Adaptive Resource Discovery for Ubiquitous Computing *

Rae Harbird
Dept. of Computer Science
University College London
Gower Street, London
WC1E 6BT, United Kingdom
r.harbird@cs.ucl.ac.uk

Stephen Hailes
Dept. of Computer Science
University College London
Gower Street, London
WC1E 6BT, United Kingdom
s.hailes@cs.ucl.ac.uk

Cecilia Mascolo
Dept. of Computer Science
University College London
Gower Street, London
WC1E 6BT, United Kingdom
c.mascolo@cs.ucl.ac.uk

ABSTRACT

The terms pervasive and ubiquitous computing are used to describe a smart space populated by hundreds of intelligent devices that are embedded in their surroundings. Characteristically, ubiquitous computing devices must blend into the background, unobtrusively collaborating to provide value-added services for users. Services are thus essential to the success of this technology and, as a result, both service discovery and service management will play a vital role in generating the revenue stream that is a prerequisite for sustainable ubiquitous deployment. On the one hand, the services provided should be evident by their richness and variety and on the other, the complexity inherent in the environment must be hidden from users. In this paper, we describe RUBI, a resource discovery framework for ubiquitous computing. RUBI represents a novel approach to resource discovery, because the primacy of the need for adaptive autonomic behaviour is established within its design.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed Applications*; C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Applications*

General Terms

DESIGN, ALGORITHMS

Keywords

Resource discovery, ubiquitous computing, mobile ad hoc networks, autonomous, adaptive

1. INTRODUCTION

The pervasive computing vision is one in which the everyday environment becomes populated with embedded microprocessors, and where that computing is invisible to users. It comes as a surprise to most people (and a testament to the invisibility of these devices) that the average household in the developed world already possesses in the order of 40-50 processors – some cars alone possess in excess of 60 separate processors. Over 250 million 8 bit processors are sold *every month*, dwarfing the sales of anything produced for conventional computers: less than 2% of microprocessors are destined for PCs, and that proportion is falling. Few of these embedded microprocessors are currently networked; however, the decreasing cost of wireless networking and the developing range of different wireless technologies means that this is about to change. Given the scale of deployment this, in turn, opens a huge potential market, in which return on investment will be driven by the provision of novel joined-up services that enhance the capabilities of these devices[1]. Consequently, service (or, more generally, resource) discovery and management are necessarily vital components in the realisation of the ubiquitous computing vision, and a key enabler for the generation of revenue streams.

Resource discovery is, conceptually, relatively simple. A service advertises its attributes and location in some form of a distributed directory; a client queries the same distributed directory, specifying the attributes that it desires. Implementing this simple dialogue is, however, complex. Amongst many other things, there is a need to decide: how the attributes should be expressed, [2]; what should be the form of the distributed directory used; where to place information about services and how that information should be moved around the system, [3]; and how to secure access to that information. More generally, ubiquitous computing systems operate in a domain of uncertainty; huge heterogeneity, huge scale, and great dynamicity are expected. Many devices are likely to be very resource poor and yet will be complex to administer, since the number, types, and precise attributes of the services in use will change frequently, as devices change context of operation. As a result, autonomic mechanisms for resource discovery and configuration are important in reducing the cognitive load on naive human users who would otherwise be expected to select from a wide and ever-changing range of services and then manage dynamic access. Autonomy is essential to fulfill the requirement of unobtrusive operation.

*This work was supported by BT Labs, Martlesham, as part of the UCL@Adastral Park, MARS project.

2nd Workshop on Middleware for Pervasive and Ad-Hoc Computing
Toronto, Canada

[1]Given the current state of both radio and broadband technology, we do not expect large returns from simply providing connectivity.

The effects on resource discovery are severalfold:

- An ubiquitous computing resource discovery protocol must be distributed, available at many points in the network, and highly scalable, in order to cope with the potential number of resources available.
- Users can be expected to possess a variety of devices that range from very resource poor to resource rich. They will expect largely device independent interaction using common interfaces in which human intervention must be minimal to reduce confusion. Clearly, capabilities such as memory, processor speed and bandwidth should be exploited where these exist.
- The communication model will influence the design of the resource discovery protocol. There is no single communication paradigm for ubiquitous computing and the communication model in operation depends upon several factors: the network technology used, the relative mobility of the nodes and the availability of a fixed infrastructure. In a situation where nodes have low relative mobility and there is a base station in the vicinity then the most obvious configuration is that of a wireless Local Area Network (LAN). Mobile Ad hoc Networks (MANETs) do not require the presence of a base station and, in some circumstances, it is feasible to form a MANET even if nodes have relatively high mobility, or are failure-prone, provided that there is cooperation (which can be negotiated on-the-fly). In reality, and particularly in less well developed areas, the network is unlikely to conform to one model or the other: the communication model might be mixed, with areas of the network that have fairly good stability and reachable infrastructure, and other areas in which there are neither.

Following from the requirements outlined above, a resource discovery protocol must be adaptive, providing up-to-date resource information over a range of network topologies, despite the frequent failure or unavailability of devices, services and the links between them. It must accommodate and exploit the heterogeneity of participants, whilst still serving information about a large number of fast-changing resources.

In the remainder of this paper, section 2 contains an analysis of current approaches to resource discovery. Following this discussion, section 3 contains a description of RUBI, our resource discovery protocol, which has been designed to address the requirements identified above. Section 4 contains a critical analysis of RUBI compared with the service discovery protocols reviewed in section 2. Section 5 concludes the paper with an outline of proposed future work.

2. PREVIOUS WORK

Existing resource discovery protocols can be categorised according to whether a global index of resources is maintained centrally, a distributed index is maintained, or resources are discovered as they are needed. The main purpose of maintaining a central directory is to provide efficient management of a large number of resources. Jini [16] conforms to this architecture, assuming a stable, fully routed, underlying network and the presence of a reliable node to act as the directory. These assumptions are valid for Jini as it was developed for use in fixed networks but it may not fit an ubiquitous computing environment where the topology of the network is volatile and the directory may not be reachable at all times.

Resource location mechanisms developed for peer-to-peer networks provide interesting examples of distributed indexes. A series of distributed filesystems based on hash tables, known as Distributed Hash Tables (DHTs), has been independently developed, for example, [12]. INS/Twine, [1] is a resource location service based on a DHT with nodes in the DHT responsible for maintaining part of the hash range which represents an index to available resources. The principle advantages of DHTs are their scalability and efficiency. A relatively small amount of state is maintained per node in order to route queries; query resolution path lengths are short, only increasing logarithmically with the number of nodes in the network. However, it is unlikely that the integrity of a DHT can be maintained easily where there is a high node or link failure rate without a significant message transmission overhead. This issue is addressed in the search protocol used in JXTA [15] which is based on a DHT but differs in that it is 'loosely-consistent'. A certain amount of inconsistency is tolerated because hash table indexes are replicated at neighbours and, if a node fails, a request resolver can inspect the indexes closest in range to the target. Although this algorithm can tolerate a certain level of node failure, the message overhead increases as the distributed index becomes increasingly inconsistent. The DHT may become unusable in conditions of high relative node mobility that characterises some ubiquitous computing operations.

In [8] the resource discovery protocol maintains a distributed index of resource information but global resource information is replicated in each directory. The protocol is designed for use in a MANET and is integrated with the underlying, proactive routing protocol. Nodes with the greatest transmission range are used as both directory agents and routers. The routing algorithm may not be suitable where nodes have a high link failure rate as the overhead of electing and maintaining the directories will be too great.

Konark [6], UPnP [9] and SLP [4] allow discovery of resources on demand. They are all adaptive in the sense that they can make use of a pre-configured, centrally located server, although they do not rely on the presence of such an entity; any node may choose to act as a directory and publish requests. The resource discovery algorithms are simple but again, they rely on a fully routed network infrastructure which may not be available in a pervasive computing environment.

3. RUBI

RUBI, our resource discovery framework for ubiquitous computing, represents a novel approach. This is primarily because it encapsulates an overarching adaptive process, controlling the way that information is disseminated and retrieved, based on local views of the structure of the network. The design of RUBI is founded on the hypothesis that there are significant similarities between resource discovery in a network and the operation of a routing algorithm, since both are fundamentally concerned with the dissemination of information about the availability and efficiency of access to resources. It is, therefore, our contention that the lessons learned in this domain can be applied more generally and, indeed, that the regular exchange of routing information can form the basis for the exchange of more general information

about services.

Within the field of mobile ad hoc networking, there are, broadly, two different approaches to the dissemination of routing information. The first is *proactive*, in which tables of forwarding paths are periodically exchanged, as in fixed network routing algorithms such as OSPF, [10]; the second is *reactive*, in which information about routes is determined on an as-needed basis such as in DSR and AODV, [7, 11]. These different approaches are useful in different circumstances. Thus, for example, in areas of high relative mobility, routing information becomes outdated rapidly, so reactive discovery is more likely than proactive to give a good answer. In areas of low relative mobility, proactive dissemination of information is more efficient, since little changes. Unsurprisingly, a range of hybrid approaches to dissemination have also been developed; ZRP, [5] is an example.

The same distinction between high and low relative mobility is important in more general service discovery: the former means that efficient placement of service information is difficult and expensive, the latter that it is reasonable to expend cycles in order to ensure efficient lookup. In line with this, in areas of low relative mobility, RUBI uses a discovery mechanism based on a proactive routing algorithm and resources are advertised throughout the network. In regions of high relative mobility, where it is possible to guarantee neither the availability of a node that will act as a directory nor the accuracy of resource information, RUBI initiates dynamic resource discovery using a reactive protocol and discovers resources as they are needed. It is important to note that RUBI allows nodes to make *local* decisions about which of these approaches it will adopt at any given point: in other words, within a large connected network, there are expected to be dynamically defined and dynamically changing areas of proactivity and of reactivity that must interwork to allow seamless service discovery for clients. This approach is reflective of the fact that it is unreasonable to expect networking environments to be either entirely static or entirely free moving.

The ability to adapt also means that RUBI is resilient to failure over the wide range of network conditions that may be found in an ubiquitous computing environment and can operate in both infrastructure-based and mobile ad hoc network types, migrating seamlessly between both. RUBI has been designed to operate at either the network layer of the OSI model or above the transport layer. It relies on the services provided by a link layer protocol such as 802.11 but does not require a routing protocol or a reliable transport service: as RUBI is based on routing protocols, it can provide this functionality if needed.

In the following section, we examine the detailed operation of RUBI. For the sake of clarity we break this down into four key aspects: the proactive and reactive discovery mechanisms themselves; the local decision procedure that determines which type of service advertisement and discovery to use; the issues in ensuring interoperation between the different discovery mechanisms; and the issue of failure.

3.1 Discovery operation

The two ad hoc network routing algorithms selected as a basis for RUBI are OLSR [14], which is a proactive algorithm suited to fairly stable, densely populated networks, and AODV, a reactive protocol suited to networks of up to thousands of nodes in size, where the population of nodes may have a relatively high mobility rate. These algorithms have been chosen because they integrate well in situations where both are running in close proximity: both algorithms allow nodes to cache resource information and requests can be answered by an intermediary that is not the resource provider yet has up-to-date cached information. There are four message types used in RUBI and these are resource advertisements, resource cancellations, resource requests and replies.

The two resource discovery algorithms process these messages differently. The proactive resource discovery algorithm uses information from the neighbour establishment process described in section 3.2 and each node elects a subset of its neighbours as relays. Relays are highly-connected nodes responsible for propagating resource information further through the network using other relay nodes in turn. In this way, the use of relays enables optimised flooding of messages, reducing the transmission costs of distributing resource information as compared with classic flooding.

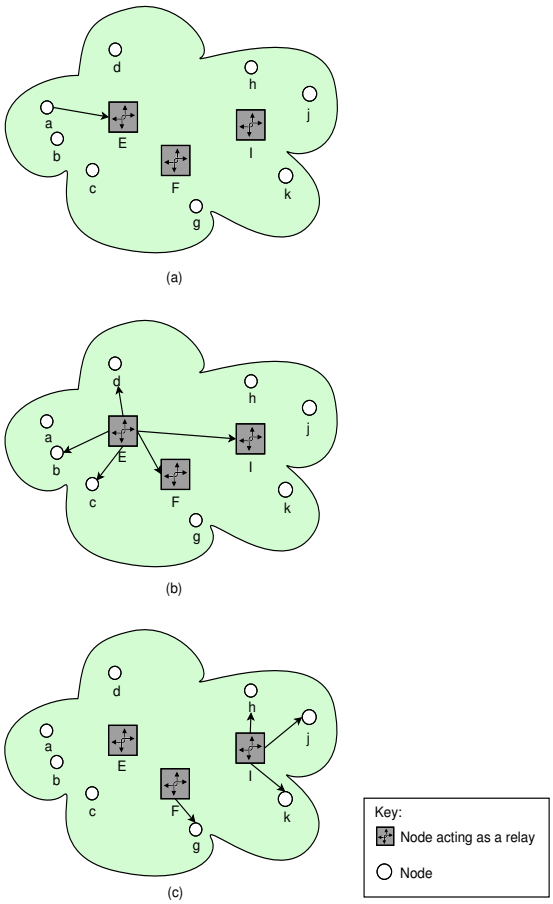

Figure 1: Proactive Resource Advertisement

Figure 1, shows proactive resource advertisement as a series of steps with the arrows representing message transmission. In Figure 1(a), *node a* broadcasts a resource advertisement which will be received by *relay E*. The relay may not forward the advertisement immediately but will periodically construct and forward composite advertisements that contain details about all the resources available at those nodes

directly connected to it. Figure 1(b) shows the resource advertisement being forwarded in turn by *relay E* to its own relay set. Finally, in Figure 1(c), an advertisement is forwarded to all the neighbours of *relay F* and *relay I*. At this point nodes in the network will know about every resource available through *relay E*. Although resource advertisements are distributed periodically, the period may vary in length depending on whether there have been any changes in either the neighbour membership or the resources available. If changes are detected by a relay then the resource advertisements will be more frequent. The overhead of caching resource information is only incurred by relays, although any node may store the resource details it receives. In the case where a node has chosen not to cache resource information then it will send a resource request message to its relays as needed and the relay nodes will respond with reply message if the resource exists.

A resource cancellation message received from a resource owner will be treated in the same manner as an advertisement. The relay receiving the message will update its cache, this time deleting the resource details, and subsequent resource advertisements will reflect the change.

Proactive discovery is a good strategy for regions where nodes have low mobility. The primary reason for this is that current information about resources is maintained at many points: in this case, wherever there exists a relay.

Reactive resource discovery is achieved by maintaining resource information in the network for recently queried resources only. Unlike proactive discovery, all nodes are responsible for maintaining their own caches of resource information. A client needing to locate a resource initiates a request that is flooded through the network. The scope of the request, in terms of the number of hops it traverses, is controlled by a time-to-live (TTL) value and nodes can perform an Expanding Ring Search (ERS) by repeating the request with a higher TTL if there is no response initially. In this way, the overhead of finding a resource in terms of the total number of messages generated is reduced in circumstances in which a resource is available nearby. Any node that receives a request and owns, or has learnt about, a matching resource may reply to a request; this reply is then unicast back to the client with information about the distance of the resource holder in hops from the responder. Any node operating the reactive discovery algorithm will ignore resource advertisement and resource cancellation messages, only processing requests and replies. Reactive discovery is a good strategy for areas where nodes are fairly mobile and the overhead of maintaining accurate resource information throughout the region would carry a high message transmission overhead.

For both algorithms, routes to nodes serving resources could be constructed if necessary just as they would be in the normal operation of the routing algorithms used as a basis for RUBI. In the case of the proactive algorithm, routes would be calculated as resource advertisements travel through the network; whereas for the reactive algorithm, routes would be calculated at the same time as request messages are forwarded.

3.2 Neighbour establishment and algorithm selection

Having seen that RUBI can operate in individual areas of proactivity or reactivity, we need to consider how a node determines to which such area it belongs. Each node must be capable of making this decision in an autonomous fashion, based on the information it is able to glean from its immediate locality. A node will only use a proactive resource discovery algorithm if the links with its neighbours are sufficiently stable, that is to say, they have remained neighbours for at least the duration of a given threshold value. For RUBI, the optimum range for this threshold will be determined through simulation.

Although neighbour establishment represents an overhead in terms of the resource discovery process, in that it does not directly contribute to it, it serves an important purpose. The objective of neighbour establishment is twofold. Firstly, as with OLSR, it enables nodes to ascertain the most efficient method of disseminating the protocol, (resource discovery) messages through the network. Essentially, this is achieved by electing nodes with the greatest transmission capability and the highest number of links in the locality to act as relays thus ensuring that messages are propagated through the network with the minimum number of retransmissions, an important consideration for power-constrained nodes. Secondly, in RUBI, neighbour establishment allows each node to monitor the longevity of links in its immediate locality and, based on the perceived stability, to autonomously select the most efficient resource discovery algorithm to use.

In RUBI, the neighbour establishment process used is exactly the same as that used in the OLSR routing algorithm. A RUBI node broadcasts HELLO beacons periodically and the receiving nodes use these as a way of building up a picture of the local topology to a 2-hop distance. Each HELLO message contains a willingness value and a list of the node's neighbours. Willingness is expressed as a numeric value between 0 and 7 representing the amenability of the sender to fulfill the role of a relay by forwarding messages, caching resource location information and responding to resource discovery requests. A limited amount of information about neighbours is also passed on with their identity and neighbours are categorised according to whether the sender has selected the neighbour as a relay. In this way, each node can build up a picture of both its immediate neighbours, and those neighbours 2 hops away (that is to say, its neighbours' neighbours). It will also learn whether it has been selected as a relay by the originator of the HELLO message and should fulfill the relay function for that neighbour. Information about both immediate and 2-hop neighbours is stored by nodes in a neighbour table and a 2-hop neighbour table and these tables are updated with every HELLO message received. Entries are associated with an expiry time and will be removed from the table if no update arrives.

The heuristic used by each node to select the subset of its neighbours to use as relays can be summarised in 3 steps. Firstly, all neighbours advertising the maximum willingness value are automatically selected as relays. Secondly, neighbour nodes that are the only nodes through which a 2-hop neighbour is (or neighbours are) reachable are added to the set. Finally, neighbours through which the remaining nodes in the 2-hop group can be reached are included in the relay set. If there is more than one neighbour node that fulfills this condition then the one with the greatest number of links is chosen.

Relay selection is illustrated in Figure 2 and in this example, the selection process for *node Z* is described. It is assumed that neighbouring nodes have bi-directional communication. The transmission coverage for *node Z* is shown

as a circle with a solid circumference and *node Z*'s neighbours are *w*, *x* and *A*. The circle with a dotted circumference represents the transmission coverage for *node A*. *Node A* is a base station and has much greater transmission capabilities than the other nodes surrounding it. *node A's* neighbours are *b,c,d,e,f,w* and *Z*. Assuming that *node A* sends *node Z* a HELLO beacon containing the maximum willingness value it will be selected by *Z* as a relay. Using the neighbour information contained in the HELLO beacon from *node A, node Z* can tell that it can reach all of its 2 hop neighbours apart from *node Y*, through *A*. Assuming that *node Y* can only be reached through *node x*, *node x* will be added by *Z* to its relay set. At this point, calculation of the relay set is complete as all neighbours and 2-hop neighbours can be reached. When *node Z* constructs its own HELLO beacon it will set a flag indicating that it has chosen *node A* and *node x* to act as relays on its behalf.

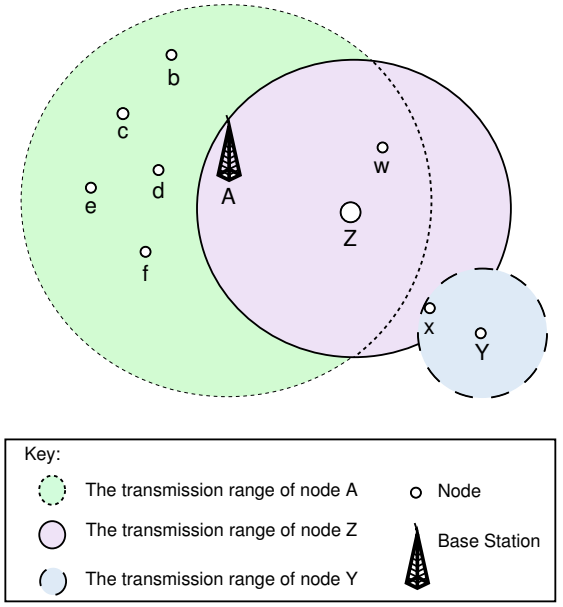

Figure 2: Neighbour Establishment for Node Z

3.3 Coexistence of proactive and reactive protocols

In order for a global system to operate, the areas in which the different approaches to service discovery are deployed must interwork. Since, by definition, such areas will border one another, there must be adjacent hosts that consider themselves part of different types of area and can act as gateways between the areas. In most cases, operation between the proactive and reactive regions at gateway nodes is seamless, but there is one scenario in which the proactive protocol must be modified. A simple alteration is proposed in the first instance.

In Figure 3, there is a user at *node a* in the region where the proactive resource discovery protocol is used that wishes to know about a printing service. Thus, as *node a* is already aware of all the resources available in the area, (or can find this out from *relay D*), it is aware that that there is no such resource available. In this case, it is necessary to forward the query into the reactive region and *node a* attempts to search beyond the bound of the immediate area by forwarding the request according to the rules of the reactive algorithm. This is achieved by setting a flag in the resource request message to indicate that it should be dealt with in the same way as the reactive protocol and the request is broadcast onwards. At *node q*, the request is satisfied, and a reply is constructed and returned along the route of the request.

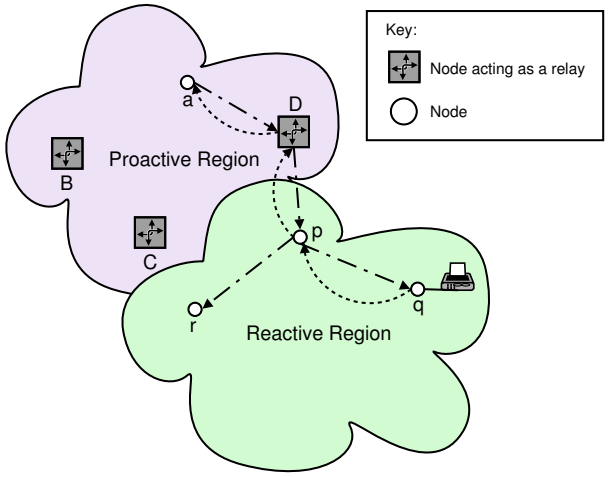

Figure 3: Integrated Discovery Protocols

3.4 Recovery from failure

Failures can be categorised as those in which a *resource* ceases to be available, and those where a *node*, and therefore all the resources it offers, is no longer available for some reason. The freshness of resource information in RUBI is maintained in the same way as in Jini. That is to say, any resource information cached is associated with an expiration period, after which it is deleted. If a resource becomes unavailable before the expiration period has elapsed then the information served will be incorrect. Where a reactive algorithm is in operation, there is no simple way to detect whether the resource remains available throughout the expiration period. This is because a resource request may be answered by nodes that have recently cached information, rather than by the resource owner. If a proactive algorithm is used, resource failure detection may be more accurate as it is possible for a resource owner to distribute a resource cancellation message. As a result, a relay can delete resource information and will not distribute it with the next advertisement. RUBI has no special recovery procedure for resources that go offline and return a short while later and in this case, resource owners will register resources with relays in the usual way if a proactive algorithm is in use.

A node that notices a change in its local topology, such as the disappearance of a link to its neighbour, will permanently delete the neighbour from its database along with any resources hosted on it. Incorrect information regarding the availability of the failed node and its resources will remain in the network until either, a resource advertisement is propagated that reflects the change or, information about the node and its resources expires.

In terms of resource information availability, the proactive algorithm is resilient in the face of relay failure and if one or more of a client's relays fails, a client can still have its queries answered by any remaining relays in its set. In the

situation in which a relay comes back online after a short disappearance, there are several possible recovery mechanisms. A relay that temporarily loses connectivity could request resource information from one of its own relay set when it comes back online and if the RUBI node is also providing route information for the resources it has then this can also be requested.

4. DISCUSSION

In this section, RUBI is compared with the other resource discovery protocols described previously in section 2 and the advantages and disadvantages of the RUBI framework are evaluated. Unlike the other resource discovery protocols reviewed, RUBI has been designed with an ubiquitous computing environment in mind. The novel design of RUBI reflects a tradeoff between the ability to adapt dynamically based on knowledge of local conditions and the efficiencies gained by assuming a fairly stable environment. Adaptability comes at a cost and RUBI must periodically monitor the duration of neighbour links to assess the stability of its locality. This represents an overhead in terms of bandwidth consumption that the resource discovery protocols reviewed do not incur as they assume a fully routed infrastructure. Furthermore, RUBI will not achieve the efficiency of those service discovery protocols that cache information centrally (Jini) in terms of the number of messages transmitted in order to maintain an up-to-date view of resources. Neither will it attain the scalability of those protocols that maintain a distributed index such as INS/Twine. In RUBI, information is cached at many more places in the network, incurring additional resource overhead to update the information in each cache.

5. CONCLUSIONS AND FURTHER WORK

Resource discovery will be a key factor in enabling the ubiquitous computing vision where the devices embedded in our surroundings autonomously cooperate to discover and provide services with minimal human intervention. By identifying the features that typify operation in such an environment, it has been possible to design a resource discovery framework that is well-suited to it, namely RUBI. Primarily, the applicability of a resource discovery algorithm depends on the features of the network in which it is operating. For ubiquitous computing, the network is characterised by regions that are fairly stable possibly coexisting with regions where nodes are highly mobile and failure-prone. RUBI represents a novel approach because it is based on routing algorithms, thus ensuring the efficient dissemination of resource information. Furthermore, RUBI is adaptive and, by monitoring the network in the locality, nodes autonomously select the most suitable routing algorithm to use.

In the future, the performance of RUBI will be evaluated through simulation under different network conditions, varying the network size, node degree and node mobility level and the number of resources and requests. Furthermore, a series of design improvements will be assessed. As a starting point, the protocol integration in RUBI is too simplistic with relays flooding requests through the proactive region if they cannot be resolved. It would be more efficient to unicast requests to a gateway node that borders the proactive region and a means of achieving this will be explored. Finally, the security of the RUBI protocol will be addressed by integrating it with it with ADAM [13], a middleware architecture for distributed access-control.

6. REFERENCES

[1] M. Balazinska, H. Balakrishnan, and D. Kargar. INS/Twine: A scalable peer-to-peer architecture for intentional resource discovery. In *Proc. of the International Conference on Pervasive Computing (PERCOM 2002)*. Springer-Verlag, 2002.

[2] D. Chalmers, M. Sloman, and N. Dulay. Map adaptation for users of mobile systems. In *WWW10*, pages 735–744, 2001.

[3] K. Cheverst, N. Davies, K. Mitchell, and A. Friday. Experiences of developing and deploying a context-aware tourist guide: the GUIDE project. In *Mobile Computing and Networking*, pages 20–31, 2000.

[4] E. Guttman. Vendor Extensions for Service Location Protocol, Version 2, Network Working Group, Request for Comments: 3224. `http://www.ietf.org/rfc/rfc3224.txt`, 2002.

[5] Z. Haas. A new routing protocol for the reconfigurable wireless networks. In *Proc. of the IEEE Int. Conf. on Universal Personal Communications*, 1997.

[6] S. Helal, N. Desai, V. Verma, and C. Lee. Konark A Service Discovery and Delivery Protocol for Ad-hoc Networks. In *Proc. of the Third IEEE Conference on Wireless Communication Networks (WCNC), New Orleans*, 2003.

[7] D. Johnson, D. Maltz, and Y.-C. Hu. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR), IETF draft. `http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-10.txt`, 2004.

[8] U. C. Kozat and L. Tassiulas. Network Layer Support for Service Discovery in Mobile Ad Hoc Networks. In *Proc. of IEEE INFOCOM, San Francisco,USA*, 2003.

[9] Microsoft. Understanding UPnP: A White Paper. `http://www.upnp.org/download/UPNP_UnderstandingUPNP.doc`, 2000.

[10] J. Moy. OSPF Version 2, Network Working Group, Request for Comments: 1583. `http://www.ietf.org/rfc/rfc1583.txt`, 1994.

[11] C. Perkins, E. Belding-Royer, and S. Das. Ad Hoc On-demand Distance Vector (AODV) Routing, Network Working Group, Request for Comments: 3561. `http://www.ietf.org/rfc/rfc3561.txt`, 2003.

[12] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scaleable content-addressable network. In *Proc. of ACM SIGCOMM 2001, San Diego, California*, 2001.

[13] A. Seleznyov, M. Ahmed, and S. Hailes. ADAM: An Agent-based Middleware Architecture for Distributed Access Control. In *The Twenty-Second International Multi-Conference on Applied Informatics: Artificial Intelligence and Applications*, pages 200 – 205, Innsbruck, Austria, 2004. IASTED, ACTA Press.

[14] E. T. Clausen and E. P. Jacquet. Optimised Link State Routing Protocol (OLSR), Network Working Group, Request for Comments: 3626. `http://www.ietf.org/rfc/rfc3626.txt`, 2003.

[15] B. Traversat, M. Abdelaziz, and E. Pouyoul. Project JXTA: A Loosely-Consistent DHT Rendevous Walker. `http://www.jxta.org/project/www/docs/jxta-dht.pdf`, 2004.

[16] J. Waldo. The Jini architecture for network-centric computing. *Communications of the ACM*, 42(7):76–82, 1999.