originators than others [3]. We do not expect our protocol to provide unlinkability under the stronger attack models typically used to evaluate high-latency mix networks [25, 26].

Our protocol would not be suitable for devices with very little storage capacity, such as mobile phones, due to the overhead described in Section 3.9. The simulation results also suggest that our protocol is unlikely to be suitable for very large networks – the route discovery process does not scale well in its current form. Because of the need to amortise the cost of route discovery over the length of the flow, adaptive routing is more likely to be useful for applications that produce long flows of messages between the same endpoints – such as video conferencing, file transfer and instant messaging – than for applications that produce short flows between a large number of endpoints, such as web browsing and email.

5.2 Future work

The simulation results presented here are only preliminary. Much work remains to be done to evaluate adaptive routing in a wider range of scenarios: issues to consider include churn, mobility, heterogeneous bandwidth, and complex topologies such as social networks.

We are currently exploring ways to improve the scalability of route discovery using a meet-in-the-middle technique based on the U-ACK mechanism. The simple dependability estimators described in this paper can doubtless be improved upon, perhaps by using control theoretic techniques such as Kalman filters.

We would also like to explore our protocol's resilience to a wider range of malicious and strategic behaviour. Classical approaches to Byzantine fault tolerance involve strong authentication [19, 23, 5]; we would like to see whether it is possible to achieve weaker probabilistic guarantees without authentication, by reducing the information available to faulty nodes in order to restrict the potential complexity of their misbehaviour.

REFERENCES

[1] Avramopoulos I., Kobayashi H. and Wang R. (2003), 'A routing protocol with Byzantine robustness', in *IEEE Sarnoff Symposium*.

[2] Awerbuch B., Curtmola R., Holmer D., Nita-Rotaru C. and Rubens H. (2005), 'On the survivability of routing protocols in ad hoc wireless networks', in *Proceedings of the 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm 2005), Athens, Greece*, IEEE Computer Society Press, pp. 327–338.

[3] Borisov N. (2005), *Anonymous Routing in Structured Peer-to-Peer Overlays*, Ph.D. thesis, UC Berkeley.

[4] Boyan J. and Littman M. (1994), 'Packet routing in dynamically changing networks: A reinforcement learning approach', *Advances in Neural Processing Systems*, 6:671–678.

[5] Castro M. and Liskov B. (1999), 'Practical Byzantine fault tolerance', in *3rd Symposium on Operating Systems Design and Implementation, New Orleans, LA, USA*.

[6] Cheshire S., Aboba B. and Guttman E. (2005), 'RFC 3927: Dynamic configuration of IPv4 link-local addresses', IETF standard.

[7] Claerhout B. (1996), 'A short overview of IP spoofing: Part I', Available from `http://www.cs.ucl.ac.uk/staff/mrogers/cache/ipspoof1.txt` (accessed September 2007).

[8] Claerhout B. (1997), 'A short overview of IP spoofing: Part II', Available from `http://www.cs.ucl.ac.uk/staff/mrogers/cache/ipspoof2.txt` (accessed September 2007).

[9] Danezis G. and Clayton R. (2007), 'Introducing traffic analysis', in A. Acquisti, S. di Vimercati, S. Gritzalis and C. Lambrinoudakis (eds.), *Digital Privacy: Theory, Technologies, and Practices*, CRC Press.

[10] DiCaro G. and Dorigo M. (1998), 'Ant colonies for adaptive routing in packet-switched communications networks', in *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature, Amsterdam, The Netherlands*, Springer-Verlag, vol. 1498 of *Lecture Notes in Computer Science*, pp. 673–682.

[11] DiCaro G., Ducatelle F. and Gambardella L. (2005), 'AntHocNet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks', *European Transactions on Telecommunications*, 16(5).

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

[13] Elmer-DeWitt P. (1993), 'First nation in cyberspace', *Time*, 142(24).

[14] This reference has been removed during the anonymous review process.

[15] Friedman E. and Resnick P. (2001), 'The social cost of cheap pseudonyms', *Journal of Economics and Management Strategy*, 10(2):173–199.

[16] Johnson D., Maltz D. and Broch J. (2001), 'DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks', in C. Perkins (ed.), *Ad Hoc Networking*, Addison-Wesley, chap. 5, pp. 139–172.

[17] Karagiannis T., Papagiannaki D. and Faloutsos M. (2005), 'BLINC: Multilevel traffic classification in the dark', in *SIGCOMM 2005, Philadelphia, PA, USA*.

[18] Kent S., Lynn C., Mikkelson J. and Seo K. (2000), '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*.

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

[20] MUTE website, `http://mute-net.sourceforge.net/` (accessed September 2007).

[21] Pain J. (ed.) (2006), *Internet Annual Report*, Reporters Without Borders.

[22] Partridge C., Cousins D., Jackson A., Krishnan R., Saxena T. and Strayer W. (2002), 'Using signal processing to analyze wireless data traffic', Technical Memorandum 1321, BBN Technologies, Cambridge, MA, USA.

[23] Perlman R. (1988), *Network Layer Protocols with Byzantine Robustness*, Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

[24] Pfitzmann A. and Köhntopp M. (2001), 'Anonymity, unobservability, and pseudonymity - a proposal for terminology', in H. Federrath (ed.), *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA*, Springer-Verlag, vol. 2009 of *Lecture Notes in Computer Science*, pp. 1–9.

[25] Raymond J. (2001), 'Traffic analysis: Protocols, attacks, design issues and open problems', in H. Federrath (ed.), *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA*, Springer-Verlag, vol. 2009 of *Lecture Notes in Computer Science*, pp. 10–29.

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