Lattice: An Anonymous, Scalable Peer-to-Peer System

Sean Blanchfield and Mads Haahr

Distributed Systems Group
Department of Computer Science
Trinity College Dublin, Ireland

Introduction

The Peer-to-Peer (P2P) communications paradigm is emerging as the infrastructural basis for a new suite of Internet applications. Different applications deploy different topologies, i.e., ways of arranging the nodes to form P2P networks. Each topology has its own characteristics in terms of performance and anonymity/privacy, and there is often a trade-off between those characteristics involved. This position paper describes a P2P system based on a novel topology aimed at achieving high scalability in terms of number of nodes and communication, while retaining strong user anonymity. The most prominent feature of the topology is its highly structured nature, which allows a method of anonymous addressing. The approach described in this paper is designed to be generally applicable to P2P applications. At the time of writing, it is being deployed in a file sharing application for proof-of-concept purposes.

Design

The protocol forms a decentralised peer-to-peer network, termed the *lattice*. This name derives from the network's grid-like topology, similar to cellular automata's common Von-Neumann topology [Neumann, 1966]. The lattice can be visualised as a two-dimensional grid, with nodes positioned at the intersections. Each node maintains connections to four other nodes, which are positioned at the intersections in the lattice directly to its north, south, east and west. This network structure is inspired by the "Grid" topology proposed by Ben Houston [Houston, 2000], which proposes a method of efficient broadcasting for a Gnutella clone [Kan, 2001].

In order to ameliorate the connectivity of nodes that are situated near the edge of the lattice, the protocol contains a functionality that enables opposite edges of the lattice to join together, thus forming a toroid, as shown in Figure 1. The connections between the edges are called rubber connections, and are an exception to the regular structure that the rest of the lattice exhibits. Hence, messages sent via rubber connections are handled slightly differently than normal messages. This paper focuses on the normal (non-edge) behaviour of the network, and the treatment of the routing protocol assumes the network is fully populated (i.e., does not contains vacancies). A more detailed treatment dealing also with these issues can be found in [Blanchfield, 2001].

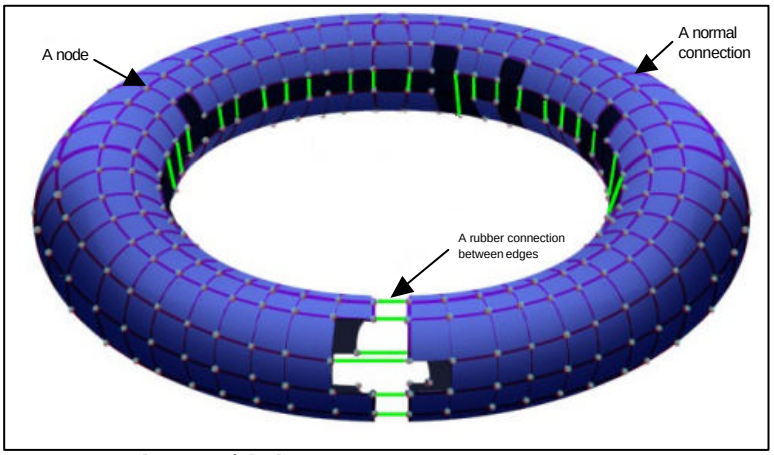

Figure 1 visualisation of the lattice

Broadcasting

The file sharing application allows nodes to issue queries (search requests) and receive responses. The former are issued via a *broadcast* mechanism and the latter via directed *unicast* back to the origin of the search request. When a query is to be broadcast, the node in question sends the query to its four neighbours, who in turn pass it on to certain neighbours of their own, and so on. The forwarding algorithm is as follows:

(1) If generating the query, send it to the N, S, E and W neighbours.
(2) If the query is received from the S node send it to the N, E and W neighbours.
(3) If the query is received from the N node send it to the S, E and W neighbours.
(4) If the query is received from the W node send it to the E neighbours.
(5) If the query is received from the E node send it to the W neighbours.

These rules produce the propagation shown in Figure 2. Because of the grid structure, there are no loops in the graph and no node can receive the same query twice. Hence, redundant traffic is prevented without the need for explicit loop detection, and the depth of the search tree is not limited by loops in the neighbour graph. Broadcast messages contain addressing information that specifies how many hops the message can take before it ceases to propagate. This is called the Hops-To-Live (HTL) count, and effectively defines the search area for a query.

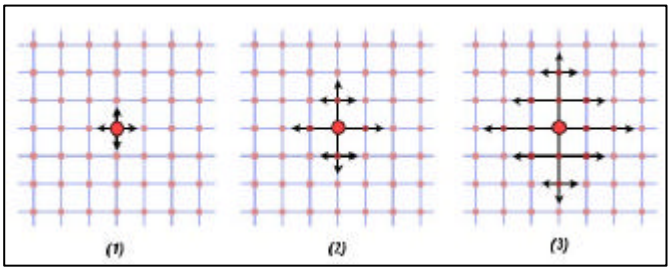

Figure 2 The propagation of a broadcast in the lattice

Unicasting

Every broadcast query contains a unique transaction identifier (UTID) that is recorded by each node that it passes through. A response packet can retrace the path of the original query to the target node using the UTID. This is called a unicast and it allows two nodes to communicate together via the lattice. Unicasting is normally used to respond to previous broadcasts and to other messages.

Anonymous Addressing

The regular structure of the lattice enables us to anonymously address individual nodes. The requirement for anonymity means that we cannot address a node on the basis of any characteristics that can identify the user of that node, e.g., an IP address. We therefore address nodes on a basis that requires no sensitive identifying information about the node – namely the location of the node in the lattice structure.

A node can imagine itself to be at the origin of a Cartesian coordinate plane that is superimposed on the lattice. All other nodes then occupy a unique relative (x,y) coordinate. If node A wishes to anonymously address node C and transmit the anonymous address to node B then we must reconcile the differences between the relative position of node C with respect to node A and node B.

We solve this problem by performing a correction on the coordinates at each hop the message takes between node A and B. For example, say the message that initially travels west across the lattice. After the first hop the coordinates can be corrected by adding 1 to the x value. In general the coordinates can be corrected by adding or subtracting 1 from either the x or y values, depending on which direction the message was received from. When node B finally receives the message it will find that the coordinates have been incrementally translated so that they actually reflect node C's position relative to B.

It is now a simple matter to extend the protocol to allow nodes to contact each other on the basis of their relative coordinates. To achieve this we ensure that such a message is relayed at each hop in a

direction such that either the *x* or *y* coordinate is adjusted to zero. The message will be routed in this manner until eventually the relative coordinates that it contains become (0,0), which indicates that it has reached its destination.

Large Data Transfers

Once a node has responded to a request, a mechanism is required that enables the transfer of large amounts of data (e.g. files) between nodes without compromising anonymity. Although it would be possible to do this by passing the data along a chain of nodes on the lattice as a unicast, this would be undesirable due to the increased bandwidth demands it would incur on individual nodes.

The mechanism that is provided is in essence a compromise between a direct transfer between the two parties, and a unicast transfer using a chain of proxies. A single proxy node is chosen with prejudice only towards its bandwidth. The major concern is that it must be impossible for either of the two party nodes to affect the identity of the proxy. If this were the case, it would be trivial for one of the nodes to arrange for a colluding node to be chosen as the proxy, and thus gain access to the identity of the other party. With this in mind, the system relies on anonymous addressing to uniquely select a third node. To avoid either of the two nodes having the ability to control the choice of the third node, each node is allowed to specify only one of the coordinates of the third node. When both parties have negotiated a set of jointly chosen relative coordinates they use anonymous addressing to contact the third node. This node then selects a proxy with a suitable bandwidth from the set of known nodes participating on the lattice, and informs the two parties of its IP address via a unicast.

The two parties to the transfer then exchange public keys via the proxy, encrypt the relevant data and transfer it via the proxy. This process is illustrated in Figure 3.

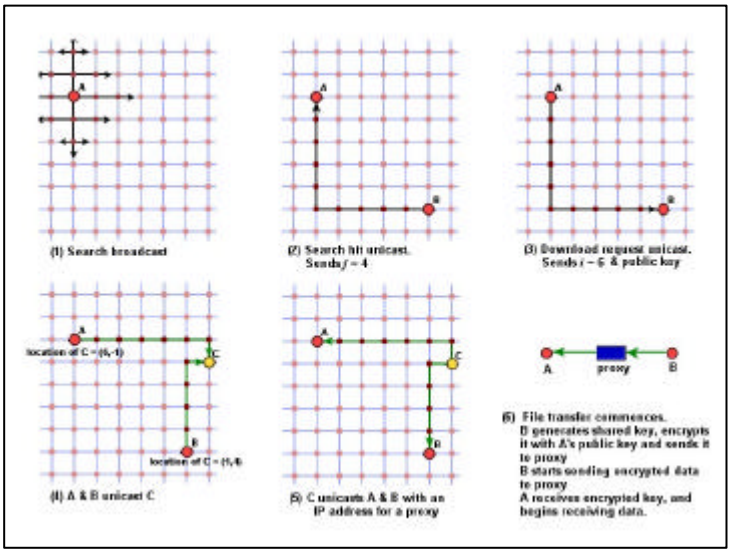

Figure 3 Negotiation of a large data transfer

How New Nodes Join the Lattice

In order to preserve the coherency of the lattice structure it is essential that new nodes join the lattice at the correct positions. There are several considerations when inserting a new node into the lattice:

(1) The node should be inserted at a point where its neighbours will be of a reasonably similar bandwidth.
(2) The node should be used to fill a vacancy in the lattice structure, if possible.
(3) Failing this, the node should be inserted at the perimeter of the lattice. This should supersede any rubber connections that exist.

The lattice learns about vacancies that exist through information that is piggybacked on regular broadcasts that are sent by nodes. A broadcast packet contains a number of fields used to indicate vacancies of various bandwidth categories. Any node that handles a broadcast can set the appropriate

field if it is adjacent to a vacancy. All nodes that see broadcast packets also record information about vacancies with the UTIDs of the respective packets.

When a new node wishes to join, it must first contact a current node on the lattice. That node will look up its local information about vacancies, and use the UTID to send a unicast in the direction of a suitable vacancy. When a node adjacent to a vacancy receives this unicast, it makes a decision whether or not to accept it on the basis of the new node's bandwidth. If it accepts, it will notify the new node. It will then proceed to notify the surrounding nodes, those that will become the new node's neighbours, as shown in Figure 4.

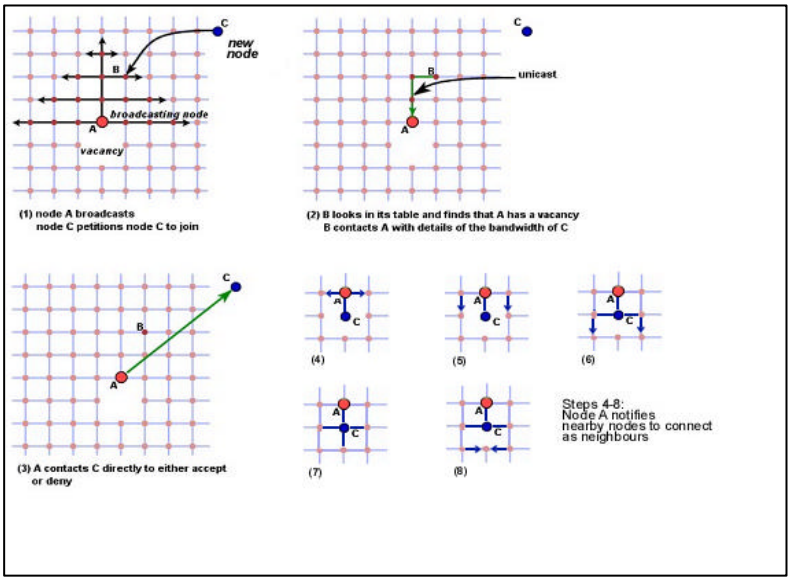

Figure 4 A new node connecting

Bandwidth Clustering

As described above, new nodes join the lattice at points where the bandwidth of the surrounding nodes is similar to their own. This results in nodes clustering together into areas of similar bandwidth. Areas of high bandwidth exist, which graduate into other areas of low bandwidth. This is desirable in the sense that it minimises the effect of bottlenecks. If low bandwidth nodes were evenly distributed throughout the lattice, they would have a greater disruptive potential towards the propagation of broadcasts. When they are isolated together a broadcast is able to propagate further without

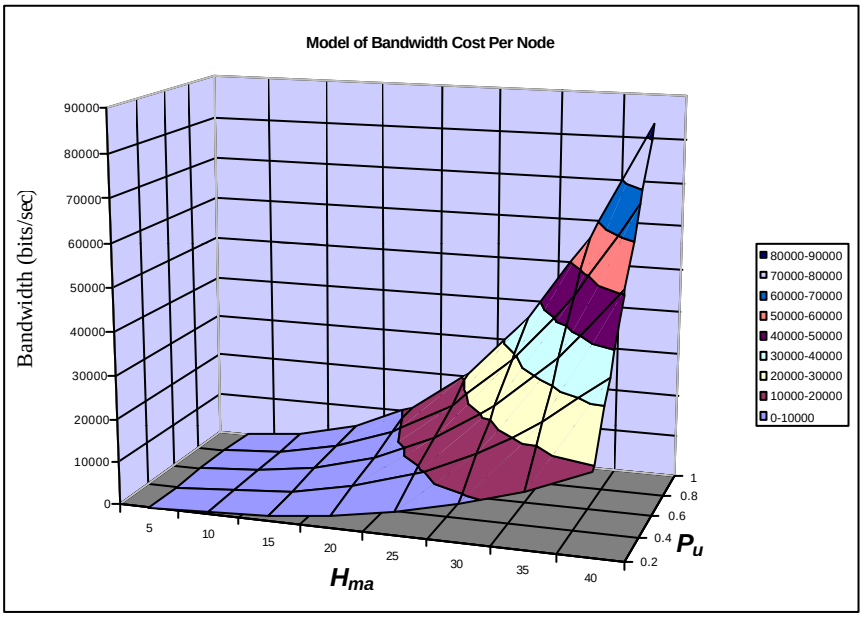

Figure 5 Bandwidth cost of participation on the network

encountering any potentially malfunctioning low bandwidth nodes.

Preliminary Evaluation

A functional client has not yet been deployed; therefore a mathematic model has been drawn up to measure the expected performance of the network. Figure 5 shows an estimate of the level of traffic that a node must handle on behalf of other peers, plotted against two factors, H_{max} and P_u. H_{max} represents the average Hops-to-Live (HTL) value that determines how many hops a broadcast can take before ceasing to be relayed any further. P_u represents the probability that any particular node will respond to a broadcast, thereby generating a unicast.

The figure shows us that even if $P_u=1$ (i.e. a worst-cast scenario where every node generates traffic by responding to broadcasts), low bandwidth users can still actively participate up to a H_{max} of approximately 40. This equates to over three thousand contactable nodes. Under normal conditions this figure could be expected to be higher.

Conclusion and Future Work

This paper has described a P2P topology called the lattice, a highly structured network of peers. This allows the protocol functions to be designed with detailed knowledge of the topology involved. As a result we can easily optimise broadcasts to eliminate redundancy. Another novel property of the approach is that it is possible to address nodes on the basis of their position in the structure, which is the basis for the network's strong anonymity properties. As of the time of writing, the protocol is being implemented for a file sharing application, with the aim of collecting data on its large-scale performance. The longer-term goal is to develop the protocol for more general applications.

Future Work

A possibility that will be investigated is that of stratifying the lattice into multiple layers, on the basis of bandwidth. This would allow for a more strict separation between nodes of different bandwidths. All nodes are currently on the same layer; therefore low-bandwidth nodes must deal with the traffic that high-bandwidth nodes are generating. As it is very likely that the users of high-bandwidth nodes will often be uninterested in downloading data from low-bandwidth nodes, much of this traffic may be redundant. By separating the lattice into layers we can allow each node to specify which layers it would like each broadcast to reach. There would exist a layer for each major bandwidth category, and each layer would be permitted to establish connections to every higher layer. By default broadcasts from low bandwidth layers would be sent to higher layers, and if explicitly desired broadcasts from higher layers could be sent to lower layers.

Another possibility is to extend the lattice into more than two dimensions. Although a two-dimensional arrangement is convenient for visualisation, it is arbitrary from a functional point of view. The lattice may perform better in a three-dimensional, four-dimensional or n-dimensional format. Until a certain point, higher dimensionality results in greater connectivity and responsiveness due to the greater number of peers in the vicinity of any one node. In the case of a 3-dimensional lattice, the edges of the network would still connect together (thus forming a hyper-toroid).

References

[Blanchfield, 2001] Blanchfield, S. An Anonymous and Scaleable Distributed Peer-to-Peer System. Undergraduate thesis, Trinity College Dublin. Online at http://ww.cs.tcd.ie/Sean.Blanchfield/

[Houston, 2000] Houston, B. 2000. The Grid P2P Topology Proposal. Online at http://www.exocortex.org/p2p/grid.html

[Kan, 2001] Kan, G. Gnutella. 2001. In Oram, A., (ed) *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology*, O'Reilly and Associates, Inc., Sebastopol, California.

[Neumann, 1966] Von Neumann, J. 1966. Theory of Self-Reproducing Automata. In A. W. Burks (ed), Univ. of Illinois Press, Champaign.