A Recursive Software Architecture for Location-Aware Services

Alan Dearle[1], Graham Kirby[1], Ron Morrison[1], Kevin Mullen[1], Yanyan Yang[1], Richard Connor[2], Paula Welen[2], and Andy Wilson[2]

[1] School of Computer Science, University of St Andrews,
North Haugh, St Andrews, Fife KY16 9SS, Scotland
{al, graham, ron, kevin, yyyang}@dcs.st-and.ac.uk
[2] Department of Computer Science, University of Strathclyde,
Livingstone Tower, 26 Richmond Street, Glasgow G1 1XH, Scotland
{firstname.lastname}@cis.strath.ac.uk

Abstract. A GLObal Smart Space (GLOSS) provides support for interaction amongst people, artefacts and places while taking account of both context and movement on a global scale. Crucial to the definition of a GLOSS is the provision of a set of services, which we term location-aware services, that detect, convey, store and exploit location information. We first describe a framework (ontology), using a small set of concepts, for defining a GLOSS. This allows different services to be implemented without duplication of the basic mechanisms, and abstracts over specific details of the technologies used, thereby accommodating both heterogeneity and evolution. Secondly, we introduce a set of location-aware metaphors that are defined in terms of the GLOSS concepts. Thirdly, we propose a recursive software architecture to support the implementation of the metaphors within the framework. Finally we outline how the software architecture may be applied recursively to provide scalability for location-aware services in the global context.

1 Introduction

The ubiquitous computing paradigm has the goal of providing information and services that are accessible anywhere, at any time and via any device [1]. Within this paradigm, a GLObal Smart Space (GLOSS) provides support for interaction amongst people, artefacts and places while taking account of both context and movement on a global scale.

The "SmartHome" and "SmartOffice" scenarios [2], which consist of intelligent services that are accessible to users via handheld devices connected over short-range wireless links, are particular examples of a LOcal Smart Space (LOSS). Some attempts at building a LOSS have concentrated on intelligent configuration of an environment based on presence. For example, air conditioners and lights might automatically turn on/off to individual requirements, or blinds may open/close depending on natural light levels in the room [3]. Other applications have implemented proximate selection interfaces, where nearby objects are automatically easier to select, such as

automatically defaulting to the nearest printer in a print command [4]. The presentation of contextual information can be handled in a similar manner, where information or annotations about a particular location or object is automatically displayed to a person on entering an area. Finally, systems have been proposed that monitor a user's location and actions, and then utilise this information in an application such as an automatic diary [5].

Hitherto, research into smart spaces has almost exclusively concentrated on providing interaction paradigms that hide the computer in small and highly constrained environments (LOSSes) [6]. Such systems are mostly focused either on a particular application or on a specific sensor technology with little or no general platform or infrastructure. As such the current state-of-the-art for smart spaces is, for the most part, complex, expensive to install, and optimised for the environment dynamics at installation time. Since environment dynamics drift over time, these installed systems frequently atrophy and require reconfiguration, often at great expense.

Here we concentrate on the global context (a GLOSS) and in particular location–aware services. This has been made feasible by the recent advances in sensor and location technologies, the increased performance of miniature and embedded devices, and the increased availability of ubiquitous and wireless networking. Our approach is to define a GLOSS framework (ontology) [7] using a small set of concepts. The framework allows different services to be implemented without duplication of the basic mechanisms and also abstracts over specific details of the technologies used thereby accommodating both heterogeneity and evolution. Using the framework we introduce a set of location-aware metaphors and show how they interact with these concepts.

Additionally, we introduce a recursive software architecture that facilitates the implementation of the location-aware metaphors within the framework. Finally we outline how the software architecture may be applied recursively to provide scalability for location-aware services in the global context.

1.1 GLOSS Concepts

The GLOSS framework describes a universe of discourse for understanding global smart spaces. The key concepts are **people**, **artefacts** and **places**. For the moment we will not model people and artefacts in great detail, save that: a particular type of artefact is identified, the **conduit** which is associated with a person, being an artefact that is part of, or in communication with, the general (connected) GLOSS fabric; and that people have **profiles** that define their context.

Of particular importance to GLOSS, given its emphasis on *global* and *spaces* and our interest in location-aware services, is the treatment of geo-spatial concepts. Three separate concepts are identified: **physical location**, **region** and **symbolic location**, unified by the general term **where**, deliberately selected to avoid the inevitable connotations associated with more precise terms. A physical location is a single fixed point in space. One particular type is a **coordinate**, expressed using a coordinate system (as opposed to some more general description). A region is an extended area or volume, fixed in space, such as a city. A symbolic location is a potentially mobile logical entity

that occupies a region that may vary over time, such as a car or a train. A number of other auxiliary concepts are significant in the GLOSS framework but here we only require a profile, which defines the context for a person.

A simplified UML model for location information is given in **Fig 1**.

Fig 1: UML class diagram for location information

1.2 Location–Aware Metaphors

Central to the definition of a GLOSS is the provision of a set of services, which we term *location-aware services*, that detect, convey, store and exploit location information. To illustrate these services we introduce three metaphors termed *Radar*, *Trail* and *Hearsay*. These are described as compositions of the GLOSS concepts and form a basis for the design of practical tools to support mobile users in a global context.

1.2.1 Radar

Radar was originally proposed [8] as follows:

> "*This is a tool that will give you an overview beyond the immediate environment. With this tool you will be able to locate low and high densities, crowds and groups in a larger area. This can help you find the special gap of freedom, the emptiness, you sometimes lack in a public environment. This instrument can search the streets and spaces for you on a hunt for either noise or silence.*"

Radar can be described in terms of the primitive concepts as:

```
Radar: Time X Region -> Set[GLOSSObject]
```

That is, radar provides a means for the user to discover the significant objects within a bounded region at a given time. The region describes the radar horizon.

1.2.2 Trails

Trails were originally proposed [8] as follows:

> *"Trails: an ancient way of finding your way around in an area, known or unknown. To find and navigate the way now you follow static landmarks and signs; if you lose sight of the signs or find something "off-trail" interesting, you lose your trail. We propose a Trail that is plastic, a trail that changes, grows and evolves with you. A morphic trail that guides you in your daily life. This is also a tool to be shared with other people. Trails could be borrowed, given, bought or swapped."*

Three distinct varieties of trails are modelled: **observed trails**, **archetypal trails** and **intentional trails**. To give the flavour of our reasoning we will only describe observed trails here.

An observed trail is an ordered sequence of observations of a person or artefact, each recording a time, a place (coordinate, region or location) and optionally some additional information. It can be described in terms of the primitive concepts as:

```
ObservedTrail: GLOSSObject X Time X Time X
    Sequence[Where X Time X Information]
```

Thus an observed trail contains an observed entity (a specific person or artefact), a start time and end time, and a sequence of observations, each comprising an individual observation in space and time, together with some optional additional information.

1.2.3 Hearsay

Hearsay was originally proposed [8] as follows:

> *"This is an intimate, sensitive tool that will be there to remind and pick up small notes in the environment left for you. It will make sure that you will only find the message left for you if the context is right. Some messages you will never find. We suggest a new form of "snail mail" that will give you the same experience the sender had when the message was composed. Posted or left in the global environment the message waits at the same place to be delivered at the right time for whom it's left for. A mail where time is not an option but the context is."*

Hearsay can be described in terms of the primitive concepts as:

```
Hearsay: Time X Where -> Set[Information]
```

That is, hearsay provides a means for the user to send and receive messages that are delivered only when the receiver enters a specific **where**.

1.3 Use-Case Hearsay Scenarios

To motivate our design decisions on the software architecture for implementing location-aware services we introduce two use-case scenarios based on hearsay. In general, two services are required for hearsay:

- placement of hearsay
- delivery of hearsay

Hearsay placement allows a GLOSS **user** to insert a message into the GLOSS fabric at any point, parameterised by a **where** and a **profile**. The **where** describes the geospatial region in which the message should be delivered, and the **profile** restricts delivery to certain GLOSS users.

In both use-cases we will concentrate on the delivery of hearsay except to note that the placement can originate from any GLOSS node. In each use-case the hearsay will be placed (stored) on the GLOSS node that corresponds to the **where** in which the hearsay should be delivered—if there is such a node—or, otherwise, on the GLOSS node whose **where** most closely contains the delivery **where**.

Hearsay delivery is triggered when a user enters a **where** that has associated, relevant hearsay attached to it. There are thus two separate implementation aspects: how the event describing that user movement reaches the hearsay delivery service, and how the hearsay is then delivered to the user. The communication channels used between the user and the GLOSS fabric may be the same in each case, or different.

The first use-case allows hearsay to be delivered at any point within a GLOSS-aware **where**, while the second allows hearsay to be delivered at a particular GLOSS-aware **coordinate**. The use-cases also illustrate different ways in which user movement events may be routed through the GLOSS fabric.

1.3.1 Use-Case One

In our first scenario, Anna places some hearsay relevant to Bob (about a café) at a **where** (the street containing the café), due for delivery when Bob enters the street.

Bob has a PDA that is GPS-enabled, and communicates with the GLOSS fabric via SMS over GSM. **Fig 2** shows Bob's PDA sending SMS messages, containing his GPS coordinates, to his SMS gateway (in Brussels) since Bob is from Belgium. The SMS gateway calls an agent to process the GPS data and informs the street, identified from the GPS data, that Bob has entered the street. Once Bob's profile is matched with that of the hearsay, the hearsay is delivered to him.

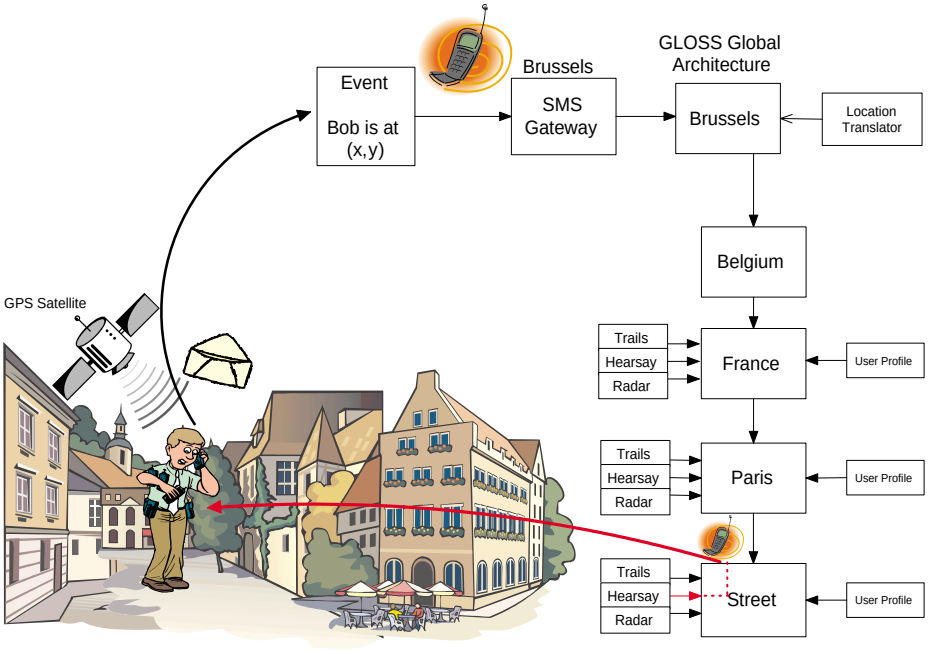

Fig 2: Hearsay delivery anywhere in a GLOSS-aware **where**, with location detection by user

Fig 2 gives a hint of the infrastructure that may be required to support a GLOSS. We will develop this later, but emphasise here that the location information enters the GLOSS fabric at a remote point (Brussels) and that the hearsay is delivered when Bob enters any part of the street.

1.3.2 Use-Case Two

Our second scenario involves the staff at the Pompidou Centre placing hearsay relevant to all art lovers at a **coordinate** (the *Eli Lotar III* sculpture [9]) for delivery at precisely that **coordinate**.

Fig 3 shows Bob receiving hearsay about the statue in the museum. The hearsay is associated with the statue **where**, and it should only be delivered when Bob is in the immediate proximity of the statue. Here Bob's PDA contains a passive TIRIS tag [10] that enables it to be detected within a short range. The TIRIS tag informs the museum that Bob is at the coordinate.

Once Bob's profile has been matched with the hearsay in the museum, the hearsay may be delivered to Bob's by any means (in this case by radio-ethernet-enabled PDA).

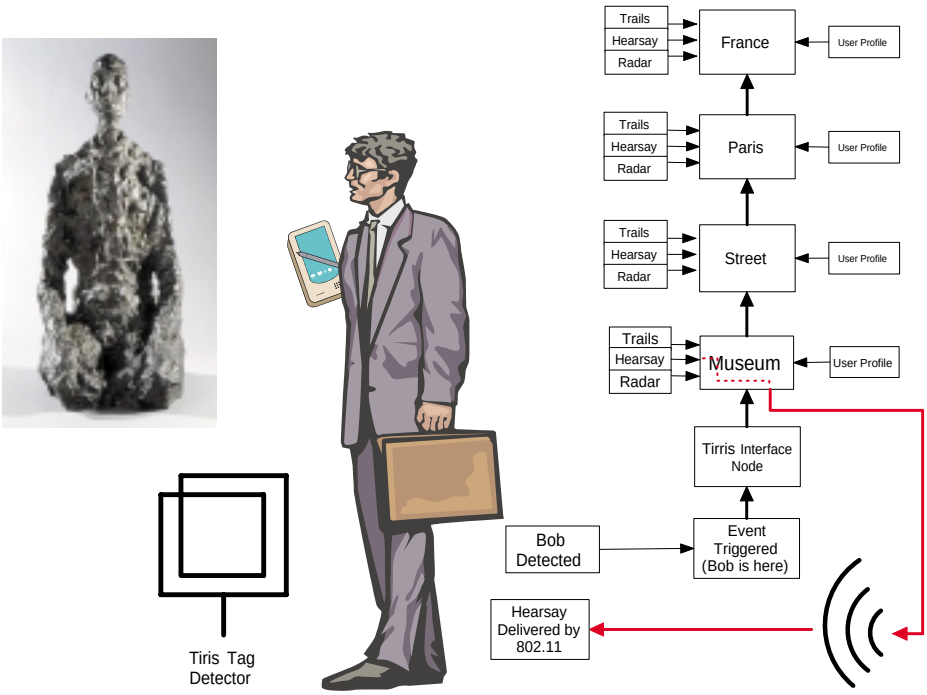

Fig 3: Hearsay for delivery at a GLOSS-aware **coordinate**, with location detection by fabric

Fig 3 again hints at the GLOSS infrastructure but this time with the emphasis on location information entering the GLOSS fabric locally, and that the hearsay is delivered when Steve arrives at a specific coordinate.

2 Implementation Dimensions

When engineering a global smart space there are many implementation dimensions that must be considered, including:

- location detection
- storage
- computation
- communication
- identity of users and devices
- user interface
- format and content of information

Since we concentrate here on location-aware services, we only discuss the first four dimensions.

2.1 Location Detection

Two broad classes of location detection devices are considered:

- those where the artefact being detected has knowledge of its position
- those where the external fabric has knowledge of the artefact's position

An example of the former is a GPS device being carried by a user. The GPS device is aware of the position of its user but that knowledge is not available in the environment without it being actively propagated. An example of the second is a RF tag such as TIRIS, which is capable of detecting a person or artefact carrying a tag, but the person or artefact has no knowledge of their own position. Hybrids of the two location detection mechanisms exist, for example IEEE 802.11 [11] cards may be used to detect other cards in their proximity.

A GLOSS should be able to exploit whatever location detection technology is available. This will mean using different technologies in different contexts.

2.2 Storage

The cost, bandwidth and capacity of storage, and the durability of data stored vary considerably from device to device. Mobile phones are at one end of the spectrum having relatively low volumes of storage with data typically being stored in one place—the device itself. Server machines are at the other extreme with high volumes of low cost storage featuring high bandwidth access, which is easily and cheaply replicated. Data resident on servers tends to have better availability characteristics than data resident on hand-held devices.

A GLOSS may store information in the locations most appropriate for the use-case scenarios being supported. This may necessitate maintaining cached copies of client data on servers in order to (a) obviate high communication costs to devices, (b) have data available close to where computation will occur, and (c) make data available when client-devices are disconnected.

2.3 Computation

There are three possibilities as to where computation may occur in a GLOSS, namely at the client, at the server, and in the network. Computation at the client is usually limited in terms of CPU power, storage capacity, I/O bandwidth and battery capacity. We therefore normally discount the client as a site for heavy computation such as performing complex matching or database queries. However computation at the client is necessary for a number of tasks including: user event notification, the integration of data from devices (such as GPS receivers) and the processing of information sent to the device. None of the power, storage, bandwidth or processing limitations of the client exist at the server thereby making it more suitable for heavy computation. We recognise that processing may also take place in the network should computational resources be available there.

The GLOSS infrastructure needs to be able to find the most appropriate use of whatever computational power is available. This will often mean performing matching and searching tasks on servers and conserving battery life on handheld devices by only using them for the presentation of data. However, the placement of computation will always be balanced by the communication cost of moving data to an appropriate place for computation to occur.

2.4 Communication

In a GLOSS, a variety of communication mechanisms is required for inter-server, inter-client, and client-server communication. Inter-server communication is dominated by Internet protocols and this seems likely to continue. However, a variety of technologies are available for (mobile) inter-client, and client-server communication including GPRS, SMS, TCP/IP via a modem connection, 802.11 (using TCP/IP) and Bluetooth. These technologies have quite different characteristics in terms of end-point connections, cost and bandwidth.

A GLOSS may accommodate any communication technology; in this paper we show how 802.11 and SMS may be used to facilitate client-server communication.

3 Implementation of the Location-Aware Metaphors

3.1 Hearsay

Hearsay placement is a storage activity. However this begs the question as to where is the best place to store the association between a location and some information associated with it. There are a number of obvious choices:

- on the depositor's machine
- on a central server offering a hearsay service
- broadcast the association and store the association somewhere on the network
- on a machine associated with the location referred to in the hearsay message

The first of these choices is unattractive since it makes it hard to find appropriate hearsay for delivery when a user enters a **where**. If this choice is made for depositing hearsay, the discovery of hearsay becomes a distributed network query without any obvious criterion for the termination of the query in the case of success or failure. The second choice, that of a central server, is easy to manage but has limited potential for scalability. In particular it is unlikely that it will scale to a global number of users. Broadcasting or publishing the association on the network is an attractive option. However, some mechanisms must be provided so that the hearsay will be stored in a location where it can be discovered. Associating hearsay with the location referred to by that hearsay is also an attractive option. This requires a hearsay server to be associated a particular geo-spatial area and for some algorithm to exist whereby that server can be found when a location is supplied. Later in this paper we outline how it is possible to build a peer-to-peer network containing rules that route messages to an appro-

priate server. This has the benefits that it is relatively simple to add additional servers, making it scalable, and allowing hearsay discovery to be relatively simple.

> **Key decision**: In our GLOSS prototype we have chosen to store hearsay on servers that are associated with particular geo-spatial areas. Each server will normally be located within the geo-spatial area with which it is associated.

Hearsay delivery is triggered when a user enters a **where** that has associated, relevant hearsay attached to it. The implementation of hearsay delivery has four technological dimensions:

- where the detection of the user's location takes place
- the communication between where the detection has occurred and where the profile matching takes place
- where the computation matching the user's location to the hearsay takes place
- how the user is notified of the hearsay

The detection of the user's location may occur in a number of places depending on the technological mix. In first use-case scenario, the event describing the user's movement is triggered by the GPS device and is propagated to the GLOSS infrastructure via an SMS message. The SMS message is sent to a SMS gateway that is likely to be located on a machine that is not in proximity to the sender of the message or to the location of the hearsay information. In the second use-case scenario, the detection occurs in the fixed infrastructure when a tag reader detects a tag being carried by the user. In this case it is highly likely that the tag reader is associated with a computer that is in the proximity of the user, and may be in close proximity to a server on which the hearsay is stored.

In both cases the machines receiving the detection event must do something with it—again there are a number of choices. Each machine could:

- send the event to the hearsay service
- send the event to the user's home machine
- broadcast the event

Before examining these choices, it is important to appreciate that the location detection may be driving services other than hearsay—for example, it may also be feeding a trails service and a radar service. Consequently, the first choice, which appears to be the obvious candidate, may not be ideal, since the radar and trails services may not be co-located with the hearsay service.

The second choice assumes that each user has a designated *home* machine somewhere on the network. This could be the user's web server for example. This is an attractive option—it scales well—servers can be added on demand as users are added. It means that every user has a well known home to which all messages about that user can be sent. However, it does not solve the problem of hearsay matching—it only delays it. When the message is sent to the user's *home* machine, another message must be sent elsewhere for the matching to occur.

In the third choice, the event may be broadcast and used by whichever services are interested in it. However, if every server on a network were broadcasting every event,

the network would quickly saturate with messages of no interest to most servers. Another problem with uncontrolled broadcast is that the messages would propagate until they had reached every node in the network—this is wasteful of resources and it is unlikely to deliver timely data. Some mechanisms therefore need to be provided to route messages to those servers that are interested in the events. We return to this later.

Next, we turn to the question of where the computation matching the user's location to the hearsay takes place. There are three main contenders:

- on the user's conduit
- on the user's home machine on the network
- on the machine storing the hearsay

For matching to occur on the user's conduit, the user's location, profile and all the hearsay from the hearsay server would have to be sent to the conduit machine. This wastes two precious resources on the conduit: network bandwidth and battery life. In addition to being wasteful, the memory and CPU resources on the conduit are likely to be limited, making it a poor choice. If matching is to occur on the user's home machine (if there is one), a message must be sent to the location on which the hearsay is stored. This message could request all the hearsay on the server (which may result in a large amount of network traffic) or it could send the user's profile to the hearsay server, in which case the matching would occur on the machine storing the hearsay.

> **Key decision**: In our GLOSS prototype we have chosen to perform the computation matching hearsay to the user's location on the machines storing the hearsay.

If the user profile is large (we have empirical evidence to suggest it would not be) the option we have chosen appears a poor one. However, we believe that it is not, due to the *region transition hypothesis* described in the next section.

The last issue is the delivery of hearsay to the user. Once the hearsay service has matched the user and the hearsay, it must send the hearsay to the user. There are two choices for this delivery:

- it may be sent to the user's home machine
- it may be sent directly to the user's conduit

The first choice has the advantage that a fixed server is much more likely to be reachable than the user's conduit. It is therefore a good candidate for somewhere to send information in the event of not being able to contact the conduit. This approach suffers from lack of geo-spatial proximity. The user's home machine may not be in proximity to the user or the machine storing the hearsay. Sending a hearsay message to the user's home machine is unlikely to make efficient use of network bandwidth. Sending the message directly to the user's conduit makes the best use of available network bandwidth. However, the user may be connected to the GLOSS infrastructure using a variety of different technologies. Some abstraction over the technologies is required to insulate the hearsay services (and other services) from network.

Key decision: Each user's profile contains information about how to contact that user. This may include alternatives to deal with failure.

If the hearsay service has access to the user's profile it may send the hearsay message directly to the user using whatever services are at its disposal. This may involves sending a message to a proxy server (in the case of an SMS attached user) or sending the message directly (in the case of a TCP/IP 802.11 connected user).

3.1.1 Region Transition Hypothesis

The *region transition hypothesis* states that as the granularity of the geo-spatial region increases, the total frequency of transitions between regions falls. Globally, there are billions of transitions occurring every second at a fine granularity: a user in France enters a café at the same instant that a user in the UK leaves a shop and a user in Ireland goes into a bar. There may be 50 shops, cafés and other GLOSS locations in a street comprising a region, with frequent transitions between these locations, but the street itself has only a few access points (its junctions with other streets) and users move between streets less frequently. Similarly, the overall frequencies of transitions between cities and countries are still lower.

The *region transition hypothesis* has implications for the GLOSS architecture and for caching. If servers are organised into a tree, with each server being associated with a smaller geo-spatial area than its parent, data may be effectively cached at the nodes of the tree and obviate the need for high volumes of data, for example profile data, to be shipped. However, tree architectures are not scalable—as the root node is approached, the volumes of traffic increase—some mechanism is required to prevent network saturation close to the root. We will return to this problem later.

3.2 Approach to GLOSS Service Implementation

The use-case studies highlight two points:

- A local architecture is required to permit GLOSS clients and servers to be constructed. Both clients and servers are essentially data-flow architectures. Data arrives at a user's device from several sources—communications devices, position servers etc.—and passed to elements for processing which might include displaying information to a user, sending it to another device, or storing it for future presentation or processing. Similarly on a server, data arrives from several devices is processed and delivered to other devices for further processing or presentation to the user.
- A global architecture is required to mediate the global flow of information between clients and servers.

4 Local Architecture

Our prototype GLOSS architecture is designed with the following motivations:

- to abstract over any particular technology
- to make GLOSS components independent of each other
- to allow components to be assembled into GLOSS applications

To permit GLOSS applications to be constructed, we arrange components into pipelines and register components with event buses. The local architecture is based on XML pipelines and an XML event bus. To abstract over location, GLOSS clients transparently pass messages to a server and vice-versa by plugging appropriate components into the pipelines and event buses of the client and server. A high level overview of this architecture is shown in **Fig 4**.

Fig 4: Local architecture

Data enters the system from various devices in various formats. Most components in the system are compliant with the *GOPipe(GLOSS Object Pipe)* interface. This is a component that takes XML encoded GLOSS objects and processes them. *GOPipe* objects can be assembled into pipelines of processing components. An interface *ExtensibleGOPipe* is provided which provides *GOPipe* functionality and permits downstream *GOPipe* objects to be registered with it. In many GLOSS scenarios there is a requirement to distribute information about GLOSS actors and artefacts to several components. To accommodate this, an *EventBus* interface is provided. It presents the *GOPipe* interface so that it may be inserted into a pipeline, and permits multiple objects presenting the *GOPipe* interface to be registered with it. The components and

interfaces described above achieve the first two objectives. In order to manage collections of component instances we introduce another abstraction—an *assembly*. This is a collection of components that are linked together via pipes and buses.

4.1 Applying the Local Architecture to the Use-Cases

To accommodate use-case one, a pipeline is assembled on Bob's conduit containing the GPS device, an adapter (not shown in the diagram) and an event bus. Two *GOPipe* objects are attached to the event bus: a hearsay user interface tool and an SMS device. The SMS device is also attached to the event bus, permitting it to supply the bus with non-local events. In the use case, the GPS device passes events via the adapter to the event bus, which passes them to the GPS device. This device sends the events to Bob's SMS server located in Brussels.

On the SMS server, another pipeline has been assembled, capable of processing incoming SMS events. In this instance of the architecture, a location service (another *GOPipe* compliant object) is plugged into the event bus. It passes location information to an IP device which results in the message being passed to a street server located in the same street as Bob.

A third pipeline has been assembled on the street server. It contains a *GOPipe* complaint IP module to supply GLOSS Object Events to the pipeline that contains an event bus with the hearsay service plugged into it. The hearsay service determines that Bob is in the street (determined by the event sent to it), and that a café recommended by Anna is close by (the hearsay). A message needs to be sent to Bob to inform him of this fact. The message could be sent to Bob using a variety of mechanisms. To abstract over these mechanisms, a *GOPipe* compliant proxy for Bob is provided. This component will send a message to Bob using the most appropriate technology—in this case, an SMS message.

The GSM/SMS card on Bob's conduit receives the SMS message, where it is passed to the event bus and from there to the hearsay user interface tool, which will present the hearsay to Bob.

In use-case two, Bob is at the Pompidou Centre and looking at the *Eli Lotar III* sculpture by Alberto Giacometti. In this scenario Bob is wearing a tag, which uniquely identifies him, and his PDA is equipped with an 802.11 card. Each exhibit in the exhibition has a tag reader that detects tags in its proximity. For simplicity we assume that there is a server associated with each exhibit, and a GLOSS server in the museum.

Bob's tag is detected in the proximity of the exhibit by the tag reader, which sends a message to a serial port which is connected to a *GO* pipeline. After a *GOAdapter* has processed it, the event (containing Bob's unique ID) is passed to an event bus where it is passed to the only component connected to the bus—an IP device that sends the event to the museum server.

The IP device on the museum server is connected to another pipeline and the event is passed to the hearsay service. This service locates Bob's profile (using mechanisms not described here) and determines what information he may like to know about the statue which he is looking at. This is sent via another object implementing the *GOPipe* interface, via 802.11, to Bob's PDA. In the manner described above, Bob is alerted to the new information about the statue.

5 Global Architecture

We have described a local architecture for GLOSS conduits and servers. Next we briefly outline how these components may be organized into a global architecture for smart spaces. As described above, tree architectures, such as the one shown in **Fig 5** (a), are not scalable—as the root node is approached, the volumes of traffic increase [12]. An alternative to a hierarchical architecture, shown in **Fig 5** (b) is a peer-to-peer (P2P) architecture composed entirely of peers, with no node in the network assuming any more responsibility than any other. Instances of this style of architecture, for example Gnutella [13] and Kazaa [14], have recently become popular for file sharing applications. Many of these architectures depend on searching a limited network centric horizon of machines governed by a number of network hops from the point of query. Such an architecture is not perfectly suited to an environment in which messages require routing to a specific destination. For example, in the SMS hearsay example, a SMS location message must be routed to an appropriate hearsay server to enable matching to occur. However, it may be injected into a GLOSS network a considerable distance (in terms of both hops and geo-spatial proximity) from a server storing hearsay.

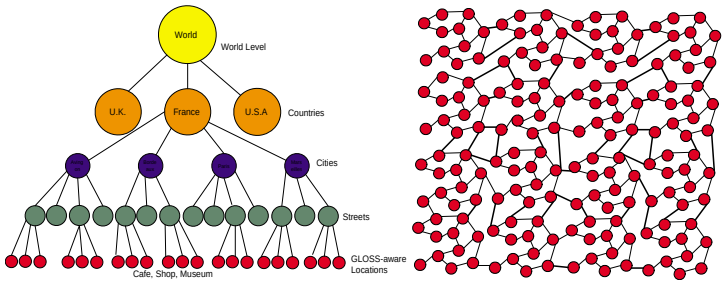

Fig 5: Possible global architectures: (a) hierarchical and (b) peer-to-peer

The network supporting the GLOSS infrastructure must therefore provide some mechanism to route messages to an appropriate location. Hybrid architectures offer the opportunity to tailor topologies and protocols in such a way that locality knowledge may be exploited. In [12], Carzaniga et al state that an acyclic peer-to-peer architecture with subscription forwarding appears to scale well and predictably under all circumstances and is likely to represent a good choice to cover a wide variety of scenarios. Consequently, we are currently exploring a global peer-to-peer architecture consisting of a hierarchy of peers such as that shown in **Fig 6**.

In addition to avoiding network bottlenecks, a hierarchy of peers is well suited to the geo-spatial nature of GLOSS queries. As shown in **Fig 6**, it is relatively easy to partition the world recursively into non-overlapping regions. These may be mapped on social, economic or organisational boundaries. The servers need not be co-located with the geo-spatial region that they represent—the only necessity is that peers understand the peering relationships. A final benefit of this architecture is that it is capable of evolving to cope with stress placed upon it.

Partitioning the network into a hierarchy of peers makes the routing of messages to appropriate locations with the GLOSS infrastructure relatively straightforward. As proof of concept we outline how the three technological aspects of communication, computation and storage in the two use-case scenarios relate to this architecture.

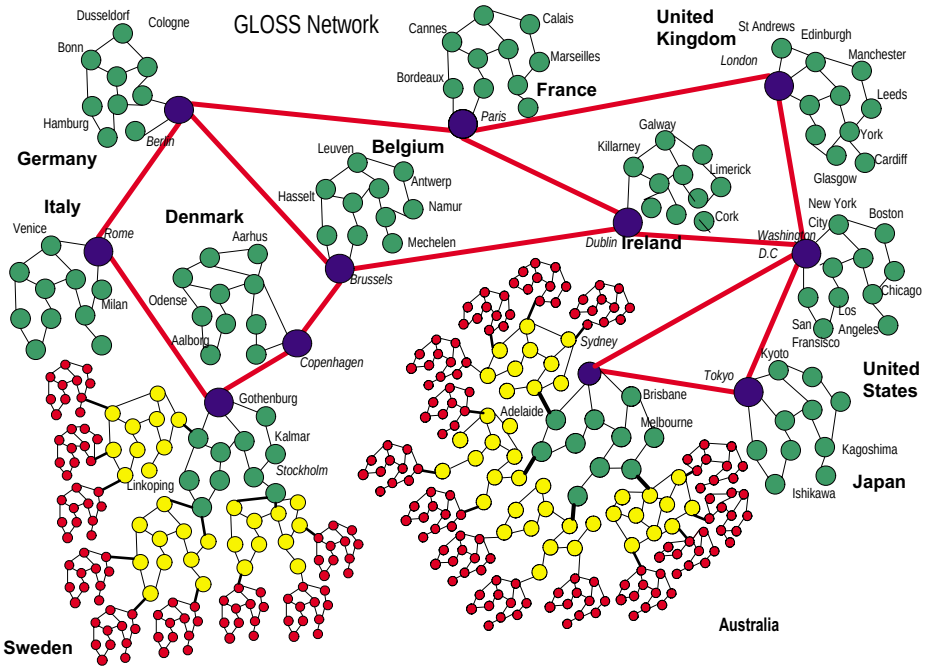

Fig 6: Hybrid global architecture

In the first use-case, Bob's conduit relays his position to an SMS server located in Belgium. As described above, the GLOSS object pipeline on that server examines location and determines that it must be sent elsewhere. Since the SMS server is part of a P2P network, it does not require a final destination address for the message. However, it must determine if the message should be sent to its children, its parent, broadcast to its peers or dealt with locally. This calculation may be performed by comparing the geo-spatial location of the user with that being managed by the server. In the case of the message about Bob's location, the location in the message is about a location in Paris so it is passed to the Brussels server which either sends it directly to the Paris server (if it has knowledge of this server and the geo-spatial region it manages) or broadcasts it to its peers. When the Paris server receives this message, it is passed down the hierarchy until it reaches the street server where the computation matching the hearsay to Bob can occur.

The hierarchy of peers dovetails well with the region-transition hypothesis. The hierarchy gives a nested set of locations in which Bob's profile may be cached (stored). Assuming Bob's profile is stored at his *home* server, when Bob is first detected in France, a request for Bob's profile will be propagated up the network and down to Bob's home in a manner similar to that described above for hearsay messages. The

reply may be cached in servers at multiple levels in the network to obviate the need to repeatedly fetch this information. A similar technique under-pins the replicated document cache used in Freenet [15]. The use of different cache policies at different levels in the hierarchy is likely to be beneficial. For example, leaf servers (where change is fast) might only cache data for a short period but servers close to the root (where change is slow) might cache data for longer periods.

In use-case two, the location information is injected into the GLOSS architecture close to the location at which the hearsay is stored. In this case, the location event needs to propagate up the tree of peers. As propagation occurs, computation matching hearsay and the location of user's occurs, triggering hearsay delivery to users as described above. If knowledge of Bob's location is cached in the hierarchy as described above, at some point the message will reach a node that already knows that Bob is in this locale. At this point the message need not propagate up the tree any further. If a match is not made as the message is sent up the tree, subject to policy, the message might propagate back to Bob's home where his current location could be stored.

6 Status of Implementation

At the time of writing, instances of the GLOSS Object Pipeline architecture are running on a mobile device (currently a laptop for ease of prototyping) and a server. A pipeline running on the laptop integrates a GPS device from Garmin, a Nokia GSM phone card and a prototype hearsay user-interface. We have developed Java based *GOPipe* interfaces to the GPS device which supplies the pipeline with location information, and another to the GSM device capable of sending and receiving SMS messages. The sever pipeline contains a pipeline containing an SMS server and supplies messages to an event bus. Currently, this bus supplies a Trails capture tool with location data.

We are currently implementing the necessary algorithms for global message routing and plan to integrate these with the local pipeline architectures shortly.

7 Conclusions

We have outlined a framework for the description of Global Smart Spaces that supports interaction amongst people, artefacts and places while taking account of both context and movement on a global scale. We have also identified a set of location-aware services, which detect, convey, store and exploit location information. These services are introduced using the radar, hearsay and trails metaphors and we have explained their relationship with the framework concepts. The essence of the paper is the introduction of a recursive software architecture that facilitates the implementation of the metaphors within the framework. For scalability we show how the software architecture may be applied recursively to provide location-aware services in the global context.

8 Acknowledgements

The early part of this paper is based on, as yet unpublished, research conducted jointly in the EU-funded GLOSS project (5th Framework IST-2000-26070) [16], with partners at Strathclyde University, Trinity College Dublin, and Université Joseph Fourier. This will be published in full separately.

Joëlle Coutaz and Gaëtan Rey, in particular, contributed to the GLOSS ontology. Mark Dunlop and Paddy Nixon developed a prototype SMS/Hearsay implementation. Peter Barron and Álvaro Rebón-Portillo wrote parts of the current recursive software architecture prototype.

The work was also supported by EPSRC grants GR/M78403 and GR/M76225, “Supporting Internet Computation in Arbitrary Geographical Locations”.

9 References

1. Weiser M. The Computer for the 21st Century. Scientific American (1991),September 94-104
2. Pentland A. Smart Rooms. Scientific American (1996) 274,4 68-76
3. Elrod S., Hall G., Costanza R., Dixon M., Des Rivieres J. Responsive Office Environments. Communications of the ACM (1993) 36,7
4. Schilit B., Adams N., Want R. Context-Aware Computing Applications. In: Proc. Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, USA (1994) 85-90
5. Lamming M., Brown P., Carter K., Eldridge M., Flynn M., Louie G. The Design of a Human Memory Prosthesis. The Computer Journal (1994) 37,3 153-163
6. Nixon P., Lacey G., Dobson S. Managing Interactions in Smart Environments. In: Proc. Workshop on Software Engineering for Wearable and Pervasive Computing, 22nd International Conference on Software Engineering (ICSE2000), Limerick, Ireland (2000) 251
7. Working Document on Gloss Ontology. GLOSS Consortium (2002)
8. Munro A., Welen P., Wilson A. Interaction Archetypes. GLOSS Consortium (2001)
9. Giacometti A. Eli Lotar III, `http://www.rmn.fr/gb/01rmn/01missions/giacometti.html`
10. Texas Instruments. Radio Frequency Identification Systems, `http://www.ti.com/tiris/` (2002)
11. IEEE. IEEE 802.11 Wireless Local Area Networks, `http://grouper.ieee.org/groups/802/11/` (2002)
12. Carzaniga A., Rosenblum D.S., Wolf A.L. Design and Evaluation of a Wide Area Notification Service. ACM Transactions on Computer Systems (2001) 19,3 332-383
13. Kan G. Chapter 8: Gnutella. In: A. Oram (ed) Peer-to-Peer: Harnessing the Power of Disruptive Technologies. O'Reilly (2001)
14. Kazaa, `http://www.kazaa.com/en/technology.htm` (2002)
15. Clarke I., Miller S.G., Hong T.W., Sandberg O., Wiley B. Protecting Free Expression Online with Freenet. IEEE Internet Computing (2002) 6,1 40-49
16. GLOSS Consortium. Global Smart Spaces, `http://www.gloss.cs.strath.ac.uk/` (2002)