Chapter 7

Stamp Trading Networks

This chapter describes how capacity constrained flow networks may be used to build a scalable and robust peer to peer network infrastructure. Peer to peer networks have to deal with the following questions, usually much more straightforward in centralized networks:

- How can a node discover other nodes to communicate with?
- If there is a subset of nodes offering a particular service, how can a node discover nodes in that subset?
- How can scarce resources (storage and bandwidth) be allocated in an open network, where services are provided to all peers?

These questions are so difficult that most implementations of peer-to-peer networks decentralize only part of the total network architecture (for example, movement of bulk data), while retaining centralized servers for this "network metadata." Only very innovative

designs such as Freenet[**?**] and Gnutella[**?**] embrace a completely decentralized approach, and such designs tend to suffer from both scalability problems and vulnerability to malicious attacks. For example, there is discussion of "cancerous nodes" in the Freenet community.

This chapter describes a partial design for a peer to peer network in which the discovery of other nodes, as well as the provisioning of services, is governed by a flow of "stamps," or simple authentication tokens. The initial flow of stamps is determined by manually configured trust relationships, similar to those of Advogato and the other trust metrics described in this thesis. The flow of stamps is effectively a capacity-constrained network, thus providing the network as a whole with some attack resistance.

The design presented in this chapter is somewhat speculative. It has not yet been implemented, even as a prototype, and there is no doubt a considerable amount of refinement needed to become truly practical. Even so, it represents a dramatic improvement in robustness over any peer to peer network design currently known, and at a level of efficiency rivalling that of the most advanced designs.

7.1 Stamps

We assume a peer-to-peer network of *nodes* on the Internet. Each node is implemented as a daemon process with access to storage for the node's state. Further, each node has an associated public/private key pair. The hash of the node's public key (typically a 160 bit string, assuming SHA-1 as the hash function) serves as its id. Each node listens for connections on an IP address and port, and can request connections to other IP address/port tuples. Such a model is entirely standard, and is shared by numerous other

peer-to-peer networks, including MNet[34], Freenet[**?**], and Freehaven[11].

We use the XOR distance metric between node id's, as proposed in Kademlia[**?**]. The distance $d(x_0, x_1)$ between two nodes with id's x_0 and x_1 is simply $x_0 \oplus x_1$, the bitwise XOR of the id values.

As with all the trust metrics discussed in this dissertation, we also assume trust links between the nodes. The owner of each node will enter a list of node id's of other trusted nodes. These correspond to successors in a trust graph. With the addition of these trust links, the model is quite similar to the distributed network model for generalized metadata as presented in Section 6.4.

Nodes on a stamp trading network also, naturally, create and circulate *stamps*. A stamp is essentially a five-tuple:

- The *issuer,* which is the node id of the node that created the stamp.
- A random nonce, used by the issuer to check the validity of the stamp.
- The current IP address and port number for the issuer.
- The expiration date.
- The stamp's *value,* which is a real number.

Each node creates a steady stream of stamps, and distributes them to its successors in the trust graph. After circulating in the network, stamps are *redeemed* by a node with a connection to the issuer, in return for services. The issuer verifies that the nonce belongs to a stamp it issued, the expiration date has not passed, and that the stamp hasn't already been redeemed. If so, it goes on to provide the service requested.

A number of peer-to-peer network designs provide services in exchange for some currency-like token. Mojo Nation[34] is perhaps the best known of these, using a "token server" as the source of currency tokens (known as "mojo") and validation service for all token transactions. Mojo Nation was inspired by the agoric systems literature, including the classic papers by Miller and Drexler[23, 22].

7.1.1 Stamp trading

Each node contains a set of "stamps on hand." These consist of stamps received directly from predecessors in the trust graph, and stamps acquired through trading operations.

Nodes provide the service of offering stamps from its stamps on hand collection. A critical design property is the policy used for setting the *exchange rate* for trading stamps, in other words the ratio of total value of stamps given to the total value of stamps redeemed. One such policy is given below in Section 7.3.

Each node has a background process tasked with maintaining a particular distribution of stamps. The space of node id's is divided into *buckets,* each representing an interval in the space of node id's. These intervals form an exponential series, with the smaller intervals clustered around the node's id. The background process aims to keep the total value of stamps in each bucket roughly the same. In particular, it will attempt to acquire stamps for underfull buckets by redeeming stamps in overfull buckets.

We can define the buckets more precisely. For $0 < i < k$, bucket i for node id x covers all distances from x in the range $[2^{160+i-k}..2^{161+i-k})$. Bucket 0 covers all distances in the range $[0..2^{161-k})$. Typically, k will be on the order of $\lg N$, where N is the number

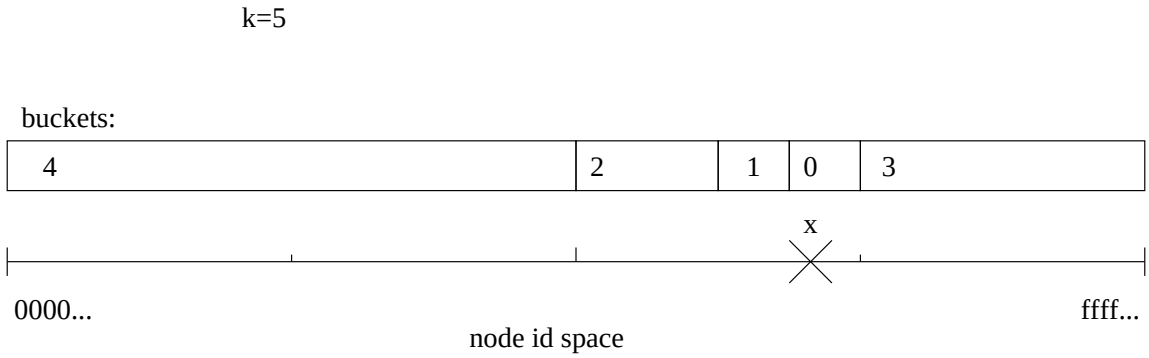

Figure 7.1: Example of XOR-metric buckets.

of nodes in the system. An example with 5 buckets ($k = 5$) is shown in Figure 7.1.

The distribution of stamps on hand induces a "stamps on hand" graph. If node s has stamps issued by t on hand, then there is an edge from s to t in this graph. If a node wishes to obtain services from some arbitrary other node, it must find a route to that node in this graph. For each node in this route it redeems stamps issued by that node in exchange for stamps issued by the next node in the route. Note that, unlike the trust graph, which is manually configured and generally fairly stable, the stamps on hand graph may change quite dynamically.

The distribution of buckets is very similar to that of Chord's "finger tables"[10]. Assuming that this background process is successful in maintaining the even distribution of stamp value in the buckets, then the result of Chord that nodes can find short ($O(\log N)$) routes to arbitrary other nodes applies to stamp-trading as well.

TODO: summarize Kademlia argument. Briefly, each hop reduces the distance to the target by 1/2 in the worst case.

7.2 Attacks on the stamp trading network

There are at least two distinct attacks on the stamp trading network worth considering. One is the attempt by bad nodes to consume resources. This is a noted problem in peer-to-peer networks, where the resources in question are typically bandwidth and storage[1]. However, other resources, such as human attention paid to emails, faxes, voicemail messages, instant messages, etc., may be even more worthy of zealous guarding. Spam is one important subclass of this class of attacks[14].

Given minimal assumptions about the setting of exchange rates, movement of stamps through the network is a capacity constrained flow network. This bounds the amount of services an attacker can consume by the number of edges to bad nodes. TODO: state these minimal assumptions on exchange rates.

A dual attack is to block good nodes from acquiring valid stamps. In the system presented above, there is nothing preventing an attacker from flooding the network with *bad stamps*. These stamps may come in many different flavors:

- Entirely valid stamps, but for bad nodes.
- Valid stamps issued by good nodes, but already spent.
- Completely fabricated stamps.

Some of these attacks may be preventable with known techniques, such as using Chaum's digital cash protocols[8] rather than opaque random nonces for the stamp data. However, such techniques are ineffective in the face of valid stamps for bad nodes. Thus, we seek more general techniques that cover all such attacks.

There are two such techniques worth considering. First, we can simply prune all non-reciprocal trust links. Nodes can simply refuse to accept immediate stamps from their trust predecessors unless they are also trust successors. This pruning restricts the inflow of bad stamps to edges from good to bad nodes.

Second, the policy for setting exchange rates will ideally drive the exchange rates for bad stamps to zero, at which point they cannot do any damage.

7.3 Exchange rate policy

In outline:

Note that some bad stamps (double-spent and fabricated) have detectable consequences: the attempt to redeem the stamp will fail. However, it's not always possible to detect which peer was responsible. Intuition: you can keep track of at least some of the possible wrongdoers. When you detect a failure, ding all nodes that may have been responsible. This idea is similar to that of Dingledine's reputation system for MIX cascades[12].

For each stamp on hand, maintain an audit trail of trading partners responsible. For immediate stamps received from trust predecessors, tag each stamp with a singleton list containing the stamp's issuer (ie the predecessor). When a stamp is redeemed in exchange for new stamps, assign the new stamps the audit trail for the old stamp, and append the issuer of the new stamps. Invariant: the last element in the audit trail is the stamps issuer.

Each node maintains a "credibility" factor for (some subset) of other nodes. The final redeem of a chain of stamp trades can either succeed or fail (with a number of subflavors). Apply ++ or – (respectively) to the credibilities of all nodes listed on the stamp's

audit trail.

Nodes nearby in the trust graph and/or nearby in Chord distance will participate in many transactions, thus the credibility ratings are likely to be accurate. The credibility rankings will become less useful for nodes farther away.

Each stamp also has a *confidence factor*. For stamps from immediate trust predecessors, confidence factor has a (high) initial value. As part of the stamp trading service, report confidence factors. For every stamp trade, new stamp has confidence factor = old stamp's confidence factor times old stamp's issuer's credibility factor times new stamp's reported confidence factor times a damping factor. All these factors better be damn near 1, as exponent is $O((\log n)^2)$.

Exchange rate is set based on confidence, but will reflect other factors (such as difficulty of getting the stamp, and/or difficulty of keeping bucket full). Also, want to make sure that exchange rate on immediate stamps (ie those obtained from trust predecessors) is inversely proportional to volume of stamps received.

7.4 Applications

Potential application for stamp trading: spam-resistant email. One variation: whether to ring the target's cell phone or go to voicemail.

7.5 Open questions

The design presented in this chapter is quite speculative. There are two large open questions that should probably be answered before the design is implemented.

First, can the system efficiently and reliably find short routes through the network? An analysis similar to Kleinberg's algorithmic analysis of the small-world property[17] would appear to be fruitful, but Kleinberg's analysis can not be applied directly to the stamp trading network. Under the reasonable assumption that the trust graph obeys the small-world principle, it is clear that such short ($O(\log n)$) routes exist, but it is less clear that the system can find them using only locally available information.

Second, is the policy described in Section 7.3 sufficient to prevent against false-negative attacks?

These questions remain as directions for future research.