Weaving the Pervasive Information Fabric

Mark Thompson, David De Roure, and Danius Michaelides

Department of Electronics and Computer Science,
University of Southampton, UK
{mkt,dder,dtm}@ecs.soton.ac.uk

Abstract. As the pervasive computing infrastructure emerges, we consider the impact on information applications and their middleware requirements – what we call the pervasive information fabric. In particular, we consider the evolution of hypermedia link services to support the structuring of ad hoc information spaces. We describe a prototype implementation of an ad hoc distributed link service using a tuple-space model.

1 Introduction

Advances in network technologies and miniaturisation of computing devices are providing us with a new target architecture for computer science and engineering, with devices embedded in our environment, in wearable computers and potentially in everyday artefacts. Mobility of users and devices is a particularly profound paradigm shift; it also introduces a new type to our information space, i.e. location information. This pervasive computing infrastructure is with us now, with increasing numbers of devices available, motivated for example by the needs of the mobile e-business user. We now anticipate the gradual disappearance of the general purpose 'personal computer' in favour of task-specific devices that we will not think of as computers at all.

Meanwhile, the majority of hypermedia systems that we work with are intranet based with a traditional client-server model based on large servers accessed by many clients and interconnected by a static network. Activities such as WAP (Wireless Application Protocol) [5] and JINI [8] are establishing an infrastructure for the mobile and pervasive computing environments, but we believe that the information perspective, in particular the hypermedia perspective, is one that has been little explored thus far.

We focus here on one particular aspect of pervasive computing: ad hoc networking. In this scenario, a number of devices that happen to be in the same physical space can establish communication and provide exchange of information. The network technologies and protocols to support this are becoming established. Above this level, we regard hypermedia link resolution as a basic service that helps applications provide customised information spaces. Hence the focus of our investigation is the provision of link services in the ad hoc setting, in order to provide hyperstructure to facilitate navigation of the available information space.

This paper motivates the study and describes some aspects of our research agenda to create this pervasive information fabric. In section 2 we discuss the fabric, followed in section 3 by a motivational scenario, which leads us to the ad hoc information space discussed in section 4. Section 5 focuses on link service issues, and section 6 presents a prototype implementation. Finally we describe future work with the system and where it interacts with existing projects within our laboratory.

2 The Pervasive Computing Fabric

The pervasive computing vision was established by work at Xerox around 1989 on 'ubiquitous computing' and brought to public attention through Weiser's Scientific American article [15] and Communications of the ACM [12, 16]. The last few years have seen the infrastructure technologies emerge. Next generation Internet Protocol (IPv6) supports many aspects of pervasive computing, including the address space for large numbers of devices, and there is ongoing activity at IETF (Internet Engineering Task Force) in support of mobility, ad hoc networking, service discovery and automatic configuration of devices.

This same period has seen a huge increase in the availability, diversity and richness of digital content, compounding the difficulties of content delivery in this increasingly pervasive setting. WAP addresses delivery of content, and techniques for dealing with adaptive and scalable delivery of multimedia content is attracting much attention (e.g. [11]). This technology is a further constituent of the pervasive multimedia computing infrastructure.

3 An Ad Hoc Linking Scenario

What are the applications that will employ these underlying devices, protocols techniques for content processing and delivery? Our particular focus here is on information applications that take advantage of a hyperstructure. In particular, we adopt the open hypermedia model, whereby links are first class citizens and we can readily work with linkbases and linkbase fragments.

Imagine you are walking along a corridor, you look at a poster on the wall and the text and images on the poster prompt you to think of a number of associated things. Think of this as having a few collections of hypermedia links (the sets of associations) active in your mind. In our scenario, a wearable computer behaves in a similar way, reifying and augmenting our mental linkbases with various personal linkbases, a linkbase associated with your current task and one associated with the building you are in. This highly dynamic, highly context-sensitive and ad hoc situation is what we are trying to realise through the hypermedia link services in our pervasive information infrastructure.

We extend this vision to a collaborative setting: you walk into a meeting room and sit down with several other people. Everyone is carrying computing devices and these form a shared workspace of documents relevant to the meeting; perhaps each has a Web server. How is this information space structured?

Imagine everyone has linkbases that they publish – then we have an ad hoc aggregation of linkbases to assist in constructing the hyperstructure for the shared information space. Looking at the agenda of the meeting, you now have links available according to the linkbases of everyone in the room. Even if you do not use these directly, your agent may wish to – perhaps, trivially, for search purposes, or in general to answer a context-sensitive query.

The key notion here is that the topology of the underlying network is not predetermined or particularly organised, rather it is self-organised and ad hoc. Also, the location and availability of network services (web servers, etc.) are fluidic in nature.

4 Ad Hoc Information Space

Whilst the meeting is in progress, users can browse documents present on the collection of web servers in the meeting. Any of the published linkbases can also be applied to these documents, rendering an augmented view, specialised to the meeting at hand. Once the meeting is over, the hyperstructure that has been created in the meeting through the amalgamation of linkbases, documents and interactions (our ad hoc information space) is deconstructed.

In order to preserve the space, documents, linkbases and accompanying information, data shared during the meeting would have to be mirrored. Many tools exist to perform the mirroring of entire documents present on web servers; however it may be the case that only a summary of the document is required to be taken away, or that the document exists on the globally accessible Internet and thus an entry in a linkbase will suffice.

One approach to mirroring linkbases is to export the complete repository to a persistent entity, in a format that can either be re-loaded into the application at a later date, or into other link services at a different location. This same process enables the merging of multiple linkbases into a conglomeration of links pertaining particularly to the meeting that has just expired. More complicated merging can be achieved by combining linkbases using linkbase algebra, for example, merging only links that specifically relate to a particular set of documents.

5 Mobile and Ad Hoc Linking

Without studying specific application requirements, we have a notion here that the hypermedia middleware – the link service – needs to support ad hoc networking by supporting ad hoc combination of linkbases. But first it must also support mobility of linkbases, sharing and mobile access to linkbases. These then are the goals in our next iteration on the architecture of the Distributed Link Service (DLS) [1, 3], our proven infrastructure for hypermedia linking.

Carrying a linkbase is as simple as carrying any data, but the provision of access requires protocols. The model whereby linkservers can be accessed with HTTP is well established in DLS, but we have recently produced a version using LDAP (Lightweight Directory Access Protocol) as this permits the use of standard infrastructure and deals with some scalability and update issues. We have also implemented effective query routing for multiple link servers [2]. In support of the collaborative scenario above we draw on our experience of collaborative filtering [4] on an Intranet, and look to existing hypermedia models supporting collaboration such as activity spaces [13, 14].

LDAP is well suited as the communications protocol to manage and query the numerous linkbases present in the scenario. It is an existing, off-the-shelf technology with multiple language bindings and multiple server implementations. The protocol itself deals with the issues of scalable access, whilst not strictly relevant to this particular scenario this is an important feature should the techniques developed here be deployed in wider-area information spaces.

The flexible design of LDAP facilitates complex filter-based directory querying (including phonetic matching) and a readily available, extensible data model. The naming scheme employed by LDAP enables scoped directories, which is paramount in this scenario where we have many linkbases being built and distributed between potentially many link servers.

Also, through our collaboration with IBM Hursley Labs, we have introduced Tspaces [17] as a link repository and messaging infrastructure. TSpaces, conceptually nothing more than a communications buffer with database capabilities, enables inter-application messaging in a network of heterogeneous devices and operating environments. Besides the simple primitives of `write`, `read` and `take`, borrowed from its Linda background, TSpaces is an interesting platform for DLS implementations in that it provides event notification services, offers user-definable operations on the repository and has the ability to store arbitrary sized objects (e.g. document chunks), all enabling unique approaches to this DLS scenario. TSpaces is small in memory-, processor- and disk-usage, thus fits well with the long-term target architectures for Pervasive Computing. That is to say it brings the power of the network to palm devices, making them fully-fledged network computers.

6 Prototype System

We have built a system comprising a modified DLS implementation and an application that generates the linkbases automatically in the ad hoc setting. Users submit documents they deem relevant to the meeting via a shared resource; the application then extracts linking information that is made available to the DLS as a custom linkbase for the meeting. In an initial implementation, users at the meeting set their web browser clients to use the DLS as their web proxy; as a result, documents they browse throughout the meeting have the newly created links inserted where relevant. The proxy is also able to record the users' trails and we intend these to be a source of further navigation support in later systems. The second phase of prototyping introduced a local Meeting Participant (MP) "agent" to which the local web-browsers and other Internet applications are pointed. The MP module is then responsible for the interactions of the user with the meeting space, including all resource discovery, service interaction and management.

Extraction of link information from HTML documents is a straightforward task, and when working with open hypermedia systems the separate link databases are already available. The link extraction mechanism used in this prototype is a tool that parses a set of documents, saving link data generated by a specified criteria – for example, by keyword match – in a relational database and making local copies of the documents with the anchors (HREFs) rewritten. The SQL database facilitates the subsequent generation of multiple linkbases. The initial implementation was based on libwww from W3C and ran on FreeBSD and Linux.

In our prototypes, a linkbase is a repository of four-tuple Link objects, with a well-defined set of operations for Creation, Destruction and Query for links and Initialisation, Extend-by-import and Publish for linkbases. Unlike other DLS implementations, the interaction with linkbases is distributed, as are the linkbases themselves. The linkbase algebra is trivial, yet flexible, in that links are added to the linkbase "piecemeal", through submission of grounded tuples; destroyed by "filter", where the filter components are conjunctive; and navigated or resolved by "query", where the exactness of a link can vary from fully grounded (exact match required to verify an existing link against an existing linkbase) to completely vague (just matching on a selective keyword, or the source document location).

In order to enable the different component applications of our scenario, the interface to the DLS has to be specified and well known so as to enable interoperability, and in the extended prototype, automated service discovery. For the purposes of these experiments, this interface is partially defined as follows:

Both the LDAP and TSpaces implementations implement a link service using this interface. The LDAP implementation maintains the repository of links where each link originates from a particular linkbase and has an id within the linkbase. In LDAP terminology, this gives a Distinguished Name for the object, for example '`lid=1., lb="Links about LDAP"`'. The actual information associated with a link become attributes of the LDAP object. To facilitate link algebra, other attributes such as user could be associated with a link. The methods defined by the LinkService interface map to various LDAP operations, for example the destroyLink method implementation for an LDAP link server is a direct wrapper around the LDAP delete operator, qualified by LDAP search filters generated from the arguments.

The TSpaces implementation has links represented as Tuple objects in their own namespace, with additional namespaces used for housekeeping data. This differs from the LDAP approach in that each linkbase is a separate tuplespace, containing Link objects. As with the LDAP implementation of the link service interface, the TSpaces version is a glue-layer to API calls, which result in either a query over a network socket connection or a direct intra-application call. This is dependent on whether the TSpaces server is executing in the same JVM as the client application, as may be the case if participating clients are executing their own link servers locally.

```
interface LinkService {
  boolean authorLink( Link lnk);
  boolean destroyLink( Link lnk);
  boolean destroyLinkSource( String src);
  boolean destroyLinkTarget( String tgt);
  boolean destroyLinkType( String typ);
  void resetLinkbase( );
  boolean importLinkbase( URL src, Format fmt );
  void export (PrintStream out, Format fmt);
  void exportBySource( PrintStream out, Format fmt,
                       String src);
  void exportByTarget( PrintStream out, Format fmt,
                       String tgt);
  void exportByType( PrintStream out, Format fmt,
                     String typ);
  Link queryBySource( String src);
  Link queryByTarget( String tgt);
  Link queryBySelection( String sel);
  Link queryByType( String typ);
}
```

Fig. 1. Standardised LinkService Interface

The interface chosen enables linkbase publication to a number of different formats, including LDIF, TSpaces tuples, Webcosm [7] linkbase and human-/browser-readable bookmark formats. The choice of format is exposed through the defined LinkService API such that Meeting Participant agents can expressly request the most suitable export format for their user's need. This permits integration into other DLS implementations, and also is the mechanism used by the prototype system to off-line snapshots of the current live set of linkbases known by a MP node at runtime. Conversely, the import function as implemented reads a linkbase in from a given URL and linkbase format. Whilst this is fine for a prototype implementation, additional mechanisms for the bulk update of linkbases, other than over HTTP connections, would be required for a production service.

The initial prototypes require each client user to manually configure their web browser's proxy settings to point to a location (host-port pair) that executes the link service web proxy. The proxy is controlled through a web form interface that configures which linkbases to include when answering HTTP queries from a particular IP address, thus specialising the response per participant machine. The triggers to import and export linkbases were also buttons on this form, which inadvertently became the user interface to the application.

Later prototypes introduce the notion of a Meeting Participant agent, which is a local piece of code that the user's applications are proxied through. This application is lightweight in that it does not have all of the DLS implementation that the full prototype has behind it, only the session management and service discovery functionality. Its task is to discover what other DLS services are available in the ad hoc network at meeting-time and offer these additional resources to the local user. It is also the endpoint for communications from other Meeting Participant agents at other clients. Naturally, should the local client also be running a DLS service, its presence will be discovered by the MP agent and made available locally.

The automation required of the MP agent implies a need for some level of service discovery mechanism, finding the relevant resources on the network for the meeting. A trivial implementation of this utilising network sockets is to broadcast or multicast a request packet on a well-known port such that all the available services can reply, announcing their presence and available resources. A more appropriate mechanism is to use a standardised service discovery protocol, such as the IETF SLP[6], or Sun's Jini Lookup Service[8]. The approach taken by this work to define a standardised LinkService interface readily enables the use of Jini-style service lookup based on template matching in that specific requests for specific services can be made, for example to search for a LinkService that implements an export to HTML function.

7 Further Work

This is work in progress, with implications on other projects that the IAM group at Southampton is actively researching. Part of the further work with these prototype applications is to integrate the architecture developed here into the other frameworks available.

In adopting standard infrastructure components to build the prototype we have not talked explicitly in terms of building a multi-agent system; however, it is entirely possible to caste the prototype in that paradigm. This builds on the concept of software agents both as personal information assistants and as autonomous components of a distributed system in which they collaborate and negotiate to perform their tasks. Future work includes an implementation of the scenario within SoFAR, the Southampton Framework for Agent Research [10]. We also wish to consider this work in the light of the FOHM model [9].

Section 4 above discusses the need to mirror documents when deconstructing the information space at the close of the meeting, i.e. when the resource is about to become unavailable. Where current technologies permit relatively low bandwidths for wireless access, the power capability of the client's device may be such that only part of the document, or even a generated executive summary of the document should be transferred. Techniques for doing this are well known and employed in on-line information navigation systems, such as Refindment [7]. Summarisation and semantically relevant keyword matching are two extended features that would enrich the architecture under development in this work.

As a precursor to any work integrating this system with the SoFAR activity, it is likely that the subscription facilities permitted by TSpaces could be used to add additional functionality to the system. For example, by implementing a module that listens to instances of new links being inserted into a linkbase, the authoring event can be replicated on another DLS instance, thus providing a passive mechanism for linkbase mirroring. The target DLS in this case does not necessarily have to be another TSpaces DLS implementation, just one that implements the LinkService interface as defined.

8 Summary

The pervasive computing infrastructure has many research challenges, including scalability of interactive applications. In this paper, we have illustrated a holistic approach that enables scenarios to be developed to define and exercise these enticing challenges. By extending the distributed link service into this pervasive setting we hope to establish a pervasive information fabric to enable a new class of applications.

The prototypes developed within the presented scenario are designed under the premise that centralised control (i.e. a "Meeting Manager") is not acceptable, that the infrastructure should be self-forming as much as possible. Local processes, including the user, always manage local data such that access permissions, publication rights and document selection can be controlled with a high degree of confidence – especially important in commercial meeting environments.

This research is near-term in that enabling technologies for local area wireless networking and low-juice, medium-power handheld devices are available now, permitting ad hoc environments incorporating laptops, pads and PDAs with wireless interconnect. One key requirement at present, however, is the efficient management of bandwidth. The architecture under development here is flexible such that rather than entire documents being transferred between participant devices, either links to (in the case that the document is published on an accessible Internet) or summaries of the document can be generated and transferred in situ. The longer-term aspect of this is that as current trends continue, the wireless bandwidth will grow as power consumption decreases and bigger documents/media types emerge to "use up" the advances, this architecture can still apply.

Acknowledgements

This research is partially supported by the EPSRC LinkMe project and the IBM University Partnership Programme. We are grateful to our colleagues at IBM Hursley Park, particularly Adrian Colyer and Colin Bird, for helpful discussions. The LDAP and query routing experiments were supported by British Telecom Laboratories and we are grateful to Nigel Walker for his participation in this exercise.

References

1. L.A. Carr, D.C. De Roure, H.C. Davis, and W. Hall. Implementing an open link service for the world wide web. *World Wide Web Journal*, 1(2), 1998.

2. D. De Roure, N. Walker, and L. Carr. Investigating link service architectures. In *Proceedings of the Hypertext Conference HT'00*, May 2000.
3. D.C. De Roure, L.A. Carr, W. Hall, and G.J. Hill. Enhancing the Distributed Link Service for multimedia and collaboration. In *FTDCS 97 - Proceedings of the Sixth IEEE Workshop on Future Trends in Distributed Computing Systems*, pages 330–335. IEEE, October 1997.
4. S. El-Beltagy, D.C. De Roure, and W. Hall. A multiagent system for navigation assistance and information finding. In *PAAM 99 - Proceedings of the Fourth International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, pages 281–295, April 1999.
5. WAP Forum. Wireless Application Protocol. `http://www.wap.net/`.
6. E. Guttman, C. Perkins, J. Viezades, and M. Day. Service location protocol (SLP), version 2. Technical Report Request For Comment number 2608, Internet Engineering Task Force, June 1999.
7. Multicosm Ltd. `http://www.multicosm.com/`.
8. Sun Microsystems. The Jini lookup service specification (LU), version 1.0.1, November 1999. `http://www.sun.com/jini/specs/lookup101.html`.
9. D.E. Millard, L.A.V. Moreau, H.C. Davis, and S. Reich. FOHM: A fundamental open hypertext model for investigating interoperability between hypertext domains. In *Proceedings of the Hypertext Conference HT'00*, May 2000.
10. L.A.V. Moreau, N.M. Gibbins, D.C. De Roure, S. El-Beltagy, W. Hall, G.V. Hughes, D.W. Joyce, S. Kim, D.T. Michaelides, D.E. Millard, S. Reich, R.H. Tansley, and M.J. Weal. SoFAR with DIM agents: An agent framework for distributed information management. In *The Fifth International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents*, Manchester, UK, 2000.
11. J.R. Smith, R. Mohan, and C. Li. Scalable multimedia delivery for pervasive computing. In *Proceedings ACM Multimedia 99*, pages 131–140, Orlando, October 1999.
12. M. Spreitzer and M. Theimer. Scalable, secure mobile computing with location information. *Communications of the ACM*, July 1993.
13. W. Wang and J. Haake. Supporting user-defined activity spaces. In *Proceedings of ACM Hypertext'97*, Southampton, April 1997.
14. W. Wang and J. Haake. Flexible coordination with cooperative hypermedia. In *Proceedings of the ninth ACM Conference on Hypertext and Hypermedia*, pages 245–255, Pittsburgh, June 1998.
15. M. Weiser. The computer for the twenty-first century. *Scientific American*, pages 94–104, September 1991.
16. M. Weiser. Some computer science problems in ubiquitous computing. *Communications of the ACM*, pages 74–83, July 1993.
17. P. Wyckoff, S.W. McLaughry, T.J. Lehman, and Ford D.A. TSpaces. *IBM Systems Journal*, 37(3), August 1998.