



School of IT  
Technical Report



**The University of Sydney**

**IMPLICIT GROUP MESSAGING IN  
PEER-TO-PEER NETWORKS  
TECHNICAL REPORT 583**

DANIEL CUTTING AND BJÖRN LANDFELDT  
SCHOOL OF INFORMATION TECHNOLOGIES  
UNIVERSITY OF SYDNEY, AUSTRALIA

AARON QUIGLEY  
SCHOOL OF COMPUTER SCIENCE AND INFORMATICS  
UNIVERSITY COLLEGE DUBLIN, IRELAND

JANUARY, 2006

# Implicit group messaging in peer-to-peer networks

Daniel Cutting, Björn Landfeldt  
School of Information Technologies  
University of Sydney, Australia  
{dcutting,bjornl}@it.usyd.edu.au

Aaron Quigley  
School of Computer Science & Informatics  
University College Dublin, Ireland  
aquigley@ucd.ie

## Abstract

We describe “implicit group messaging”: a form of many-to-many message delivery that does not require publishers to enumerate recipients, or recipients to join explicit groups. Instead, publishers specify the characteristics of intended audiences and the system dynamically attempts to deliver messages to all matching members. We study this type of message delivery in the context of peer-to-peer networks, discussing desirable features of such a system, and presenting a novel model based on a distributed overlay network and geometric routing protocols.

## 1 Introduction

The amount of traffic carried by the Internet continues to increase each year [10, 19] and new technologies are constantly appearing which enable more people to produce increasingly specialised content [14, 15, 16, 18]. Already, many channels exist for publishers to reach interested or relevant consumers including web pages, email, instant messaging, blogs and podcasts. They can however be broadly classified into two main approaches<sup>1</sup>:

1. messages are sent to explicit lists of recipients (e.g. mailing lists or instant messaging)
2. messages are posted in publicly accessible formats where interested people can find them (e.g. web sites, blogs and message forums).

The first approach is suitable for some applications, but typically relies on recipients to explicitly subscribe before they can receive messages. If many such lists exist, they need to subscribe to many separate systems in order to receive all of the messages in which they’re interested. Furthermore, since such lists are subject-based, they may be interested in only a subset of all

messages posted. An obnoxious extension of this explicit list approach is email spam, where a publisher indiscriminately *pushes* a message to as many people as possible in the hope of reaching a few interested or relevant people<sup>2</sup>: one reason spam is so irritating is that practically none of it is interesting to the recipient.

The second approach is to post messages to public channels for consumers to find. It is often presumed that public web pages and posts in message forums are published for the benefit or consumption of all users of the Internet but while it is certainly true that anybody with a browser can visit and read a message forum, they are typically tailored to a more specific group of people: perhaps fans of a particular sport or band, or those with an interest in gardening. This public channel approach can be used where compiling explicit lists of recipients is impractical or impossible. There are some drawbacks however: people must sift through irrelevant information (scanning through search engine results, for example); and they must explicitly initiate the searches (i.e. *pull* the messages to them) which means time-sensitive messages may not be discovered soon enough. Finally, as with the former approach, the degree of effort on the part of the recipients increases as the number of publishing sources increases.

We suggest that in situations where the publisher of a message knows the intended *type* of recipient (rather than their explicit names), it is more appropriate to publish to “implicit groups” of people — groups where members are not enumerated but are specified by their characteristics, e.g. “young Australians interested in football”. This allows messages to be delivered in a timely fashion and because recipients receive predominantly relevant messages, reduces the amount of searching and sifting needed.

To concretise: a physician at a hospital is having trouble diagnosing a patient with symptoms of hypoxia

---

<sup>1</sup>We also discuss a third approach, publish/subscribe messaging, in section 4.

---

<sup>2</sup>Although the term spam often refers to junk email, we use it here in its broader sense of *any* unsolicited or irrelevant messages that are pushed to somebody.

and dysphonia that seem to indicate a severe throat infection and would like to get some assistance from colleagues with experience of similar cases. Her hospital has an email system she can use to contact her colleagues around the country, but although she knows the type of person she needs to talk to, she does not know precisely who among her colleagues may be able to help. She must decide whether to contact just a few or the entire group, and since the situation is relatively urgent she sends an email to everybody. Within a few hours, she gets a response with the information she needs to make the correct diagnosis: it is in fact acute epiglottitis, not laryngitis as she had supposed. Unfortunately, she also receives many replies berating her for the irrelevant message. If instead she had been able to send the message to just the implicit group of colleagues with experience of the symptoms she had observed she would not only have saved network resources, but also spared most of the staff from reading a message of no relevance to them.

This concept of implicit group messaging is useful in many scenarios: as the Internet enables the publication of more messages through technologies such as blogs, podcasts, online photo galleries and massive multi-player games, implicit group messaging is a potentially useful way of connecting publishers to appropriate consumers.

### 1.1 Problem and hypothesis

Our goal is not to prove that implicit group messaging is the best strategy for mitigating “information overload” in an increasingly prolific society. Rather, we assume that it has its place in the gamut of possible approaches, and intend to show that it can be plausibly achieved. The overarching problem is to demonstrate that it is possible to deliver discrete messages from producers to implicitly specified groups of consumers in an effective, efficient and robust manner.

This is a broad challenge, encompassing such diverse fields as networking, information retrieval and user modeling. We thus constrain the problem: our domain is peer-to-peer (P2P) systems over connected internet-work; all producers and consumers are equal peers in the network; and each peer maintains a set of keywords describing its characteristics or *attributes* (such as its capabilities, services or user’s interests). We are not concerned with how keywords are actually assigned, although like many Internet-based phenomena we assume that they will follow a Zipfian distribution [11, 1]. Implicit groups are specified as boolean expressions of keywords — all peers whose attributes match an expression are members of that implicit group.

Within this scope, publishing a message to an im-

1. All group members should receive group messages.
2. Non-members should not receive group messages.
3. The members of an implicit group should be the exact set of peers that have matching attributes at the instant the message is published.
4. Group members should receive messages as soon as they are published.
5. Messages should be “replayable” in the event that some peers are transiently connected.
6. The overall load on the network should be minimal.
7. No peer should do more work or be more critical to system operation than any other.

**Figure 1. Ideal properties of a P2P implicit group messaging system.**

PLICIT group becomes a form of multicasting from a source peer through a network to the subset of peers with keywords that satisfy the group description specified by the publisher. Consumers do not explicitly join groups or subscribe to any particular types of messages. Any peer must be capable of sending messages to any implicit group at any time, and the actual membership of the same implicit group may vary from message to message as peers join and leave the system or alter their attributes.

An implicit group messaging system as described above needs to have certain basic properties, which we enumerate in figure 1. Any model seeking to address the problem should satisfy these properties as far as possible, even if various properties may conflict in some situations (e.g. it may sometimes be necessary to load a peer more heavily in order to deliver the message to more group members). Of course these tradeoffs will vary for different models, but we consider the delivery of messages to all members of an implicit group to be of utmost and overriding importance.

Our contributions are several. We describe the implicit group approach to messaging and provide a network model for achieving it in large P2P networks using a number of novel concepts: implicitly addressing a peer based on its attributes and a routing algorithm based on the geometric clustering of peers in a logical address space. Our model addresses most of the properties listed above, with the remainder assigned to ongoing and future work.

Sections 2 and 3 describe our network model and associated algorithms and sections 4 and 5 discuss related work and conclude with future work.

## 2 Model

### 2.1 Background

P2P networks are overlay networks that connect Internet hosts to one another for the purposes of sharing data or delivering messages. They are typically distributed in that no peer is significantly more important than any other, although the degree to which this holds depends on the type of model: centralised, decentralised, unstructured or structured.

The centralised model, which includes systems such as the original Napster<sup>3</sup>, relies on well-known centralised registries. These registries typically index or cache the content of the peers and process queries on their behalf for the purposes of uniting peers that can meet one another's needs.

Decentralised P2P networks, such as the FastTrack network<sup>4</sup>, use a similar approach extended to a two-level hierarchy of peers (clients and "supernodes") in order to scale networks efficiently. Clients usually connect to a single supernode which caches or indexes their content (akin to a server in the centralised model), and search queries are handled by flooding between supernodes.

The unstructured model, which includes such networks as Gnutella<sup>5</sup>, has no concept of servers or heterogeneous peers, but instead connects neighbours according to various ad hoc rules. These heuristics can be as simple as connecting to the first peers encountered, or as complex as connecting to peers with similar interests or of high degree. Searches for content or peers in this model are typically flooded from the source to a certain depth. Some advanced techniques for limiting or directing these floods have been explored [25, 23, 27], although these typically only consider the case where searches are for specific objects (where it is sufficient to find just one or a few objects matching the search criteria). This is inappropriate for an implicit group messaging system, since we need to find and deliver a message to *all* peers matching the implicit group criteria.

Structured P2P networks take a very different approach to connecting neighbours and include such systems as Tapestry [29] and CAN [21]. Typically they seek to organise the peers and connections into an abstract, regular topology that allows guarantees of various properties, such as the maximum hops needed to reach a particular peer, and bounded search times for locating an object stored in the network.

<sup>3</sup><http://www.napster.com>

<sup>4</sup><http://en.wikipedia.org/wiki/FastTrack>

<sup>5</sup><http://www.gnutella.com>

### 2.2 Implicit group messaging model

Our model is a distributed, structured overlay network comprising homogeneous peers. The peers reside at locations (unrelated to their network or geographical location) on an abstract  $d$ -dimensional Cartesian hypercube which wraps around on all edges, thus forming the surface of a  $d$ -torus. Each peer is responsible for a distinct region of the surface and knows how to contact the peers that border it. Messages are routed geometrically across the surface between neighbouring peers.

Recall that each peer maintains a set of attributes describing their services, capabilities or user's interests. Peers use these attributes to calculate their logical address on the surface: thus a peer's location inherently encodes its attributes. Once a peer has joined the network, it keeps track of its region of the surface, and the set of neighbours that are adjacent. When it wishes to publish a message to an implicit group, it calculates the set of all possible regions of the surface where peers of that group may reside, using the same address encoding scheme it used to find its own address. It then uses a geometric routing algorithm which forwards the message from neighbour to neighbour, branching as necessary to efficiently reach all target regions.

This structured overlay network model was chosen since it allows us to satisfy certain of the ideal properties stated above (numbers correlate to those in figure 1):

1. There is a strong advantage to encoding the attributes of a peer into its address since we can calculate all regions of the surface where members of any given implicit group reside, meaning all members of a group can be found.
2. There is no need for flooding messages and no need for specialised registry peers which maintain mappings from attributes to peer addresses, since the locations of all group members are already known and a geometric route can be predetermined. Routing a message from its source to all members will usually incur the cost of routing through some non-members on the way to members however, depending on their locations.
3. Since there are no registries or caches, the membership of an implicit group changes instantly as peers connect and disconnect from the network. However, since peers can join and leave after the message has been initiated, there is no strong guarantee that the set of peers that ultimately receive a message for an implicit group will be exactly the set of peers making up the group at the instant of publication.
4. Messages are forwarded as they are published, so

the time it takes to reach any given peer will depend on how far it is from the source and what route is taken.

5. The “replayable” message property is not supported by the current model, but may be addressed in future work.
6. The minimum possible amount of network load for an ideal model can be determined by calculating the Steiner tree of all member peers and the source peer across the network of Internet routers, ensuring each peer only receives exactly the messages it needs with no duplication. As with many structured overlay models, our model will load the network significantly more than this, since messages will need to traverse several physical network hops for each overlay hop, possibly through the same peers. Various techniques for mitigating this are discussed in section 5.1.
7. Our model is entirely distributed, and no peer inherently needs to do significantly more work than any other. However for any given implicit group, some peers will need to carry more messages (if there are “bottlenecks” between the source and members, for example). Some techniques for reducing this are also considered in section 5.1.

We now discuss the model in more detail.

### 2.3 Addresses and extents

A peer maintains a set of attributes in the form of string keywords (representing, for example, the services it offers or interests its user has, as entered by the user or determined by the application running on the peer). We combine these attributes into a fixed length *address* using a Bloom Filter<sup>6</sup> [3]. An address is not guaranteed to be unique within the system as it is possible for several peers to calculate the same address either by having the same set of attributes or due to hashing collisions. However, there is no reason why several peers cannot reside at the same logical address. When this occurs, the peers with the same address are termed *cohabitants*.

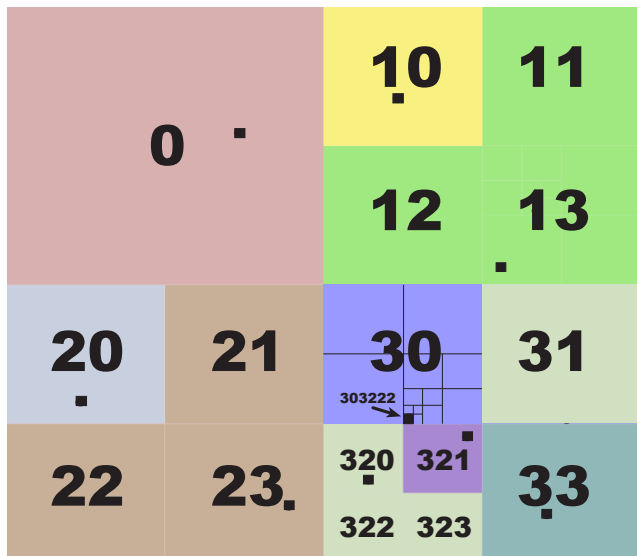
Once calculated, a peer’s address must be mapped in some way to the Cartesian surface. There are many possible functions for this, but we choose to decompose the surface with a generalised PR quadtree [20] (figure 2), using the address as an index into the tree structure. Such structures are commonly used in the spatial

<sup>6</sup>Bloom Filters are compact representations of sets of objects which allow membership tests with an adjustable error rate of false positives, effectively trading precision for reduced storage. Stored as bit strings, filters can incorporate new objects by hashing them to and setting  $k$  of their bit positions. An object can be tested for membership by hashing it and ensuring all  $k$  bit positions in the filter have been previously set.

indexing of points, and although other variations exist, the PR quadtree regularly and equally divides the space at every level, making it easy to name and calculate the Cartesian boundaries of each division. This simplifies how each peer stores its regions of the surface and communicates with neighbouring peers. It also makes certain operations easier, as described below.

The mapping function divides the address into  $d$ -bit segments (padded with 0s at the end if necessary) to form digits of base  $2^d$ . Each digit represents a progressively deeper index into the  $d$ -dimensional PR quadtree. For example, the address 110011101010 maps to the 2-dimensional quadtree address 303222. By assigning digit 0 to the top left quadrant of a 2-dimensional square, 1 to the top right, 2 to the bottom left and 3 to the bottom right, we can recursively refine the Cartesian location of an address by decomposing each digit in turn to the appropriate sub-quadrant. Each quadrant or sub-quadrant in the system is termed an *extent* of the surface.

For example, the four boxes labeled 10, 11, 12 and 13 in figure 2 are all sub-extents of the extent 1. This decomposition can continue for as many digits comprise the address. The figure shows the peer at the quadtree address 303222 as the small black sub-extent of the extent labeled 30.



**Figure 2. Extents on a 2-dimensional surface with their associated quadtree addresses.**

This mapping approach can be applied to surfaces of arbitrary dimensionality without loss of generality. For instance, the same address 110011101010 maps to the

3-dimensional quadtree address 6352 where each digit represents one of the eight boxes formed by bisecting a cube (rather than a square) along each axis.

A peer’s actual address typically occupies a small extent of the surface, but it is possible to have extents of any size. We define  $e_1 > e_2$  to mean extent  $e_1$  covers extent  $e_2$  (e.g. in figure 2, extent 32 covers extent 321, among others) and  $e_1 \sim e_2$  to mean extents  $e_1$  and  $e_2$  are *adjacent* (i.e. they share at least part of one edge and do not cover one another, such as extents 321 and 30 in the figure).  $E_1 \cap E_2$  is the *intersection* of two sets of extents, defined as a set of extents that exactly cover the surface region covered by both  $E_1$  and  $E_2$ .  $E_1 \ominus E_2$  is the *difference* of two sets of extents, defined as a set of extents that exactly cover the surface region covered by  $E_1$  but not covered by  $E_2$ . This differs slightly from an ordinary set difference, since some extents within  $E_1$  may be subdivided to arbitrary depths if a smaller sub-extent needs to be subtracted from them, resulting in a set containing sub-extents not explicitly found in  $E_1$ . For example, referring to figure 2,  $\{0, 2\} \ominus \{20\} \equiv \{0, 21, 22, 23\}$ .

We note that although a quadtree address is used to specify the location of a peer on the surface, there is no global quadtree data structure that needs to be traversed or maintained by the peers. The concept is used primarily to map a peer’s address to Cartesian coordinates. In particular there is no need for a “root” node: the model is entirely distributed.

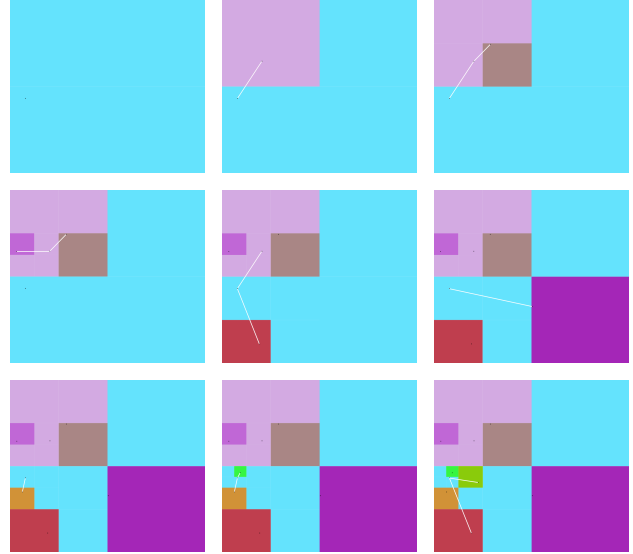
## 2.4 Peers

A peer  $p$  maintains several items of state, including its IP address ( $p^{IP}$ ), its set of attributes ( $p^{attr}$ ) and its address on the surface ( $p^{addr}$ ). Since the surface will likely never be entirely populated, peers are responsible for managing not just the extent covering their address, but also a set of arbitrarily sized extents that cover the addresses of no other peers ( $p^{ext}$ ). For example, in figure 2, the peer at 303222 manages the whole of the extent 30. Whenever a message needs to be geometrically routed through an extent on its way to its destination, it is forwarded to the manager of that extent.

Depending on how the network evolves as new peers join, the extents a peer manages may not always be adjacent (they will however always manage at least an extent covering their own address). For instance, the peer at 320320 manages the extents 320, 322, 323 and 31. This particular configuration occurred because the peers managing extents 30, 33 and 321 all joined the network after the peer 320320, which initially managed the entire extent 3.

Figure 3 demonstrates a similar surface integrating

new peers. Initially the entire surface is owned by a single peer with address 2003... The next peer to join is 0213..., followed by 0300..., 0202..., 2231..., 3122..., 2021..., 2001..., and finally 2013....



**Figure 3. Stages in the formation of a 2-dimensional surface as nodes join.**

A peer also needs to keep track of the extents and IP addresses of the peers that border its own extents ( $p^{nbr} : e \rightarrow IP$ ) in order to be able to forward messages. Peers  $p$  and  $q$  are *neighbours* if they have at least one pair of adjacent extents: i.e.  $p \sim q \iff \exists e \sim e'; e \in p^{ext} \text{ and } e' \in q^{ext}$ . In figure 2, the neighbouring extents of the peer 320320 are 23, 30, 321, 33, 13, and since the surface is wrapped on all edges, also extents 10 and 20.

## 2.5 Implicit groups and targets

An implicit group is the set of all peers that have attributes matching a *target expression*, e.g. a peer with attributes *doctor* and *dysphonia* matches the expression  $(\text{surgeon}|\text{doctor}) \& \text{dysphonia}$ , though a peer with only *doctor* does not.

### 2.5.1 EXPRESSION LANGUAGE

The expression language allows the formation of target expressions using logical conjunctive (&) and disjunctive (|) operators to arbitrary complexity. Target expressions are commutative and associative, so the expression

$$(\text{surgeon}|\text{doctor}) \& \text{dysphonia}$$

is logically equivalent to

```
(surgeon & dysphonia) |
(dysphonia & doctor)
```

and targets the same set of peers. The formal grammar for the target expression language is:

```
expression := factor (“&” factor | “|” factor)*
factor := attribute | “(” expression “)”
attribute := [a-zA-Z]+
```

At present the target expression is limited to specifying the attributes that need to be declared by a recipient (i.e. one cannot specify the absence of an attribute, for example), although this may be addressed in future work.

### 2.5.2 MAPPING EXPRESSIONS TO THE SURFACE

The address of a peer is directly related to its attributes, so a target expression can be converted to a *target*, a pattern that represents all possible addresses for peers matching the expression.

The expression is first rewritten as a logically equivalent disjunction of conjunctions. Each disjunctive term is then used to create a *target element* by hashing its keywords to a Bloom Filter in the same way an address is created. We use this filter as a template to find the addresses of all matching peers: i.e. the bits that are set in the target element must be set in the addresses of all matching peers. However, since any peer that has a matching address may have attributes in addition to those in the target, the unset bits in the filter are considered wildcards, able to match either 0s or 1s. For example, the target element 1?11?1 represents the address permutations 101101, 101111, 111101 and 111111.

A target,  $T$ , is simply a set of target elements, one for each disjunctive term.  $T$  is said to *match* ( $\bowtie$ ) a peer,  $p$ , if  $p^{addr}$  is a permutation of at least one of the target elements: i.e.  $T \bowtie p \iff \exists te \in T; te \& p^{addr} = te$ , where  $\&$  is the bitwise AND operator. Note that due to the use of Bloom Filters there is a certain probability of false positive matches.

Given a target, a peer can accurately calculate the extents on the Cartesian surface that it matches. Any peers that reside at addresses covered by those extents are likely (depending on the error rate of the Bloom Filters) to be members of the implicit group represented by the target. Furthermore, it is impossible for any peer that is a member to reside at a location not covered by these extents. With this knowledge, the peer can initiate a message that traverses all of the extents, either by visiting one after another, or branching into a tree-like structure. Section 3.4 describes our particular routing algorithm in more detail.

## 3 Algorithms

Several algorithms are needed to support the system. The ROUTE algorithm (section 3.1) forwards a message towards destination extents on the Cartesian surface and is used to direct messages as part of the joining and multicasting procedures. The JOIN algorithm (section 3.2) is used when a peer receives a JOIN request from a new peer and needs to partition its extents, and the LEAVE algorithm (section 3.3) is initiated by a peer when it wishes to leave the system. The CAST algorithm (section 3.4) delivers a multicast message to all peers matching a target expression.

### 3.1 ROUTE

The ROUTE algorithm (algorithm 1) forwards a message from the current peer towards any of the extents found in the set of clustered destination extents,  $C$ . It is used to route JOIN requests from bootstrap peers when new peers join the network (in which case  $C$  contains just a single destination extent — the address of the new peer), and to forward messages towards target extents when multicasting a message (section 3.4).

---

**Algorithm 1** The ROUTE forwarding algorithm.

---

**Require:**  $C$  is a set of destination extents.

**Require:**  $\forall e \in p^{ext}; \nexists c \in C, e > c \vee c > e$ .

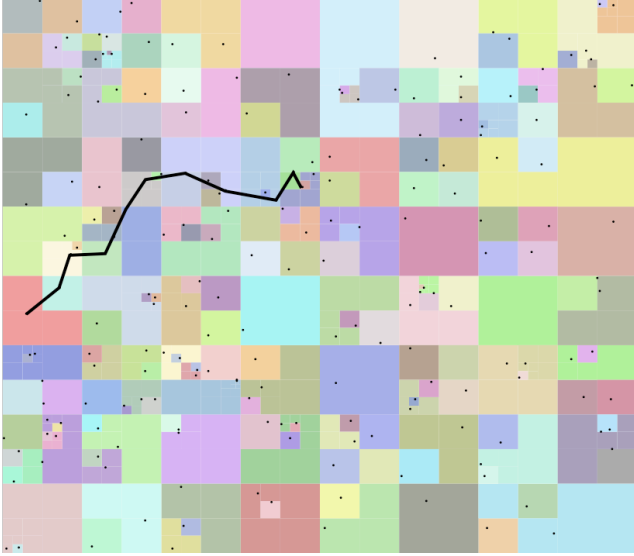
```
1:  $\Delta_{min} = \infty$ 
2: for all  $c \in C$  do
3:   for all  $(e, IP) \in p^{nbr}$  do
4:     if  $e > c \vee c > e$  then
5:       send( $msg, IP$ )
6:     return
7:   end if
8:   if  $\Delta(c, e, \delta) < \Delta_{min}$  then
9:      $\Delta_{min} \leftarrow \Delta(c, e, \delta)$ 
10:     $nextIP \leftarrow IP$ 
11:   end if
12: end for
13: end for
14: send( $msg, nextIP$ )
```

---

Various cell-forwarding criteria are possible. We have implemented a nearest neighbour criterion which preferentially forwards the message to any neighbouring extent that covers or is covered by a destination extent (lines 4–7), or failing this, to the neighbour that is nearest to any of the destinations in terms of Euclidean distance (lines 8–11). In algorithm 1, the function  $\Delta$  in lines 8 and 9 calculates the Euclidean distance between two extents on a surface of dimension  $\delta$ . Other more intelligent schemes are also possible. For example, attempting to minimise link load, avoiding known malicious peers, or passively monitoring the reliability

of neighbours over time in order to bias route selection.

Figure 4 shows how a ROUTE is forwarded across a populated 2-dimensional surface, beginning from the left and ending near the middle. At each step, the peer with the message calculates the distance from each of its neighbouring extents to the destination and forwards the message to the one that is closest.



**Figure 4. A ROUTE across a 2-dimensional surface.**

### 3.2 JOIN

The entire JOIN procedure actually includes several parts and potentially many peers. To join the network, a new peer connects to any arbitrary bootstrap peer that is already a member of the system. The means of finding a bootstrap peer is not part of the system model, but lists of potential peers could be published via a secondary system such as GWebCache<sup>7</sup>, for example.

Once connected, the new peer calculates its address by hashing its attributes (section 2.3), then uses the ROUTE algorithm (section 3.1) to forward a JOIN request to the peer that currently manages that extent, beginning from the bootstrap peer (which has nothing more to do with the procedure). Upon receipt of the JOIN request, the existing peer executes algorithm 2.

First it calculates the extent that can be partitioned to the new peer by finding its largest extent containing the new peer's address, but not its own (lines 1–16). In the case where the two peers share the same address, this is the *null* extent, meaning the new peer will not

---

#### Algorithm 2 Handling a JOIN request.

---

**Require:**  $msg^{addr} \neq p^{addr}$  and  $\exists e \in p^{ext}; e > msg^{addr}$

- 1: {Find the new peer's extent,  $newExt$ .}
- 2:  $remaining \leftarrow p^{ext}$
- 3: **loop**
- 4:  $e \leftarrow remaining.pop$
- 5: **if**  $e > msg^{addr}$  **then**
- 6:  $p^{ext} \leftarrow p^{ext} - e$
- 7: **if**  $e > p^{addr}$  **then**
- 8: {divide( $e$ ) returns the  $2^d$  sub-extents of  $e$ .}
- 9:  $remaining \leftarrow remaining \cup divide(e)$
- 10:  $p^{ext} \leftarrow p^{ext} \cup divide(e)$
- 11: **else**
- 12:  $newExt \leftarrow e$
- 13: **break**
- 14: **end if**
- 15: **end if**
- 16: **end loop**
- 17: {Tell old neighbours our current extents.}
- 18:  $update \leftarrow Update.new$
- 19:  $(update^{ext}, update^{senderIP}) \leftarrow (p^{ext}, p^{IP})$
- 20: **for all**  $nIP \in \{IP; \forall (e, IP) \in p^{nbr}\}$  **do**
- 21:  $send(update, nIP)$
- 22: **end for**
- 23: {Inform the new peer of their extent.}
- 24:  $ack \leftarrow Ack.new$
- 25:  $ack^{ext} \leftarrow \{newExt\}$
- 26:  $ack^{nbr} \leftarrow \{(e, IP) \in p^{nbr}; e \sim newExt\} \cup \{(e, p^{IP}); e \in p^{ext}, e \sim newExt\}$
- 27:  $send(ack, msg^{src})$
- 28: {Update our list of neighbours.}
- 29:  $p^{nbr} \leftarrow \{(e_1, IP) \in p^{nbr}; \exists e_2 \in p^{ext}, e_1 \sim e_2\}$
- 30: **if**  $\exists e \in p^{ext}; e \sim newExt$  **then**
- 31:  $p^{nbr} \leftarrow p^{nbr} + (newExt, msg^{srcIP})$
- 32: **end if**

---

be managing any part of the surface (but will receive duplicates of CAST messages received by the first peer at that address). After the partitioned extent has been calculated the peer informs its neighbours of its updated set of extents (which can change if the previous part of the algorithm resulted in some extents being subdivided) in lines 17–22. It then directly acknowledges the new peer's request to JOIN (lines 23–27), whereupon the new peer takes its place as manager of that extent. Finally, the peer updates its neighbour table to reflect the new configuration (i.e. it prunes old neighbours which are no longer adjacent, and adds the new peer if it is adjacent) in lines 28–32. Figure 3 shows a 2-dimensional surface evolving as new peers join (see section 2.4 for a description of this figure).

<sup>7</sup><http://www.gnucleus.com/gwebcache/>

### 3.3 LEAVE

A peer can leave the network gracefully by asking one or more of its neighbours (or preferably a cohabitant if one exists) to take ownership of some or all of its current extents, and informing all neighbours of the change.

At present, the model does not handle unexpected disconnections, though this is clearly required for a robust system. Section 5.1 discusses this in more detail.

### 3.4 CAST

The CAST algorithm (algorithm 3) multicasts a message from a source peer to the implicit group of peers matching a target expression. The source peer first converts the expression to a target (section 2.5), and constructs a CAST packet comprising the payload, target ( $msg^{target}$ ) and a list of extents within the system that have not yet received the message ( $msg^{unexplored}$ ). Initially, this is set to the complete list of top-level extents on the surface, but extents are subdivided and removed as more of the surface receives the message. Copies of the message are then sent to one or more neighbours who in turn deal with the message in the same way.

---

**Algorithm 3** The CAST forwarding algorithm.

---

```

1: if  $msg^{target} \bowtie p$  then
2:   {Inform application of CAST payload.}
3: end if
4:  $msg^{unexplored} \leftarrow msg^{unexplored} \ominus p^{ext}$ 
5:  $msg^{unexplored} \leftarrow msg^{unexplored} \cap msg^{target.extents}$ 
6: if  $msg^{unexplored} \neq \emptyset$  then
7:    $\{C\} \leftarrow cluster(msg^{unexplored}, \phi)$ 
8:   for all  $C \in \{C\}$  do
9:      $nmsg \leftarrow msg$ 
10:     $nmsg^{unexplored} \leftarrow C$ 
11:    route( $nmsg, C$ )
12:   end for
13: end if

```

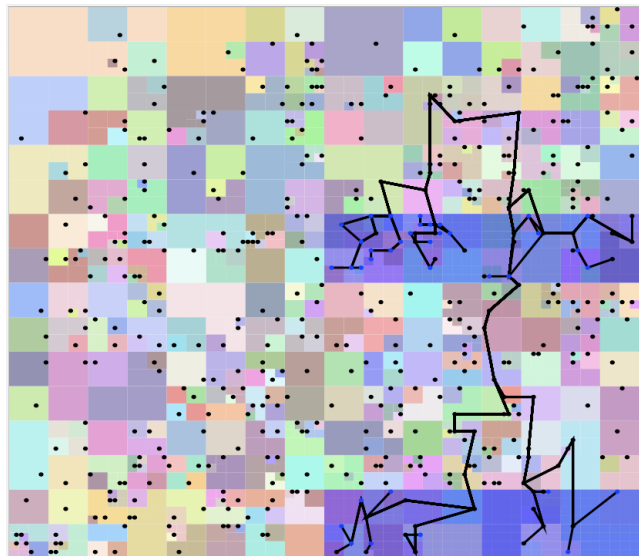
---

The algorithm analyses a CAST packet and determines to which neighbours it should be forwarded. A peer starts by first checking whether the message is intended for itself (i.e. the target matches its attributes) and if so, delivers the payload to the applications it is hosting (line 1–3). Next, the set of extents managed by the peer are removed from  $msg^{unexplored}$  since these have now been explored (line 4). Line 5 removes from  $msg^{unexplored}$  all extents that do not intersect with the target such that only foreign extents that intersect with the target that have not yet been explored remain. The algorithm terminates if this set is empty since there are guaranteed to be no other peers that should receive the

CAST (line 6).

The peer then clusters the remaining unexplored extents using a hierarchical top-down divisive cluster<sup>8</sup> (line 7). In order to minimise the number of identical messages sent along a route, the distance metric used in the clustering is the angular difference between the extents as measured from the current peer’s Cartesian coordinates.

Intuitively, if multiple unexplored extents lie in the same direction away from the current peer (and hence have minimal angular differences), then only one message needs to be sent in that direction. As the message gets closer to the extents, the angular differences will become more marked, and once they exceed a threshold the message can be cloned and the route can branch as necessary.



**Figure 5.** An illustrative 2-dimensional CAST with  $\phi = 1.0$ . The source is at the top right of the surface and the targeted region is shown shaded.

The *branch factor*,  $\phi$ , is used as the threshold for the clustering, and can thus take a value  $0^c \leq \phi^c \leq 2\pi^c$ . A low value results in many clusters (which will likely result in many message replicas traversing the same overlay and physical links), while a high value results in few clusters (which may necessitate excessive message backtracking or path crossing). The optimal value for  $\phi$  likely depends on the nature of the network, though

<sup>8</sup>A divisive cluster begins with a single cluster and compares each pair of elements to find the two which are maximally distant. If this distance is greater than a threshold, the cluster is recursively subdivided, terminating once no two elements exceeds the threshold.

validating this is currently left for future work. Once the clusters have been found, the message is cloned (line 9) and routed towards them (line 11) using the ROUTE algorithm (section 3.1). Each cloned message is given only the unexplored extents for that cluster (line 10) which, being mutually exclusive, means that the various branches each explore and terminate in distinct regions of the surface. To illustrate this, figure 5 shows the path of a CAST message from the top right corner of the surface to the shaded target extents using a branch factor of 1.0 (which produces relatively few branches).

## 4 Related work

Multicasting is a well-studied problem in networking. IP multicast<sup>9</sup> allows hosts to explicitly join a group and receive all messages sent to the group address. A spanning tree of all members is constructed that can be source-rooted in the case of a single producer, or rooted at a rendezvous point when there are multiple producers. An advantage of multicasting at this level in the network stack is that duplicate packets can be eliminated, optimising the usage of physical network links. Similar application-level approaches have also been explored [12, 8], that although unable to offer network utilisation as optimal as IP multicast have the advantage of not requiring additional network support. However, none of these schemes effectively address the problem of implicit group messaging as they require the creation and maintenance of a separate multicast group for each implicit group (which may potentially be used just once or never) and offer no inherent mechanisms for discovering the explicit hosts that should participate.

There have been a number of multicast schemes constructed over distributed hash tables, such as Scribe [6] (based on the Pastry DHT [24]), CAN multicast [22] over the CAN DHT [21] and Bayeux [30] (on the Tapestry DHT [29]). Again, these use the notion of explicit groups: a group is created by a peer and may be joined by other peers. Any message sent to the group is delivered to all members of the group. This approach has the same drawbacks as traditional multicast when used for implicit group messaging.

The concept of implicit groups has formed a part of some systems. The PEACH project [4] uses the concept of “implicit organization” in order to communicate between various components of an interactive museum guide and Chambel et al [7] describe the use of implicit groups to aid navigation of and retrieval of data from large distributed “hyperbases”. However these approaches are limited to their specific application and

<sup>9</sup><http://www.ietf.org/rfc/rfc3170.txt>

are not directly applicable to large distributed P2P networks over the Internet.

Publish/subscribe messaging has two variations: subject-based and content-based. Subject-based systems such as Tibco Rendezvous<sup>10</sup> allow subscribers to join channels in much the same way as hosts can join multicast groups to receive all messages sent to the group. In the content-based variation such as Elvin [26], subscribers register a subscription with the system indicating the type of message they would like to receive. When a producer publishes a message it is compared to the set of subscriptions and forwarded to only those hosts that have registered a matching subscription.

Content-based publish/subscribe (C-BP/S) is conceptually similar to implicit group messaging in that messages are delivered to a group of recipients based on a match made at the time of publication. The primary difference between C-BP/S and implicit group messaging is by whom the group is specified: in the case of implicit group messaging, the publisher of the message selects the type of recipient but in C-BP/S the consumers select the type of message. These converse semantics lead to differing expressiveness between the two approaches and each is suited to different classes of applications.

C-BP/S is well-suited to applications where consumers know the sorts of messages that will be published and need to be able to listen for specific events without tying themselves to particular publishers: for example, a GUI component in a distributed application can listen for updates to the data model from any other component and display them to the user as they occur. In this case, the publisher does not care how the information is handled or whether a GUI is updated - it is simply making the information available to components that may wish to make use of it. Implicit group messaging is better suited to applications where publishers are trying to communicate with a certain class of consumer. In this case, the consumer has no knowledge of what kind of publishers or messages exist in the system, but is still able to receive messages of interest. This is more suitable for use in large many-to-many communities such as P2P networks.

There have been systems that decentralise content-based publish/subscribe over wide area networks. Siena [5] provides an overlay network of event brokers that allow hosts to subscribe to events generated anywhere in the network. These subscriptions are replicated among the brokers, or combined into “covering subscriptions”. Thus, a form of multicast tree can be traversed each time an event is generated, and only de-

<sup>10</sup><http://www.tibco.com/>

livered along branches that contain subscribers. In very large networks, it is preferable not to have bottlenecks responsible for routing much of the overall traffic of the system as they typically require significant investment in bandwidth and processing power and represent single points of failure or targets for attack. As such, a completely distributed approach is desirable, where every participant is equally responsible for routing messages. Unfortunately, this makes the use of covering subscriptions far more difficult.

Mirinae [9] is a content-based publish/subscribe system that uses a hypercube overlay routing network to group similar subscriptions. Subscriptions are mapped to a corner of the hypercube based upon an application schema, and events are mapped to partial identifiers according to the same schema and sent along the edges to corners that cover it. This scheme works because subscriptions that are semantically close can be made close in the topology of the hypercube. Since this topology can be constructed before any events are published, high routing efficiency can be achieved. However, this approach is not directly applicable to implicit group messaging, since the peers forming an implicit group are potentially different for each message and hence cannot be intentionally placed near one another ahead of time.

Interest management in war game simulations and virtual environments [28, 17] refers to the distribution of update messages from entities in the environment to all other entities that need to know about them. Usually, an entity creates an expression of interest in events created within a certain virtual distance of it, in a similar manner to content-based publish/subscribe. An additional feature of interest management however is that extrinsic attributes of the publisher of a message (such as their logical position, or name) can be matched in addition to the intrinsic attributes of the message.

Khambatti [13] has researched the discovery and utilisation of interest-based communities in peer-to-peer networks for directing distributed searches. His approach connects peers with common interests in an overlay network, but does not specifically support the boolean expressions of group attributes necessary for implicit group messaging.

## 5 Conclusion

We have described the concept of implicit group messaging, whereby publishers of a message are able to specify the characteristics of the group of recipients, rather than explicitly name them. We have identified seven properties that should be part of an ideal implicit group messaging model and presented a novel model that satisfies many of them, particularly the pri-

mary properties of guaranteeing delivery to all implicit group members while minimising participation of non-group members, and distributing the responsibilities for system operation across all nodes. Finally, we have outlined necessary algorithms for the model.

### 5.1 Future work

The model presented in section 2, is sufficient and complete when peers are reliably connected and functioning after they JOIN. For more realistic conditions however, the existing algorithms are somewhat brittle, and do not handle unexpected disconnection of peers. We expect that a similar approach to that used in the CAN DHT [21] will be applicable for remedying this. Additionally, the CAST algorithm will need to be modified to either route around holes in the network caused by disconnected nodes, or automatically repair such holes as they are encountered. Closely related to this is the need to improve the handling of simultaneous JOIN requests.

The most pressing work however is to evaluate the overlay network model over a realistic physical network in order to evaluate the fundamental properties of the approach. We have implemented a simulator in OMNeT++<sup>11</sup> using hierarchical internetwork topologies as produced by the BRITe<sup>12</sup> topology generator and are currently comparing the model to several alternative approaches using metrics that correspond to the ideal properties in figure 1. Once this fundamental simulation is complete, we intend to experiment with altering model variables such as the attribute distribution over peers and the dimensionality of the surface.

Some of the ideal properties (figure 1) are not adequately supported by the current model, particularly the “replayable” message (5) and fairly distributed workload (7) properties. Property 5 can be implemented by having peers cache messages they receive for a short time. A new peer can ask for copies of these by CASTing a request to peers sharing their attributes. Property 7 is less than ideally supported in the current model, since the order in which peers JOINs the network in part determines which extents a peer manages and it is possible for older peers to be managing large disjoint extents of the surface. This leads to unfair loading of these peers, since more paths over the surface are likely to pass through them. The problem can be mitigated by having the peers LEAVE the network, and then reJOIN, similarly to how Mirinae [9] rebalances its peers. By LEAVEing the network, the peers hand off their extents to each neighbour most suited, and by reJOINing they will be assigned just the single

---

<sup>11</sup><http://www.omnetpp.org/>

<sup>12</sup><http://www.cs.bu.edu/brite/>

extent covering their address, thus spreading their old extents among their neighbours. It is important that this “reshuffling” does not occur too often as there is a small, but not insignificant cost for LEAVEing and JOINing.

Some extensions to the model are also planned, including random shortcuts through the surface between peers to reduce routing cost. These can be inserted when peers join the network at no extra cost by linking back to one of the peers encountered on the initial ROUTE from the bootstrap peer. These random links offer the potential to greatly reduce the diameter of the network without sacrificing much additional storage or communication overhead.

We suspect that the base model presented in this paper will suffer significant latency when measured over a physical network since the overlay address of a peer by its nature cannot correspond particularly well to the underlying network. However, by prefixing addresses with several bits, we may be able to place peers on the surface with some approximation to the underlying network, while still retaining the other bits for attribute matching. This technique may reduce the number of physical network hops a peer typically covers when traversing an overlay hop.

There are some privacy and security implications with a system such as this. Since groups are implicit and consumers do not explicitly request messages, the system is arguably open to abuse by malicious message producers: for instance spammers looking to send large amounts of advertising material. It can be countered however that such a system allows advertisements to be far more targeted, thereby benefitting both producers and consumers. If spammers were to abuse the system by directing messages to very large implicit groups or all peers, spam filters could be installed at the client peers. Combatting spam within the network itself may also be possible: peers asked to route a very generally CAST message may choose to silently drop it.

Similar to this problem is the ability of the system to cope with malicious peers (peers that seek to disrupt the system by not forwarding messages, or forwarding them incorrectly). These problems can be somewhat mitigated by using different values of the branch factor,  $\phi$ , for CASTing messages (thereby taking different routes to recipient peers), or introducing some amount of randomness or redundancy in the routing decisions.

Future possible extensions of the general implicit group messaging concept may include the combination of the core overlay network with mobile fringe nodes connecting either through wireless access points directly or with mobile ad hoc networks [2]. Some interesting applications are conceivable, for example, the

delivery of interesting new music to people jogging with portable, wireless music devices, and connecting doctors around a hospital to implicit groups of their colleagues via wireless PDAs.

The model and approach may also have some applicability to information retrieval problems. By routing queries to implicit groups in large decentralised databases or search engines, more accurate and precise results may be achievable.

## Acknowledgment

The authors would like to acknowledge the ongoing support of the Smart Internet CRC and the residents of G61b at the University of Sydney, particularly Adam Hudson, for many valuable discussions.

## References

- [1] L. A. Adamic and B. A. Huberman. Zipfs law and the internet. *Glottometrics*, 3:143–150, 2002.
- [2] I. F. Akyildiz and X. Wang. A survey on wireless mesh networks. *Communications Magazine, IEEE*, 43(9):S23–S30, 2005.
- [3] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.
- [4] P. Busetta, M. Merzi, S. Rossi, and M. Zancanaro. Group communication for real-time role coordination and ambient intelligence. <http://citeseer.ist.psu.edu/busetta03group.html>, 2003.
- [5] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Trans. Comput. Syst.*, 19(3):332–383, 2001.
- [6] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications (JSAC)*, 20(8), October 2002.
- [7] T. Chambel, C. Moreno, N. Guimaraes, and P. Antunes. *Concepts and Architecture for Loosely Coupled Integration of Hyperbases*, volume 3 of *Broadcast Technical Report Series, ISSN: 1350-2042*, chapter 4. Broadcast Secretariat, Department of Computing Science, University of Newcastle-upon-Tyne, UK, 1994.
- [8] Y. Chawathe, S. McCanne, and E. Brewer. An architecture for internet content distribution as an infrastructure service. <http://citeseer.ist.psu.edu/chawathe00architecture.html>, February 2000.
- [9] Y. Choi and D. Park. Mirinae: A peer-to-peer overlay network for large-scale content-based publish/subscribe systems. In *NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*, pages 105–110, New York, NY, USA, 2005. ACM Press.

- [10] K. G. Coffman and A. M. Odlyzko. *Internet growth: is there a "Moore's law" for data traffic?*, volume Handbook of massive data sets, pages 47–93. Kluwer Academic Publishers, Norwell, MA, USA, 2002. ISBN: 1-4020-0489-3.
- [11] S. Golder and B. A. Huberman. The structure of collaborative tagging systems. Online at Information Dynamics Laboratory, HP Labs, August 2005. To appear in *Journal of Information Science* (2006).
- [12] Y. hua Chu, S. G. Rao, and H. Zhang. A case for end system multicast (keynote address). In *SIGMETRICS '00: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 1–12, New York, NY, USA, 2000. ACM Press.
- [13] M. Khambatti. *Peer-to-peer communities: architecture, information and trust management*. PhD thesis, Arizona State University, December 2003.
- [14] R. Kumar, J. Novak, P. Raghavan, and A. Tomkins. On the bursty evolution of blogspace. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 568–576, New York, NY, USA, 2003. ACM Press.
- [15] M. Light and M. T. Maybury. Personalized multimedia information access. *Communications of the ACM*, 45(5):54–59, May 2002.
- [16] C. Lindahl and E. Blount. Weblogs: simplifying web publishing. *Computer*, 36(11):114–116, November 2003.
- [17] K. L. Morse. Interest management in large-scale distributed simulations. Technical Report ICS-TR-96-27, Department of Information & Computer Science, University of California, Irvine, 1996.
- [18] B. A. Nardi, D. J. Schiano, and M. Gumbrecht. Blogging as social activity, or, would you let 900 million people read your diary? In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 222–231, New York, NY, USA, 2004. ACM Press.
- [19] Netcraft. July 2005 web server survey. [http://news.netcraft.com/archives/2005/07/01/july\\_2005\\_web\\_server\\_survey.html](http://news.netcraft.com/archives/2005/07/01/july_2005_web_server_survey.html), July 2005.
- [20] J. A. Orenstein. Multidimensional tries used for associative searching. *Inf. Process. Lett.*, 14(4):150–157, 1982.
- [21] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *SIGCOMM'01, San Diego, California, USA*, Berkeley, CA, August 2001. ACM.
- [22] S. Ratnasamy, M. Handley, R. M. Karp, and S. Shenker. Application-level multicast using content-addressable networks. In *NGC '01: Proceedings of the Third International COST264 Workshop on Networked Group Communication*, pages 14–29, London, UK, 2001. Springer-Verlag.
- [23] S. C. Rhea and J. Kubiawicz. Probabilistic location and routing. In *Proceedings of INFOCOM 2002*, 2002.
- [24] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001.
- [25] N. Sarshar, P. O. Boykin, and V. P. Roychowdhury. Percolation search in power law networks: Making unstructured peer-to-peer networks scalable. In *4th International Conference on Peer-to-Peer Computing (P2P 2004)*, Zurich, Switzerland, pages 2–9. IEEE Computer Society, August 2004.
- [26] B. Segall and D. Arnold. Elvin has left the building: A publish/subscribe notification service with quenching. In *Proceedings AUUG97, Brisbane, Australia, September 1997*. Distributed Systems Technology Centre, University of Queensland, Australia, September 1997.
- [27] B. Yang and H. Garcia-Molina. Efficient search in peer-to-peer networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, July 2002.
- [28] A. P. Yu and S. T. Vuong. Mopar: a mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games. In *NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*, pages 99–104, New York, NY, USA, 2005. ACM Press.
- [29] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiawicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, 2004.
- [30] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiawicz. Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination. In *NOSSDAV '01: Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*, pages 11–20, New York, NY, USA, 2001. ACM Press.

School of Information Technologies  
Madsen Building F09  
University of Sydney NSW 2006 AUSTRALIA  
T: +61 2 9351 4917 F: +61 2 9351 3838  
W: [www.alumni.it.usyd.edu.au](http://www.alumni.it.usyd.edu.au)

ISBN 1 86487 804 5