Peer-to-Peer Networking for Discovering Programmable Resources

Paul Smith
p.smith@comp.lancs.ac.uk

Steven Simpson
ss@comp.lancs.ac.uk

David Hutchison
dh@comp.lancs.ac.uk

Computing Department
Lancaster University
Lancaster, LA1 4YR, UK

ABSTRACT

Some forms of programmable networks such as funnelWeb allow service components to be deployed discretely (i.e. out-of-band) on a suitable configuration of elements, but do not define mechanisms to determine such configurations.

We present a mechanism to resolve arbitrary service-specific deployment constraints into a suitable node configuration. To focus constraint resolution, we arrange programmable elements into an overlay, and use this to interpolate/extrapolate more favourable locations to search. Programmable service components are used to evaluate suitability of individual nodes.

1. INTRODUCTION

Programmable networks enable the rapid deployment of novel services. This is achieved by deploying programmable service components onto *general purpose* programmable network elements. When considering programmable service deployment, two main approaches have been adopted to date: the capsule (or in-band) approach, where computational components are included with or referenced from within network traffic that is to be augmented [14, 13]; or the discrete (or out-of-band) approach, where computational elements are loaded prior to the traffic's arrival by a third-party process [3, 11]. The discrete approach requires a process to select the appropriate configuration of programmable elements to deploy components on. However, mechanisms to determine such configurations remain undefined. Arbitrary service-specific deployment constraints define which element configurations are acceptable. We present a mechanism to resolve such constraints into a suitable node configuration.

Scalability and availability are prevalent properties of peer-to-peer networking based services. This has been demonstrated to be the case in a number of widely deployed resource sharing services used in the Internet today [6, 2]. The mechanism we present in this paper adopts the peer-to-peer networking paradigm in order to inherit these properties.

In this paper we present an approach to programmable resource discovery which arranges elements into a meshed overlay network. A property of the overlay construction enables us to interpolate/extrapolate more favourable locations for searching, removing the necessity to flood the network with constraint resolution activity. Due to the arbitrary nature of service deployment constraints, programmable service components are used to evaluate the suitability of individual nodes.

The paper is arranged as follows. In Section 2 an introduction to service deployment constraints in a programmable networking environment is given. Also in Section 2, our approach to programmable resource discovery will be presented along with an example deployment scenario. Following this in Section 3 we describe our approach to mesh construction called `GROUPNET`. Section 4 presents an evaluation of `GROUPNET`, demonstrating its suitability and scalability properties. Related work are presented in Section 5 and conclusions are drawn in Section 6.

2. PROGRAMMABLE SERVICE DEPLOYMENT

A significant challenge in programmable service deployment is to take a set of service-specific deployment constraints and resolve them into a suitable set of programmable elements on which service components can be deployed. Peer-to-peer networking services such as Gnutella perform single-dimensional constraint resolution. For example, they transform the name of a shared file into a set of peers from which the file can be obtained. In programmable networks there are potentially multiple service deployment constraint dimensions that can be distinct on a per-service-instantiation basis, for example the monetary cost of a service and the required network bandwidth. In some cases, values within constraint dimensions can be traded against others in different dimensions – for example, a customer may be willing to pay a price in proportion to the service quality.

A large class of service have applicable constraint dimensions where the values within a dimension vary approximately in relation to network location. For example, the cost of a telephone call (an example constraint dimension) may cost less when the end points are nearer and progressively get higher as the end points become distant. Programmable services that demonstrate these properties can be seen at [4, 12].

NGC '02 Boston, Massachusetts, USA

2.1 Programmable Resource Discovery

To focus locations to search for suitable configurations of programmable elements, we take advantage of the property that values within some constraint dimensions vary roughly in relation to network location. To do this we construct a *topologically aware* overlay mesh out of the available programmable resources using our `GROUPNET` algorithm presented in Section 3. For searching to work, as one traverses the overlay mesh, values within a topologically varying dimension should change gradually. Furthermore, relatively large movement on the overlay should yield a relatively large change of values within a dimension and conversely smaller hops result in a smaller change. Given these properties, better locations to search for programmable resources can be interpolated or extrapolated.

Programmable components are used to evaluate the suitability of individual programmable elements. An element that is intended to support a number of potentially transient services cannot natively understand the nature of their operational constraints. It is for this reason we advocate the use of programmable service components. In this situation, elements expose interfaces that enable appropriately privileged programmable resource discovery components to interrogate local state and perform measurements with interesting nodes. Constraint resolution intelligence resides in programmable service components that interrogate *dumb* programmable elements.

2.2 Service Deployment Scenario

To demonstrate the applicability of our approach we will present an example service deployment scenario. The service we present in this example aims to reduce the cost of making long distance calls by using the Internet and Voice over IP (VoIP) to bridge the long distance connection [5]. At the remote end of a connection, a network element with a modem attached to the PSTN can be used to complete a call to a local receiver, therefore making the total cost of a long distance call at most two local connections: one to connect to the Internet and the other to a local receiver. In our example scenario, the network element with modem functionality is programmable and the aim is to discover from the set of elements available a subset most suited to serve our application demands.

By using the Internet to form the long distance part of a connection, one exacerbates the problem of round-trip call delay. However, the greater part the Internet constitutes the connection, the cheaper the cost of the call is likely to be. A balance must be struck between the total cost of the call and the delay experienced. Both parts of a connection contribute to a monetary cost and the total round-trip delay. We can use the function $c = c_{\mathrm{n}} + c_{\mathrm{p}}$ to calculate the total cost of making the call, where c_{n} is the fixed cost of connecting to the Internet and c_{p} is cost of the call on the PSTN. We use $t = t_{\mathrm{n}} + t_{\mathrm{p}}$ to determine the total round-trip delay, where t_{n} is the delay across the Internet and t_{p} is the delay across the PSTN. The values of t and c have acceptable upper bounds represented by $t_{\max}$ and $c_{\max}$ respectively. Elements with values greater than $t_{\max}$ or $c_{\max}$ can be immediately dismissed as unacceptable. The product of t and c can be used to select the most appropriate element from the available set. Table 1 shows example values of five candidate elements.

If $t_{\max} = 100$ms and $c_{\max} = 10$p/min we can immediately consider elements 1, 2 and 5 from Table 1 as unsuitable. Elements 3 and 4 meet our constraints and we consider the value of tc to select the most a appropriate element, in this case element 4 best meets the service constraints.

Element	t_{n}	t_{p}	c_{p}	c_{n}	t	c	tc
1	20	17	17	3	37	19	703
2	40	14	12	3	54	15	810
3	60	11	6	3	71	9	639
4	80	8	3	3	88	6	528
5	100	5	3	3	105	6	630

Table 1: Example costs for a set of candidate elements for deploying a Voice over IP service

By constructing the overlay of programmable elements using `GROUPNET`, there is a correlation between relative location in the overlay and network topology. One would anticipate the cost of the call c_{p} to change in relation to network topology. Therefore, traversing the overlay network should result in an approximate variation in the cost c_{p}. We can use this property to interpolate/extrapolate better locations for searching. For example, if a direction along the overlay yields elements with a better c_{p} than those already known, searching can continue in that direction in the hope that c_{p} will continue to improve. Other inferences such as these can be made in order to find the set of programmable elements that have acceptable values of c. If such elements are found, t_{n} can be determined and ultimately tc. Other constraint dimensions, such as available bandwidth at an element can be determined using programmable service components.

3. GROUPNET: A GROUPING MESHED OVERLAY NETWORK

To focus constraint resolution, we arrange programmable elements into an overlay, and use this to interpolate/extrapolate more favourable locations for searching. We achieve this by constructing the overlay so that an element's immediate peers are more akin to itself than those further hops away. This has the effect of *grouping* peers into regions of the overlay.

3.1 GROUPNET Overlay Construction

The procedure that a new peer wishing to join a `GROUPNET` mesh must follow can be described as follows. A peer s locally decides its desired degree d, which will influence the number of immediate peers it has. It will maintain $N(s)$, its set of immediate neighbours. (Initially, using a bootstrap mechanism, s is assigned a set of peers $N(s)$. This initial $N(s)$ can either be randomly selected, or to improve algorithm performance some heuristics can be used for selection.) s now undergoes a sequence of optimisations.

In each optimisation, for each peer p in $N(s)$, s obtains measurements to p and $N(p)$, and orders them to produce a list $L(p)$ (which includes p) of those peers in order. The new set of neighbours of s, $N'(s)$ will at least consist of the best of each $L(p)$ – if this falls short of d, the best of the union of the remaining members of all the $L(p)$'s can make up the shortfall.

The result is that for each $p \in N(s)$, $(N'(p) \cup p) \cap N'(s)$ is not empty, i.e. there is a path of no more than two hops between s and each of its original neighbours, ensuring that

the mesh does not become disconnected. We envisage the algorithm would be applied continuously at a node to start in order to reach some desired level of connectivity and then periodically afterwards for maintenance purposes.

3.2 Peer Membership Example

Figure 1 shows an example of a peer wishing to join a mesh and some of the steps toward achieving this. The node s wishes to join the mesh and has selected a degree d of 3. Using a bootstrap mechanism s is assigned $\{K\}$ as its initial $N(s)$ as shown in step 1 of Figure 1. The neighbours of K are evaluated along with K itself and the ordered list $L(K)$ is produced. In this example, peers closer to s score higher than those more distant and therefore appear toward the start of the list. From $L(K)$, the best d are taken to produce $N'(s)$ as shown in step 2. This completes a single iteration of the `GROUPNET` membership algorithm.

We anticipate a number of iterations would be performed until $N'(s)$ does not improve on $N(s)$. In step 2 of Figure 1, $N'(s) = \{O, L, K\}$ which improves upon the immediate neighbours of s. Another iteration of the membership algorithm involves: determining $N(p)$ where p is a peer in $N'(s)$ and $N(p)$ are all the neighbours of p (for example $N(O) = \{K, P\}$), evaluating them to create $L(p)$ ($L(O) =< P, O, K >$) and then selecting the best from each $L(p)$ to generate $N''(s)$. If we find that we have already selected the best from a list, we simply take the next best as demonstrated in the example. It can be seen that the immediate peers of s draw gradually closer to it. Ultimately, the algorithm will terminate with $N(s) = \{U, T, Y\}$.

3.3 Peer Evaluation Metrics

The metric used to evaluate a peer will depend on the application of the overlay mesh. For example, for programmable resource discovery we wish to construct a topologically aware overlay network. To this end, we could use round-trip delay between peers to evaluate suitability. Peers with smaller round trip delays score higher in this situation that those with longer delays.

If one considers a resource sharing application, it may be desirable to group peers sharing similar resources. By grouping peers which share similar resources (for example by music from a similar genre) searching could be more effective as peers probable to have a desirable resource will be fewer overlay hops away. If a user searches for resources that are distinct from those available from its immediate peers, inferences can be made regarding optimal locations to search.

4. EVALUATION OF GROUPNET

We have simulated the `GROUPNET` overlay construction algorithm. Peers are randomly distributed in a two-dimensional coordinate space and the group membership algorithm run at each peer. Distance between peers is used as the configuration metric. Pairs of peers with a smaller distance score better than those with larger distances. Figure 2 presents a graphical representation of the meshes produced. Node degrees of 3, 4 and 5 were assigned shown in Figure 2a–c respectively. The top line of Figure 2 (*random*) shows meshes produced after peers are assigned a random neighbour. The bottom line of Figure 2 (*groupnet*) show the meshes produced after running the `GROUPNET` overlay membership algorithm presented in Section 3. To optimise the

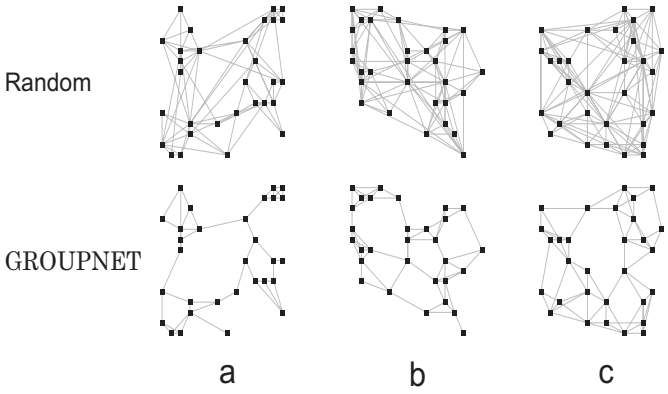


Figure 2: Meshes created both randomly and using the `GROUPNET` algorithm with different node degrees.

mesh produced, we ran the algorithm 30 times at each node[1].

The aim of these simulations is to demonstrate that the membership algorithm produces overlays with desirable properties. The random meshes shown in Figure 2 show undesirable properties as peers have links that are connected to other relatively distant peers. The searching mechanism described in Section 2 is less effective when conducted on overlay configurations such as these, since there is no correlation between distances on the overlay and the space. The meshes constructed using the `GROUPNET` algorithm have the desired properties for enabling interpolation/extrapolation as there is a smooth transition over the space as one traverses the overlay.

4.1 Suitability of GROUPNET

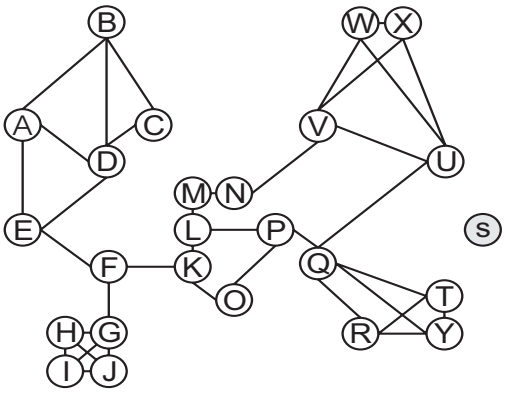
To perform the searching described in Section 2 there needs to be a correlation between the underlying network topology and the overlay mesh. The aim of these simulations is to evaluate to what degree `GROUPNET` achieves this.

Initially, we constructed random and `GROUPNET` meshes in the same manner used to produce Figure 2. A node was then selected on the overlay and a number of hops on the overlay moved from that point. The next hop was chosen by randomly selecting from a node's immediate peers. At exponentially increasing numbers of hops, distances to the originating node were measured. A variant of bloom filters [1] was used to reduce the probability of moving to a previously visited node. Simulations were also run without the use of bloom filters. Figure 3 shows the results of these simulations.

We anticipate that a relatively small number of hops traversed on the `GROUPNET` overlay would result in moving a relatively small distance in space. Similarly, a larger number of hops traversed should result in a greater movement in space. This behaviour is shown in the bottom two lines of Figure 3. The top two lines of Figure 3 show how there is little correlation between overlay and space movement on the random mesh. Note also how the use of bloom filters increases the distance travelled in space, but does not effect the overall trend for smaller numbers of hops.

A second desirable quality of `GROUPNET` is that its mem-

[1]Normally, we expect that optimisation would happen periodically, increasing in frequency the more excess links a peer has.



$d = 3$

1. $N(s) = \{K\}$

$N(K) = \{F, O, L\}$, $L(K) =< O, L, K, F >$

2. $N'(s) = \{O, L, K\}$

$N(O) = \{K, P\}$, $L(O) =< P, O, K >$

$N(K) = \{F, L, O\}$, $L(K) =< O, L, K, F >$

$N(L) = \{K, P, M\}$, $L(L) =< P, L, M, K >$

3. $N''(s) = \{P, O, L\}$

$N(P) = \{L, O, Q\}$, $L(P) =< Q, P, O, L >$

$N(O) = \{K, P\}$, $L(O) =< P, O, K >$

$N(L) = \{K, P, M\}$, $L(L) =< P, L, M, K >$

4. $N'''(s) = \{Q, P, O\}$

Figure 1: An example peer membership scenario with some of the stages of evaluation shown

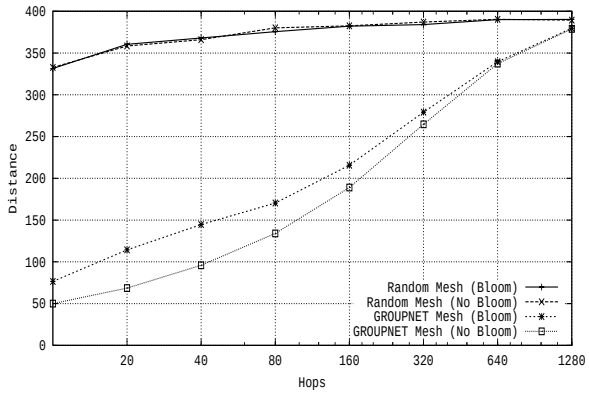


Figure 3: Distance covered versus the number of overlay hops

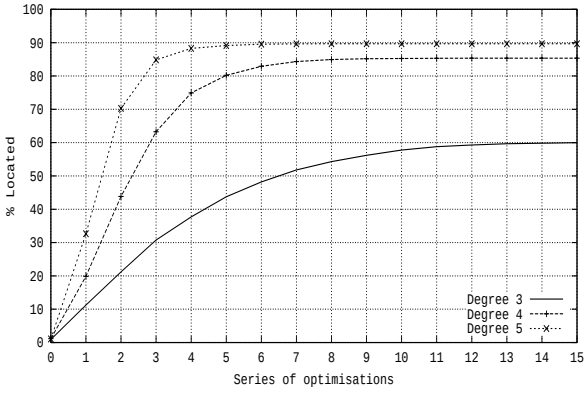


Figure 4: Percentage of the closest nodes found after a series of iterations of the GROUPNET membership algorithm

bership algorithm should converge to a node's d closest peers in space being its immediate d peers on the overlay, where d is the desired degree. We conducted simulations to determine what percentage of the closest nodes available were found by GROUPNET. To do this, we assumed global knowledge to discover the closest peers to a node and calculated what percentage of these nodes GROUPNET discovered. This was undertaken over a number of iterations each time comparing our membership algorithm against that with global knowledge. Figure 4 presents the results of these simulations.

The results show that initially after zero optimisations, which is a random graph, very few of the optimal peers are found. This rapidly increases and then remains relatively constant at approximately 80–90% for node degrees of 4 and 5 and 60% for node degree of 3 for the remaining iterations. This is due in part to the fact that the algorithm will converge to and continue to re-evaluate a known best set of peers, as described in Section 3. This has the effect of better nodes that are not directly connected to those in the currently known best set not being evaluated and subsequently selected. The accuracy presented by these results is sufficient for the searching we wish to carry out. A loose correlation between space and overlay distance is required to perform the searching described earlier. Further simulation is necessary to prove that the immediate GROUPNET peers that are not in the optimal set are not a significant distance away from a node; this may lead to bogus inferences about distance being made.

4.2 Scalability of GROUPNET

The most significant overhead associated with the GROUPNET membership algorithm is coupled with evaluating the suitability of a peer. We conducted simulations to determine the number of nodes that are evaluated with a number of different sized meshes and desired node degrees. We calculated the average and recorded the maximum number of evaluations a node performs after one iteration of the membership algorithm. Figure 5 presents the results of these simulations. Figure 6 shows these results as a percentage of the total graph size.

For small groups, a significant percentage of the mesh is evaluated to determine its suitability. As the mesh size increases, the percentage of the mesh evaluated reduces significantly as shown in Figure 6. However, note how the number of peers evaluated is in the order of hundreds as the mesh size is in the thousands. Although it can be seen that GROUPNET scales well, its practical use in its current form for very large groups is questionable. This problem could be alleviated by intelligently selecting the initial bootstrap point to begin evaluating peers instead of the random selection which takes place here. Note how the choice of desired degree has an effect on the number of evaluations performed.

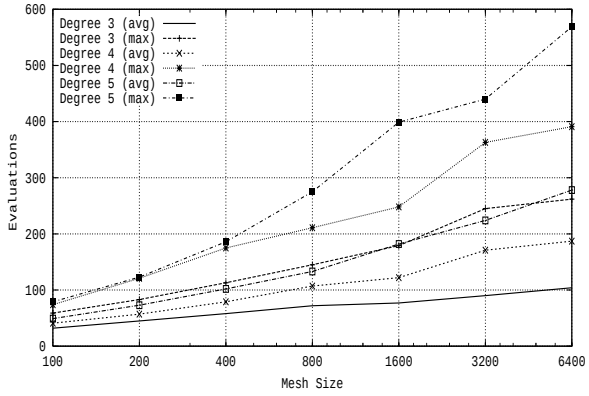


Figure 5: Average and maximum of mesh evaluated for different node degrees

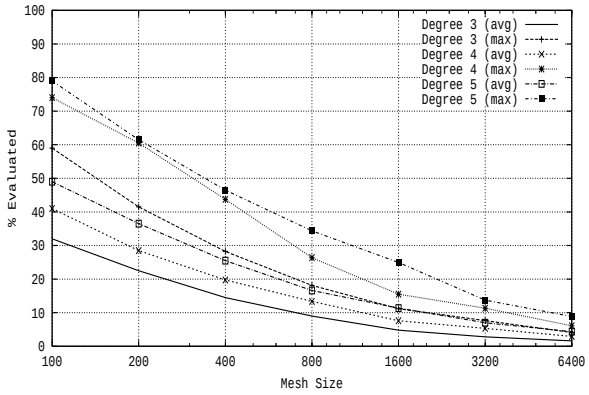


Figure 6: Percentage of the total mesh size evaluated executing membership algorithm

This is due to a node evaluating a larger set of nodes in order to discover a larger set as immediate peers. Currently, it is unclear and difficult to suggest what order of size a mesh would need to be for programmable resource discovery as described here. If this number is in the order of thousands, techniques such as using regional bootstrap points could be employed to improve the scalability of `GROUPNET`.

5. RELATED WORK

There are peer-to-peer networking based solutions that are able to take multi-dimensional resource constraints and resolve/route them to the nearest set of peers that meet the constraints. Noteworthy approaches include CANs [8] and Pastry [10].

The design of Content-Addressable Networks (CANs) is focused on the use of a virtual d-dimensional Cartesian coordinate space, which is partitioned into zones that are controlled by peers in the CAN. A peer wishing to join the CAN selects a point in the coordinate space at which they wish to reside. They send a join message to an arbitrary point in the CAN with their desired coordinates. This message is routed over the coordinate space until it reaches a peer who controls the zone that occupies that point. The peer who owns the zone partitions it with the new peer and shares neighbour information. The new peer is now reachable across the CAN and is adjacent to peers with similar coordinates.

Peers in the Pastry network possess a unique node identifier that is obtained using a secure hash of its public key. As with CANs, a peer's identifier dictates its position within the Pastry network. The group membership algorithm used by Pastry enables peers to be co-located with others that have similar identifiers. Peers in Pastry maintain routing tables containing node identifiers of adjacent peers in the network.

Points in a CAN coordinate space and public keys in Pastry can be used to identify peers that satisfy multi-dimensional resource constraints. However, the values within a dimension are absolute – they do not vary in relation to different view points. It is also unclear how such approaches manage when there are highly dynamic values within dimensions as in a programmable networking environment.

For the purpose of programmable resource discovery, `GROUPNET` is used to construct a topologically-aware overlay network. Other approaches to constructing such overlays exist, for example Distributed Binning [9] and Scalable Adaptive Hierarchical Clustering [7].

Distributed Binning involves associating nodes with logical *bins* based upon relative or absolute values that are derived from measuring a node's distance from a number of well-known landmarks. We believe that scalability issues could arise if a significant number of nodes measure themselves against a much smaller number of landmarks. More acutely, this problem could lead to a node being inaccurately associated with a bin due to significant link stress at a landmark.

Scalable Adaptive Hierarchical Clustering is a method of building a hierarchy of nodes, based on the notion of proximity, in a distributed and scalable way. The hierarchy is built through a series of "local" decisions involving only a small subset of the hierarchy's population for each decision. By using only a series local decisions and an innovative approach to adaptive cluster size distribution, the authors present a scalable means of constructing application-level overlay networks. Primarily this approach is intended for tree construction, however the algorithm can be trivially extended to generate overlay meshes.

6. CONCLUSIONS

In this paper we have presented a mechanism to resolve arbitrary programmable service deployment constraints into a suitable node configuration. To focus constraint resolution we arrange programmable elements into an overlay network constructed using our `GROUPNET` algorithm. A property of the overlay enables us to interpolate/extrapolate more favourable locations to search for suitable node configurations. Due to the nature of programmable service deployment constraints, we advocate the use of programmable service components to evaluate individual element suitability.

We have presented the `GROUPNET` overlay construction algorithm. Simulations suggest that `GROUPNET` will enable us to perform effective searching based on a set of service-specific deployment constraints of a set of programmable resources in order to discover suitable configurations. Furthermore, simulations have been presented that indicate that although `GROUPNET` in its current form theoretically scales well, its practical application for large meshes is questionable. We have presented reasons for this scaling difficulty and suggested practical approaches to alleviating this problem.

Future work will include developing approaches to im-

prove the scalability of GROUPNET for larger groups. Investigation of other application domains for GROUPNET will also be considered. We also feel that with minor adjustments to the CAN construction mechanism, the routing abilities of CANs could be used to address regions of an overlay constructed using GROUPNET.

7. ACKNOWLEDGEMENTS

This work was carried out under the Alpine project funded by BTexact Technologies. Paul Smith has a CASE studentship with the EPSRC and BTexact technologies.

8. REFERENCES

[1] B. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, July 1970.

[2] I. Clarke, O. S., B. Wiley, and T. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In H. Federrath, editor, *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *LNCS*, New York, 2001. Springer.

[3] M. Fry and A. Ghosh. Application Level Active Networking. In *Computer Networks and ISDN Systems*, 1998.

[4] A. Ghosh and M. Fry. Javaradio: an application level active network. In *Third International Workshop on High Performance Protocols (HIPPARCH '97)*, London, June 1997.

[5] A. Ghosh, M. Fry, and C. Crowcroft. An Architecture for Application Layer Routing. In *IWAN*, May 2000.

[6] P. Gummagi, S. Sariou, and S. Gribble. A Measurement Study of Napster and Gnutella as Examples of Peer-Peer File Sharing Systems. In *Multimedia Computing and Networking (MMCN)*, January 2002.

[7] L. Mathy, R. Canonic, S. Simpson, and D. Hutchison. Scalable Adaptive Hierarchical Clustering. *IEEE Communications Letters*, 6(3):117–119, March 2002.

[8] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *ACM Sigcomm 2001*, August 2001.

[9] S. Ratnasmay, M. Handley, R. Karp, and S. Shenker. Topologically-Aware Overlay Construction and Server Selection. In *IEEE INFOCOM 2002*, June 2002.

[10] A. Rowstron and P. Drushcel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM Middleware 2001*, Heidelberg, Germany, November 2001.

[11] S. Simpson, M. Banfield, P. Smith, and D. Hutchison. Component Selection for Heterogeneous Active Networking. volume 2207 of *LNCS*, pages 84–100. Springer, September/October 2001.

[12] P. Smith, L. Mathy, R. Canonico, and D. Hutchison. ALM and ProgNets for v4-to-v6 Multicast Transition. In *IEEE OpenArch 2001 - Short Paper Session "Ghosts of the Net!"*, Anchorage, Alaska, April 2001.

[13] D. Wetherall. Active network vision and reality: lessons from a capsule-based system. In *Operating Systems Review*, volume 34(5), pages 64–79, December 1999.

[14] D. Wetherall, J. Guttag, and D. Tennenhouse. ANTS: Network Services Without the Red Tape. *IEE*, April 1999.