

An Architecture for Geographically-Oriented Service Discovery on the Internet

Qiyang Li

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2002

© Qiyang Li, 2002

Author's Declaration for Electronic Submission of a Thesis

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Most of the service discovery protocols available on the Internet are built upon its logical structure. This phenomenon can be observed frequently from the way in which they behave. For instance, Jini and SLP service providers announce their presence by multicasting service advertisements, an approach that is neither intended to scale nor capable of scaling to the size of the Internet. With mobile and wireless devices becoming increasingly popular, there appears to be a need for performing service discovery in a wide-area context, as there is very little direct correlation between the Internet topology and geographic locations. Even for desktop computers, such a need can arise from time to time. This problem suggests the necessity for an architecture that allows users to locate resources on the Internet using geographic criteria.

This thesis presents an architecture that can be deployed with minimal effort in the existing network infrastructure. The geographic information can be shared among multiple applications in a fashion similar to the way DNS is shared throughout the Internet. The design and implementation of the architecture are discussed in detail, and three case studies are used to illustrate how the architecture can be employed by various applications to satisfy dramatically different needs of end-users.

Acknowledgements

This thesis would not have been possible without Dr. Jay Black, who has been guiding and supporting me in more than one way during the process of writing. His encouragement and patience have been my way out of frustration and impatience, and therefore, I would like to express my most sincere gratitude to him. Mom, Dad, and my brother have also offered much-needed spiritual support, and have being extremely understanding and encouraging, even after I was unable to finish the thesis as I had planned. Above all, Susanne, who has been by my side the whole time, has been so helpful that I would not even attempt to describe everything she has done for me on this tiny page. I thank you all for your help and support!

To Susanne

Table of Contents

Abstract	iii
Acknowledgements	iv
1 Introduction	1
1.1 The Question	3
1.2 The Challenge	5
1.3 Design Goals	9
1.3.1 Interpretation of Spatial Information	9
1.3.2 Data Model for Spatial Information	10
1.3.3 Distributed and Cooperative Nature	11
1.4 Contribution	12
1.5 Thesis Organization	12
2 Related Work	13
2.1 Service Location Protocol	13
2.2 Location-Based Service Architecture	16
2.3 DataSpace	20
2.4 Unexplored Opportunities	23

3	Architecture	24
3.1	Overview	25
3.2	Kiosks	30
3.2.1	Schema	32
3.2.2	Spatial Properties	33
3.3	Geometry Managers	34
3.3.1	Coordinate System	35
3.3.2	Hierarchy of Geometry Managers and Kiosks	36
3.3.3	Approximations to Areas	38
3.4	Resolvers	40
3.4.1	Resolver-Application Communication	41
3.4.2	Resolver-Geometry Manager Communication	43
3.4.3	Resolver-Kiosk Communication	44
4	Implementation	45
4.1	Encoding of Spatial Data	46
4.2	Protocols and Message Formats	52
4.3	Concurrency Control	54
5	Case Studies	56
5.1	A Geographically-Oriented Web	57
5.2	Service Discovery in the Physical World	67
5.3	Service Discovery on the Internet	72
6	Conclusions	76
6.1	Future work	77

6.1.1	Syntax of Keywords	77
6.1.2	Enhanced Spatial Accuracy	78
6.1.3	Performance	79
6.1.4	Data Integration	79

Bibliography	80
---------------------	-----------

List of Figures

3.1	Entities in the Architecture	27
3.2	Geometry Managers and Kiosks	37
3.3	Approximation of an Area of Interest	39
4.1	A Simple DWF File	49
4.2	Square	51
4.3	Query Criteria Uploaded as <code>multipart/form-data</code>	53
5.1	Components of a Geographically-Oriented Web	59
5.2	Population of a Kiosk Database	60
5.3	Geographically-Oriented Web Browser	62
5.4	Chinese Food in Three Buildings	63
5.5	SQL Query for the Keywords “ <code>Chinese_food</code> ”	64
5.6	DNS Resource Record of <code>lj5m-sho.uwaterloo.ca</code>	70
5.7	Database corresponding to Kiosk DC 3552D	71
5.8	Properties of a Printer	74
5.9	Selection of a Postscript Printer	75

Chapter 1

Introduction

The increasing popularity of mobile and wireless devices observed today suggests an increasing level of mobility of people in a modern society. This implies that as people become more likely to travel, they also inevitably become more likely to find themselves in unfamiliar environments. In such environments, it is often difficult to receive desired services, as it is usually difficult to locate resources offering the services. The following example can be used to illustrate such a problem scenario and how the problem can be solved.

Imagine a group of unguided tourists travelling in the city of Rome, wanting to have dinner before their bus leaves in one and a half hours. Since they are not all familiar with the city, they need assistance in finding a restaurant. Essentially, their question is “Where within our immediate vicinity can we find a place serving Greek food?” Luckily, one of them has a Personal Digital Assistant (PDA) with a Global Positioning System (GPS) receiver, which is connected to the Internet via a high-speed wireless link. Since the current location of the tourist can be determined using the GPS receiver, the device can retrieve an appropriate map based on this

location. Then, the tourist can specify in which geographic region on the map the restaurant is expected to be, a customized decision made based on a combination of virtually any factors such as travelling distance, time constraints, or other personal preferences. This *area of interest* along with other requirements, *i.e.*, a place serving Greek food, can be sent to some remote server, using the Internet connection of the PDA. The result can then be sent back as a list of hyperlinks pointing to the Web pages of the restaurants meeting the requirements. Finally, the tourists can examine the pages in turn to find out more details, and make a decision on where to go.

The above scenario demonstrates the need for discovering a *physical service*, a service directly usable by a human being. Similarly, there is also a need for discovering an *electronic service*, a service available through an electronic network and therefore only usable by another device on the network. Imagine a visiting researcher reading a Web page on a small PDA screen. Since the screen is fairly small in size, it is desirable to show the same page on a larger display. Then, the researcher notices a currently unused LCD wall display. Therefore, a query is sent out from the PDA to the Internet, essentially asking the question: “What display services are offered to the room I am in?” After the PDA obtains the necessary information about how to access and control the display, the page can then be shown on the larger display.

Although the previous two scenarios appear to be two distinct types of service discovery, some underlying similarities can be spotted, allowing a single form of question to be constructed.

1.1 The Question

The question being asked always takes the form of “What *services* satisfying *non-spatial constraints* are accessible within *spatial constraints*?” As simple as it is, answering this question requires appropriate knowledge about the environment in the appropriate form. First, it is not resources, but services that are of interest, because in general, people care about services, not their sources. For instance, a tourist wanting to buy a bottle of mineral water does not care whether it comes from a convenience store or from a supermarket. Second, spatial and non-spatial data are two indispensable elements for locating a desired service. While spatial data describes geometric aspects of a service, non-spatial data describes its geometrically-independent attributes. Therefore, an appropriate service can be discovered when satisfying certain spatial and non-spatial properties. For each service, two types of entities are always involved: a *service provider* and a *service requester*.

A service provider provides a service having some properties within its *service area*, the geographic region in which the service is *intended* to be accessible. That is, when a service requester is physically located within the service area of a service provider, the requester can receive the corresponding service without further movement. For instance, a restaurant serving only walk-in customers has a service area equivalent to the physical coverage of the restaurant itself. On the other hand, a restaurant with a delivery service has a service area equivalent to the delivery area. In other words, people can eat in the restaurant or elsewhere within the service area, but either way, the service can be received without further movement of the service requester. For an electronic service, a service is often accessible from any point on the Internet. However, for practical purposes, a service is virtually always associated with some

geographic region, after practical factors are considered. Thus, the concept of service area allows a service provider to offer its service under its own terms.

Similar to the way a service provider is willing to provide services within its service area, a service requester receives a service at some point within its *area of interest*. Since this point can be anywhere within the area, the service requester must be prepared to travel to any point within its area of interest. Thus, the requester can use the area of interest to indicate the geographic region to which the user is willing to travel to receive the service. Therefore, arbitrary factors such as means of transportation, time constraints, and personal preferences can be taken into account when this area is specified.

The concepts of service area and area of interest capture the spatial aspects of the question. They are used to define where a service will be delivered from the provider to the requester, based on a mutual agreement between them. Intuitively, a service area can be viewed as a parameter representing the mobility level of a service provider, whereas an area of interest can be viewed as a parameter representing the mobility level of a service requester. Whenever there is an overlap between a service area and an area of interest, there is an agreement on how much mobility each party will provide before the service is delivered, where “mobility” can mean either physical movement or electronic movement.

The requirements on non-spatial aspects are used to distinguish one service from another so that a service provider has a means of specifying what service is offered on the one hand, and a service requester has a means of specifying what service is desired on the other. In the case of the first scenario, one service provider may be a restaurant serving Indian food, and another provider may be a bar serving alcoholic drinks and

Greek food. By indicating an attribute of “serving Greek food,” the service requester can distinguish between a desired service provider and an undesired one. The same is true for the second scenario.

In summary, in order to answer the question asked in these scenarios, both spatial and non-spatial data about services with are required. That is, both service providers and service requesters express their service terms by specifying spatial and non-spatial constraints. While the non-spatial constraints are the same in both cases, spatial constraints are specified respectively as a service area for providers and an area of interest for requesters. Thus, how to answer the question with spatial and non-spatial constraints efficiently is the challenge for this thesis.

1.2 The Challenge

There are different ways of developing an application to answer the type of question in the previous scenarios. One approach might be that each application takes care of everything related to answering the question. That is, each application develops its own data model, storing spatial and non-spatial data. This approach is not desirable for two reasons. First, the burdens on the application developers are too heavy in that they should deal with high-level, meaningful data structures, as opposed to low-level implementation details of those structures. Second, the efforts of developing data models are duplicated among the developers, and inevitably cause inconsistency and incompatibility among different applications. Therefore, information stored in one system cannot be used by another without conversion, an inefficient and wasteful scenario.

The clear alternative is to implement an architecture using a data model general

enough to support most applications in this category. As it is targeted at a broad range of applications, the infrastructure should be a service available globally so that applications can make use of it from virtually anywhere on earth.

There are at least three approaches to developing a global infrastructure. The first approach is to set up a separate and independent infrastructure. This approach is the least likely solution, as the cost involved is likely unjustifiable for the services provided. The second approach is to re-structure the Internet topologically into a geographically-oriented Internet. This approach would require any existing networks and subnets to be reorganized by geographic area. Consequently, existing routers would have to be re-configured to deliver packets using geographic-based routing. The unfortunate implication of all these proposed modifications is that any existing routers on the Internet would need to be modified or replaced by geographic-based routers. Given the current size of the Internet, it is hardly an option to commit such a drastic change as to throw away the existing Internet infrastructure and what has been built on top of it.

The third approach is to construct a new geographically-oriented infrastructure on top of the existing Internet infrastructure, allowing incremental deployment without disrupting the current functionality of the Internet. Although this is a more realistic and cost-efficient approach, it is a non-trivial task to develop and deploy this infrastructure over the Internet due to its spatial limitations, which can be seen in the structures of the Internet Protocol (IP) address space and the Domain Name System (DNS).

The IP address space is structured into five different classes of address. The current version of IP divides the 32-bit address space into classes A, B, C, D, and E using

different IP prefixes. While class D is used for multicast and class E is reserved for future experiments, an IP address subspace of millions (class A), thousands (class B), or hundreds (class C) of hosts can be obtained for internal use within an organization. Such a subspace is characterized by a unique and constant network identifier, and that can be used to locate the owner of an IP address. For instance, the University of Waterloo has a class B network with a network identifier of 129.97. This implies that any address from the university always takes the form 129.97.a.b, where a and b are both octets forming a unique host identifier on network 129.97. Conversely, a valid network identifier can be mapped back into an organization, which can then be used to locate the organization via some other means. Hence, a network identifier can be used to locate the owner of an IP address indirectly, with a granularity of organization.

Within many organizations, subnetting is common. That is, subnets are created by allocating a portion of the host identifier as the subnet identifier. For instance, the Shoshin Research Lab uses the 129.97.105 subnet within the University of Waterloo network 129.97, which means the local network authority of the university campus uses the third octet as the subnet identifier and the last octet as the host identifier. If subnetting is used within an organization and if the internal topology of the organizational network is known, the owner of an address can be located with a finer granularity, *e.g.*, the granularity of a research group.

Despite the possibility of locating an IP subspace and entities in it using the network and subnet identifiers, the IP address space itself is spatially unstructured, *i.e.*, there is no clear spatial relationship between any two network identifiers. For example, the University of Waterloo, Waterloo, Canada, has the 129.97 network.

While the successor network 129.98 belongs to Yeshiva University, New York, U.S.A., the predecessor network 129.96 is assigned to Flinders University, Adelaide, Australia — an institution literally on the other side of the world. On the other hand, Wilfrid Laurier University, a neighbouring university within walking distance of the University of Waterloo, has the 192.54 network.

While the IP address space is insufficient for handling spatial data, DNS adds a certain spatial structure into the Internet, *i.e.*, some domain name suffixes can be used to pinpoint the owner of a host name. For instance, the suffix `.uwaterloo.ca` can be used to pinpoint the owner of a domain name to the campus of the University of Waterloo. This provides the same level of granularity as an IP network identifier. However, domain names do provide an additional level of granularity unavailable in network identifiers. This new level can be seen from the suffix `.ca`, which can be mapped back into a domain name owner in Canada. Subnetting may or may not be reflected in the domain name level depending on the rule set by the local network authority, but a finer granularity can be achieved when hierarchical domain names are in use.

The ability to handle spatial data using IP addresses or domain names is in general insufficient due to the limited granularity and resolution. It is also desirable not to tie the spatial definition into the definition of any other facilities and architectures, because their semantics would likely be misinterpreted.

In summary, the challenge is how to insert an architecture into the existing Internet so that applications using both spatial and non-spatial data can be developed more readily. In an attempt to make the infrastructure as general as possible to meet various needs of applications, there are a few design goals.

1.3 Design Goals

The first and most important design goal is to design a system capable of answering queries with mixed spatial and non-spatial constraints. More specifically, the system should be able to answer queries of the form “What *services* satisfying *non-spatial constraints* are accessible within *spatial constraints*?” These queries should be answered efficiently, *i.e.*, queries involving a small geographic region should be answered promptly. Furthermore, the computing resources required to answer the queries, such as network bandwidth and CPU time, should not be excessive. If a query involves a large geographic area, a reasonable delay should be acceptable. The generality requirement can be met only when spatial information is properly interpreted, and when an appropriate data model is chosen for spatial information. Finally, the architecture can be robust and efficient only when it is distributed and cooperative.

1.3.1 Interpretation of Spatial Information

The physical location of an entity is often the most meaningful piece of spatial information about it. For instance, the physical location of a PDA can help its user to determine the services available in the surrounding environment, and the physical location of an electronic tourist guide can be used to inform a lost tourist of the current location. Various applications have tried to explore how to use the location context to adjust their behaviours to suit the needs of users better.

At the same time, the physical location is not always the only meaningful spatial information. The physical location of a Web server of an airport authority can be extremely important for its administrator who desperately needs to perform a task that can be done only from the console. On the other hand, the *service area* of the

server should be the cities surrounding the airport, where most of the passengers come from or go to. That is, the information available from the Web server is intended for people within the service area, and although people from other areas may find this information useful, the server is most likely to serve people from within the service area.

These two different usages of spatial information suggest that the interpretation of spatial information should be flexible enough to cope with both cases. They also reflect two different views of the world: the client view and the server view. From the client view, the physical location of a client is usually more important, as the client is in general more concerned about objects within its vicinity than those far away. On the other hand, a server often has an aggregate view of the world, in which the majority of its clients should be served. Therefore, a server is concerned not necessarily about objects in its immediate vicinity, but about those in an independently-defined, service-specific geographic region, *i.e.*, its service area.

1.3.2 Data Model for Spatial Information

It is important to choose an appropriate data model for spatial information so that various geometric operations can be carried out. This includes calculating the actual travelling distance between two objects in the physical world. Since the architecture is to provide an infrastructure to support the development of applications with spatial components, this requirement becomes essential to ensure that various geometric calculations can be performed.

Another requirement on the data model is that a part of the location system must be understandable to human end-users. This comes from the fact that applications

developed on top of this infrastructure will most likely and most often be used by human users. Thus, it becomes important to present relevant spatial information in a form easily understood by humans.

Finally, the data model should allow application developers to represent spatial data to arbitrary precision using arbitrary geometric shapes. In other words, objects in the system should not always be modelled as points. Furthermore, the system should be able to provide users with a higher resolution whenever desired.

1.3.3 Distributed and Cooperative Nature

The system should be a distributed system with as little centralized control as possible so that it is more robust and does not have any single point of failure. The distributed nature of the system can also give the system some room for improving efficiency via concurrency.

Spatial and non-spatial data should be handled separately by different entities within the system. This helps to ensure orthogonality among various components within the architecture to keep everything simple. The separation of spatial and non-spatial data also raises the possibility of reusing some components for other purposes.

Management responsibilities should also be distributed among various entities, including system administrators and possibly end-users so that information flows into and out of the system easily. For instance, a service provider should be able to update its service entry in the system without much interaction with the system administrators. The cooperative nature of the architecture is important, as it has become increasingly clear that the management effort involved in any system is a determining factor of the overall cost.

1.4 Contribution

The thesis presents an architecture to support spatial information in the general environment of the Internet. This architecture allows geographic information to be integrated into the existing Internet. With this integration, application programmers are provided with a general architecture to handle spatial data easily and efficiently.

1.5 Thesis Organization

The remainder of the thesis is divided into five chapters, as follows. Chapter 2 presents some related work, similar to but different from the work done in this thesis. Chapter 3 discusses various design decisions made, along with their justifications, showing how the architecture meets the design goals using various design decisions. Chapter 4 discusses some issues in implementing the architecture and the choices made. Chapter 5 uses three case studies to illustrate the usefulness of the system, with the descriptions of the implementation of prototype software used for the case studies. In particular, the case studies employ the proposed architecture to construct a geographically-oriented Web, to locate resources in the physical world, and to discover services on the Internet. Chapter 6 concludes the thesis with a summary and future work.

Chapter 2

Related Work

This chapter explores selected research efforts related to performing geographically-oriented service discovery on the wide-area Internet. Since many services available through the Internet also have their corresponding presence in the physical world in the form of server locations or service areas, it is equally important to locate the services on the Internet as well as in the physical world. Thus, services can be discovered via the topology of the logical network, via the geography of the physical world, or a combination of both. In the remainder of this chapter, all three approaches are discussed in turn.

2.1 Service Location Protocol

The first approach is to perform service discovery via the topology of the underlying logical network, *i.e.*, services are discovered primarily using topological units such as subnets, networks, and domains, with location information as a secondary filter for a service with desired properties. This approach has been used widely in various

location-based service discovery protocols, such as the Jini™ Network Technology [20] and the Service Location Protocol (SLP) [13, 21]. Since these protocols share similar techniques, only the Service Location Protocol is covered here, as the protocol has been standardized by the Internet Engineering Task Force (IETF).

In the Service Location Protocol, a service provider uses a *Service Agent* to advertise on a service-specific multicast address, and to register with suitable service directories, called *Directory Agents*. Similarly, a service requester employs a *User Agent* to listen on the service-specific multicast address, and to query a Directory Agent in order to obtain information about a desired service. Therefore, for a small network, Service Agents and User Agents can communicate directly without Directory Agents. As service advertisements become excessive due to a larger network size and a higher number of services, a central Directory Agent should be used to reduce the network traffic. As the network size increases further, it is likely that a central Directory Agent will become unwieldy, and multiple Directory Agents with *scopes* will be used to accommodate the increased demand. A scope can be interpreted as an area or entity to which services are provided, indicating that a scope can be virtually anything chosen arbitrarily by the local network authority. A department of a company, a building on a university campus, and a hallway are examples of scopes. Directory Agents with scopes store service information according to the scopes so that the data and workload of each Directory Agent can be distributed satisfactorily.

The Service Location Protocol highlights two types of technique, commonly used in service discovery: advertisements and queries. Although both techniques have been used widely for service discovery on local networks and local domains, they both face problems as they do not work very well for remote networks and remote domains.

The advertisement approach leverages the power of multicast or broadcast in service discovery. However, broadcast is essentially unusable in a wide-area context like the Internet, simply because of the excessive amount of network traffic that it would generate. While multicast is available on the Internet, its coverage is extremely limited compared to that of unicast. Furthermore, a service requester can potentially face a response implosion problem: As the service requester usually has little or no control over the number of responses whenever multicast or broadcast is used, the requester can easily be overwhelmed by the responses from a huge number of service providers within a very brief period of time. Thus, the advertisement approach is inappropriate due to its network inefficiency.

The query approach requires the establishment of well-known service directories, with which service providers register and to which service requesters send their queries. The Service Location Protocol can easily be set up for service discovery within a single network domain, since there is a single network authority for that domain. However, service discovery within a single domain may turn out to be insufficient. As pointed out in Section 1.2, there is no relationship at all between the geographic proximity of two IP address owners and their IP topological proximity. In other words, an IP address owner may be physically next to a printer that is topologically in a remote IP domain. Thus, it is insufficient to perform service discovery only within a local domain, even if service providers and service requesters are physically close. Unfortunately, it requires administrative co-operation between at least two network authorities in order to make services from one domain available and accessible to another domain. As trivial as it may appear, this approach conflicts with the distributed nature of the Internet management, and requires a domain administra-

tive authority to predict in advance what arrangements need to be made with which domains, based on the needs of the end-users. Thus, the time and effort involved in the administrative process of human interactions are probably unjustifiable. Furthermore, if such a scheme is applied to the entire Internet, the inter-dependency created among various network domains will eventually become so strong that in the end, any network authority will find it impossible to implement any network policy without generating waves of policy changes across many other domains on the Internet. Thus, the query approach is inappropriate due to its management inefficiency.

It can be concluded that service discovery protocols are neither appropriate nor designed to scale to the size of the Internet. The fact that network topology acts as the primary axis of service discovery renders service discovery across multiple network domains an extremely complex and difficult task, preventing an efficient framework from being set up. This leads to the development of the second approach.

2.2 Location-Based Service Architecture

The second approach is exactly the opposite of the first, in that the importance of geography surpasses that of topology. That is, spatial requirements are first used to determine which services have the desired spatial properties, and then non-spatial requirements are used to check on other attributes. Under this approach, services are discovered spatially first, and it allows a rather straightforward solution to be constructed for service discovery on the wide-area Internet, as can be seen in the technology-independent architecture for supporting generic location-based services proposed by José and Davies [16].

In the architecture, each service is defined to have a service scope to indicate the

geographic area this particular service is expected to serve. For instance, the service providing information on a parking lot expected to serve people in a town would have a service scope covering the town. Every service scope is mapped into one or more location contexts, each of which is a symbolic location representing a spatial extent in which a device, an application, or a user is potentially interested. Location contexts and their relationships are stored at Location-Based Service (LBS) servers, and LBS servers are hierarchically structured among themselves by the containment relationship. Due to the fact that multiple location contexts may overlap, they form a graph as opposed to a tree. A service provider registers its services with one or more LBS servers in all location contexts mapped from its service scope. Therefore, once a device informs an LBS of which location context is of interest, services available in that particular location context can be obtained.

In this architecture, cross-domain co-operation is avoided by setting up an entity independent of network domains for handling service registrations and queries. Since service directories are reached using spatial requirements, the dependency on network topology is eliminated. This approach is therefore fundamentally advantageous compared to the network topology approach.

Another significant contribution of the architecture is its attempt to define a generic concept of location-based service in terms of logical location as opposed to physical location. The architecture stresses the importance of spatial information other than physical location, and attempts to address it by defining location contexts. It also fits nicely into the popular targeted-marketing strategy, in which a service provider has the opportunity to specify a service scope to advertise its services. The architecture also attempts to address the problem of proximity, *i.e.*, the physical

area in which an end-user has some interest. The authors recognize that this area is not always the immediate surrounding area of a user, and as such, the geometry of proximity is significantly different from scenario to scenario. Thus, it is neither sufficient nor desirable to support proximity using a single type of geometry, which suggests the need for multiple views of the space in order to support a wide range of applications. However, there are also a few problems with this architecture.

First, it is not clear how the architecture can be used by a wide range of applications without creating an extremely high number of location contexts. Therefore, scalability can become a problem as the number of location contexts and their relationships grow. Even if the location contexts can be distributed properly in the hierarchy of LBS servers, it cannot guarantee that each desirable location context has a match, as all location contexts are defined by LBS servers in advance, not by parties requesting the contexts at run-time. Therefore, this approach inevitably requires the administrators of LBS servers to foresee precisely what location contexts will be requested later in a wide range of application scenarios. In reality, such a requirement is difficult to satisfy.

Second, a technology-independent location-sensing technique is also proposed as a part of the architecture, and it is not clear why a location sensing mechanism should exist in the architecture at all. Assuming that more sophisticated and accurate location-sensing technologies will be developed in the future, and that networking technologies are not the best technology for location sensing, location-sensing mechanisms should be independent of such an architecture. All that is needed is a common coordinate system allowing different location-sensing devices to operate within it.

Third, the architecture uses the point of attachment of a device for location sens-

ing, and this may be inappropriate in many cases. In the wireless world, a link layer technology can usually be used to locate a device correctly. However, the accuracy of location sensing varies from technology to technology, and therefore may or may not satisfy the accuracy a device requires. In a wired network, the link layer may not be able to be used at all. For instance, it would be hard to determine where a laptop is attached to an Ethernet just from a link layer broadcast. Although an IP address prefix can be used to determine the context, there are at least two exceptional examples: Mobile IP and private IP addresses. In both cases, the point of attachment to an IP network cannot be used to locate the context at all, as the IP addresses are totally independent of the physical point of attachment to the network. In other words, devices having a Mobile IP address or a private IP address can be anywhere in the world. If an independent mechanism were used for location sensing, an extra mapping from the location sensed to the corresponding location contexts is required, which adds further complexity to the architecture.

Finally, the location contexts are organized into a general graph as opposed to a tree. This dictates that more complex software is required at the LBS servers to maintain the proper relationships between the location contexts. This prevents the overall infrastructure from being kept simple.

In summary, the architecture takes the correct approach of setting up an infrastructure to register services independent of any existing topological concepts such as network and domain. However, the use of scopes makes the scalability questionable, suggesting the use of absolute coordinates as opposed to symbolic locations. In the next section, an architecture attempting to leverage the combined power of topology and geography in an absolute coordinate space is discussed.

2.3 DataSpace

The third and final approach is to use a hybrid of the first two approaches, namely, discovering services by traversing the network topology and geography at the same time. This approach can be seen in DataSpace [12, 14, 15], in which a three-dimensional shell around the earth is modelled as a space containing a large number of *self-sufficient* objects. That is, processing power, memory, and network connectivity are assumed intrinsic to all objects, and each object is responsible for storing its own information locally. This property highlights an approach dramatically different from the traditional database approach, in which information is almost always stored at an entity different from the owner of the information. Thus, in DataSpace, information becomes an attribute of an object just like colour, shape, and mass.

Objects can be queried and monitored once they become visible to DataSpace. A view of DataSpace is modelled as a *datacube*, in which a directory service provides references to objects in the view. Objects need to register with datacubes in order to be seen in the views defined by the datacubes. Each object can register with multiple datacubes enclosing it, and therefore can be visible in multiple views.

Two approaches are proposed to implement this model: the multicast approach and the GeoCast approach. The former uses the multicast mechanism of the Internet Protocol, Version 6 (IPv6), whereas the latter employs a geographic-based routing protocol, GeoCast [19]. GeoCast is not widely accepted because an efficient implementation requires the replacement of all routers in the existing Internet, a risky step few are willing to take, as mentioned in Section 1.2. Thus, only the multicast approach is discussed here.

In the multicast approach, two multicast addresses are formed from the geometry

of a datacube, an indexed attribute of a class of objects, and a value range of interest. In other words, there are two multicast groups for each class of object with the indexed attribute falling in each value range within each datacube. One group is passive and the other is active. Members of a passive group are presumably objects that satisfy the spatial and non-spatial requirements encoded in the group multicast address. Therefore, these objects can be reached by sending a multicast message to the group address. On the other hand, members of the corresponding active group will receive a message whenever the membership of the passive group changes. Therefore, this is the mechanism to achieve monitoring.

Under the multicast scheme, the three-dimensional shell from 100 kilometers above the surface of the earth to 10 kilometers below the surface of the earth can be encoded with an accuracy of one cubic metre using all 112 bits allocated for multicast in IPv6 [14]. Thus, the network becomes a search engine that performs both spatial and non-spatial lookups with the application layer responsible for filtering objects that do not meet the remaining non-indexed attribute requirements.

The proposed model has some stringent requirements. First, it is hard for every physical object that wishes to reside in DataSpace to meet the processing power, memory, and network connectivity requirements. Second, the model relies on an object to register with the correct datacubes depending on its current location, and has no way of verifying the validity of the registration independently. This may cause inaccurate or incorrect query results to be produced and may become a security concern. Finally, the shape of each view is limited to a cube, which will most likely reduce the usability of the model dramatically in reality.

The geometry of objects in DataSpace is modelled in an overly simplistic way. As

mentioned in Section 1.3, the physical location of an object is not always the most meaningful piece of information. However, only physical locations can be modelled using DataSpace. Furthermore, it is also insufficient to model all objects as points, eliminating the possibility of an object having more complex geometric properties such as an area. These geometric limitations reduce the usability of DataSpace.

The proposed multicast approach also faces a few practical problems. First, the global coverage of IPv6 and multicast is extremely hard to achieve. This is particularly a problem for multicast, as it has never achieved the type of global coverage unicast enjoys on the current Internet. Second, it is very unlikely that the entire 112-bit IPv6 multicast address space can be dedicated to any one system, ruling out the possible use of multicast by any other system. Third, given the number of possible datacubes, classes of objects, and possible value ranges, a huge number of multicast groups is likely required, which raises a concern of scalability. Furthermore, a query may generate a large number of responses from objects satisfying the query, and the system provides no means of controlling the rate of response. Finally, some technical details are not clear in the specification. For instance, it is not clear how an object can obtain the geometry of a datacube, or how to decide a value range of interest — a piece of information that should mostly logically be supplied by an end-user.

Overall, the hybrid approach does not offer DataSpace any advantage over the geography approach, as the architecture is tied to the drawbacks of both approaches. On the one hand, this approach cannot be made independent of network topology, because the queries are carried out in the process of routing. On the other hand, routing also attempts to reach the object at the desired location. Since the Internet topology and the world geography are two totally incompatible concepts, finding an

elegant and simple solution becomes a difficult task.

2.4 Unexplored Opportunities

Previous research suggests that efficient location-dependent application development requires an infrastructure supporting spatial information. Although various applications have been developed, it is not clear what level of abstraction is provided from the underlying infrastructure. The complexity of such development efforts makes it impossible to design a monolithic infrastructure. Therefore, it is important to recognize and justify various pieces of the infrastructure and develop them separately and accordingly.

It is clear that in order to develop a simple and elegant solution, spatial requirements must be used as the primary search criteria augmented with non-spatial requirements. Otherwise, any proposed architecture is bound to run into efficiency, scalability, and management problems. Although a reasonable architecture has been outlined in Section 2.2, there is room for improvement, which becomes an unexplored opportunity for this thesis.

Chapter 3

Architecture

The architecture proposed in this chapter is for a distributed client-server query system capable of answering queries with both spatial and non-spatial constraints. In an attempt to maintain orthogonal functionality of each component in the architecture, spatial and non-spatial data are kept and processed separately by different entities. Assuming that each of these entities is stationed at an arbitrary network location, this design necessitates a minimum of two network transactions in order to answer a query with both spatial and non-spatial constraints. On the other hand, the same design offers overall simplicity of the architecture, and the spatial data stored in the system can be shared easily among applications using different non-spatial data, with little interference with one another. Furthermore, the latency introduced by an increased number of network transactions can potentially be reduced by the use of concurrency.

In this chapter, an overview of the architecture is given, followed by a description of each component and its responsibilities, design decisions made, and their justifications.

3.1 Overview

There are some desirable properties the architecture should have. These properties are not only crucial for keeping the architecture simple, flexible, and therefore useful for a wide range of applications, but also the determining factors for the success of the architecture.

The first property is the separation of spatial and non-spatial data. In fact, the choice between the separation and combination of the data amounts to a trade-off between the simplicity and openness of the architecture and the communication delay of information retrieval. That is, if the data are kept at one location, the communication delay will be low, because each retrieval of the data will only require passing a single message. In contrast, additional communication delay will be introduced due to at least one additional message, if the data are not co-located, and if the retrievals cannot be carried out in a true parallel fashion. However, there are more reasons to keep the data separate. First, the separation reduces the complexity of the components of the architecture by delegating a single responsibility to each of them. This reduces the complexity of the software, and allows different considerations to be given to different components. For instance, specialized hardware can be installed to speed up expensive geometric operations performed on spatial data, while regular hardware can be used for processing non-spatial data. Second, the separation ensures the openness of architecture, so that the data can be readily reused by other applications and systems at no additional cost in terms complexity. For instance, the computer system of a car can help the driver navigate through a network of labyrinth-like streets by retrieving the spatial data without the corresponding non-spatial data. Assuming the network bandwidth modern network technologies offer is sufficient for

most applications and systems, the benefit of data separation outweighs the drawback of communication delay. In the architecture, this separation is achieved by storing spatial data at *geometry managers* and non-spatial data at *kiosks*.

The second property is the distributed and cooperative nature of the system, as pointed out in Section 1.3. That is, the data should be distributed among multiple components as opposed to being stored centrally. Although centralized data storage offers lesser delay for a small amount of data, the scalability will eventually become an issue for a large volume of data. Furthermore, the central storage becomes a single point of failure of the architecture. A distributed approach also allows the control and management responsibilities to be distributed among various administrative authorities, and the history of the Internet shows that this is the management approach most economic and beneficial to the public good. The decentralized architecture is realized by structuring all geometry managers and kiosks into a tree with the geometry managers being the internal nodes and the kiosks being the leaves. Therefore, both spatial and non-spatial data are distributed to avoid overloading any particular node, giving a scalable architecture with no single point of failure.

The third property is the necessity to create a clean and simple access point to the system, so that application developers are not burdened with an excessive amount of redundant work to access the system. At the same time, other means of accessing the system should also exist, allowing the possibility of tailoring the data access behaviour based on the requirements of the application, in case the default behaviour is not satisfactory. While the separation of spatial and non-spatial data has already established multiple means of accessing the system, an illusion of a single server still remains to be created. Similar to the way a DNS server hides its consultations

with other DNS servers in the process of a lookup, *resolvers* are employed to query geometry managers and kiosks on behalf of an application to create the illusion.

With these properties, the architecture has been created with three distinct components in it: resolvers, geometry managers, and kiosks, and Figure 3.1 depicts their interactions with an application and among themselves. Each arrow represents a

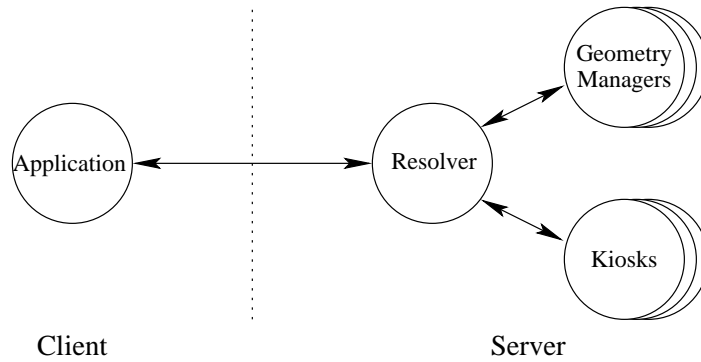


Figure 3.1: Entities in the Architecture

bi-directional communication link on which information flows back and forth between the two entities at the ends. For instance, an application interacts directly with a resolver, but not with a geometry manager or a kiosk. While the only client-side entity is the application, the resolvers, geometry managers, and kiosks make up a virtual server functioning as the server-side entity. Although all components are involved in answering queries, each component plays a distinct but crucial role in the architecture.

The application constructs queries with both spatial and non-spatial constraints based on application-specific semantics, and it is solely the responsibility of the application to construct the queries and interpret the results according to the semantics. For instance, one application might be interested in locating all printers in the rooms

down a hallway, and another application might want to find out about the daily special of the restaurants in a downtown area, but as far as the system is concerned, each application is a query issuer. The spatial constraint is used to state the area of interest of a particular service requester, within which the service in question is expected to be delivered from a service provider to the requester. The non-spatial constraint specifies the requirements on the attributes of the service other than the spatial constraint. The combination of spatial and non-spatial constraints ensures the selection of a service provider capable of delivering a service with desired properties at a desired location.

The resolver is the entry point into the system, via which an application gains access to the spatial and non-spatial information stored in it. A resolver receives a query from an application, separates the spatial and non-spatial constraints in the query, forwards the constraints to geometry managers and kiosks, waits for the results, and forwards the aggregated results back to the application. In other words, a resolver sits between an application and the rest of the system, performing spatial and non-spatial queries on behalf of the application. It is also worth noting that the resolver interprets neither the queries, nor the results. Instead, a resolver simply serves to decouple the two types of constraints, and to combine the two types of results. This process separates spatial information from non-spatial information.

The geometry managers store spatial information and answer spatial queries, but have no knowledge about non-spatial aspects of the environment. More specifically, each geometry manager keeps the spatial information about an area in which a service requester can potentially receive useful services. Each service provider registers services with appropriate geometry managers, depending on service areas. Upon re-

ceiving a spatial query, a geometry manager compares the service area of every service with the area of interest specified in the query, generates a list of geometry managers and kiosks relevant to the area of interest, and sends the list back as the query result.

The kiosks are in charge of storing and managing non-spatial aspects of the services available. That is, a kiosk does not know where the services are, but what they are, and therefore a service requester can determine what services are available by querying a kiosk. The name “kiosk” is chosen for its similarity to a real-life kiosk, providing information, services, and merchandise. The distinct characteristics of kiosks are that they are small in size, large in number, and distributed in an *ad hoc* fashion over a large area. Furthermore, these kiosks are independent and unstructured, in that they do not interact among themselves, and that there is no relationship of any kind among them.

The responsibility of each component is kept orthogonal to those of the others, as can be seen from the separation of spatial and non-spatial data between geometry managers and kiosks, as well as from the coupling and decoupling roles of resolvers. It is this orthogonality that keeps the overall architecture general and simple, and creates multiple views of the system.

- From the *application view*, the architecture is simply a knowledgeable server capable of answering queries with both spatial and non-spatial constraints. This view comes from the fact that the resolvers are hiding the distributed nature of the architecture, and therefore an illusion of a single server is created. This view is appreciated by applications caring not about how the queries are processed, but about the fact that the queries are answered through a simple interface.
- From the *resolver view*, the architecture consists of a large but unknown number

of related geometry managers and kiosks, and each of them has a piece of information about a part of the world. The geometry managers describe the spatial aspect of the world, whereas the kiosks describe the non-spatial aspect. This view is appreciated by applications caring to tailor the query behaviours to optimize the performance using application-specific semantics.

- From the *geometry manager view*, the system is a tree-structured space with each geometry manager knowing its parent and children. Furthermore, each geometry manager is responsible for interpreting its own service area, and may choose to delegate a part of its responsibility to its children. This view can be used by other systems wishing to retrieve the spatial data stored at the geometry managers.
- From the *kiosk view*, the architecture is unstructured. This view displays the distributed *ad hoc* nature of the architecture, and is appreciated by those who wish to contribute to the architecture with their own kiosks.

These views shows not only the variety of means of accessing the system, but also the flexibility and extensibility of the architecture.

In the remainder of this chapter, kiosks, geometry mangers, and resolvers are discussed in turn.

3.2 Kiosks

Kiosks describe non-spatial aspects of the objects in the world, and their only role is storing, processing, and retrieving non-spatial information about these objects. That is, a kiosk is concerned not with *where* the objects are, but with *what* they are. More

specifically, each kiosk contains information about *services* accessible to a particular environment, and this information comes from service providers who wish to make some services available to people within the environment. It is the responsibility of each provider to determine the valid and desired spatial destination, to which the service in question can be delivered to a service requester without the movement of the requester. While service providers provide service information and store it at kiosks of choice, service requesters query kiosks for this information to access services. Ideally, both service requesters and service providers should be authenticated, and granted read and write access accordingly, so that they could interact only with the kiosk software, not with the kiosk administrators. This is a rather efficient approach due to its lack of human interactions on the server side. Finally, kiosks should be large in number, but small in size in order to meet the design goal of having a distributed nature.

Based on the requirements on kiosks, normal SQL databases are chosen to serve as kiosks for their well-known ability to handle non-spatial data. In addition, existing database management utilities can be used to manage kiosks. For instance, service providers can be authenticated and granted the privilege to update their data. Since the kiosks handle no spatial data, no special software is required, and virtually any SQL database can be used, which opens the opportunity for anyone having an SQL database to contribute to the system. This is similar to the way anyone can contribute to the World-Wide Web.

Although the kiosks can be owned and operated by different authorities, all of them must share a single schema, so that service providers and requesters can store and retrieve the data seamlessly across multiple kiosks.

3.2.1 Schema

Each kiosk is a table called **Kiosk** in any arbitrary database, and all **Kiosk** tables share the same schema of three columns: **ID**, **Keywords**, and **URI**. With the **ID** field being the primary key, every row of the **Kiosk** table represents a service whose attributes are described by **Keywords** and whose location on the Internet is identified by **URI**. Each row is considered a distinct service, and therefore no relationship is assumed between any two rows of the table. Thus, even if two services are offered by the same service provider, no obvious relationship between them can be observed from the schema. Moreover, even if the **Keywords** and **URI** of one row are identical to those of another, the two rows will still be considered distinct due to the difference in the primary keys.

This schema is kept simple intentionally to avoid imposing unwanted semantics on the kiosk and therefore reducing the generality of the overall architecture. **Keywords** meets the minimal searchability requirement, whereas **URI** is a necessary reference to an object on the Internet. If the **Keywords** were missing, there would be no way of finding objects satisfying certain attributes, assuming searching within the **URI** is inappropriate and insufficient. The **URI** cannot be eliminated either because it allows a service provider to be pinpointed on the Internet. Therefore, the schema is absolutely the simplest for locating a service with some properties on the Internet, and it is up to the applications to attach semantics to the data in it according to their unique needs.

As can be seen in the schema, no spatial data are stored at the kiosks. However, each kiosk does possess some spatial properties.

3.2.2 Spatial Properties

The service area of each kiosk specifies the spatial extent of the kiosk. A service area is a closed polygonal shape used to limit the spatial extent of the non-spatial data stored at a kiosk. For instance, if a kiosk is set up for the services accessible to an office, the service area of this kiosk is the office. Although each kiosk has its service area, the geometric definition of the area is not stored at the kiosk due to the spatial nature of the definition. As a result, this piece of information is stored at the parent geometry manager of a kiosk, the handler of spatial data.

This configuration implies a kiosk is only concerned with non-spatial aspects of the objects within its service area. A kiosk cares neither about objects outside its service area, nor about the spatial aspects of these objects. In fact, a kiosk does not even know its own service area. Therefore, if a kiosk has a piece of information stating that a postal service is available, it means that the service is accessible within the service area of the kiosk. As for where exactly the service area is, it is up to the parent geometry managers to decide. Thus, the kiosks are independent of one another, and they are not aware of the existence of geometry managers, resolvers, or any other kiosks.

Each kiosk is also associated with a *scale* at which the non-spatial data it contains applies. A scale is a numerical value indicating the level of detail, and is used in conjunction with a service area to specify services common to some geographic area. A world map, the culture of a province, and the history of a city are examples of such data. Although this type of data could be replicated at various kiosks, updating would potentially involve a large number of kiosks and therefore be an expensive operation. If the area in question is sufficiently large, this type of non-spatial data

may pose a scalability problem to the system. For instance, if a world map were stored this way, a change in the world map would require an update at each kiosk in the system. Thus, each query to a kiosk is associated with a scale range in which the kiosk lies to ensure that only spatial data of the desired levels of detail are included. Similar to the service area, the scale of a kiosk is a spatial property, and is therefore stored at the parent geometry manager.

Kiosks handle only non-spatial data, and are unstructured by design. It is the geometry managers that bring a structure into the architecture.

3.3 Geometry Managers

Geometry managers are responsible for storing spatial information and answering spatial queries. Similar to the way an Internet Service Provider (ISP) provides DNS services to its customers, the administrative authority uses geometry managers to provide the road map for how to locate services in the environment. A service provider can then decide where on the map the service should be provided, and a service requester can similarly decide where on the map the service should be received. Whenever the service area of a provider overlaps the area of interest of a service requester within the desired scale range, that service provider becomes a candidate provider for the requested service. The first issue for the geometry managers is a properly chosen coordinate system.

3.3.1 Coordinate System

As pointed out in Section 1.3, the coordinate system should be absolute so that the spatial information can be used directly in geometric calculations. Thus, spatial data stored in the system is represented using an absolute geographic coordinate system, in which the global location of an object can be described without ambiguity. For example, latitude and longitude can be used as an absolute coordinate system, and the location of a bus stop can be expressed as a point on a 2-dimensional Cartesian plane. However, absolute coordinates are not sufficient, since human interactions are most likely involved in the process of service discovery. Thus, various locations should also be expressed in a human-readable form. Finally, more complex geometric shapes, *e.g.*, polygons, should be available to meet various needs of applications.

In an attempt to meet these design goals, an absolute coordinate system augmented with symbolic names is used. Service providers are allowed to use polygons to express their service areas, and service requesters also use polygons to express their areas of interest. In both cases, the polygons are described using the absolute coordinates of the vertices. For instance, a rectangular building can be modelled as a rectangle, expressed in terms of the coordinates of its four corners. Symbolic names come into play when service providers desire to attach some semantics to these polygons, *e.g.*, this polygon represents an office building, whereas that rectangle represents an office. So, optionally, a name can be given to each geometric shape stored at a geometry manager to make it understandable and informative to a human user. A symbolic name is a particularly useful piece of information for a human user to locate the service provider physically.

Such a combined approach allows various geometric calculations to be carried

out using absolute coordinates, so that the system can provide services to a variety of applications. However, this comes at the cost of an increased computational complexity at the geometry managers, as it is always more expensive to perform multiple-dimensional geometric operations than the one-dimensional counterparts. For instance, the inclusion operation, *i.e.*, determining whether a point lies on a line segment, requires only comparisons in 1-dimensional space, but multiplications and divisions in 2-dimensional space. Furthermore, the inclusion operation itself becomes more complex, as it can mean determining whether a point is included in a polygon, a yet more complex operation. Although efficient geometric algorithms are beyond the scope of this thesis, it is believed that operations such as determining the intersection of two polygons can be performed reasonably efficiently. Furthermore, by specifying the service area and scale of a geometry manager, the geometry manager should only be responsible for a limited amount of computational complexity.

With the coordinate system chosen properly, a hierarchy of geometry managers and kiosks can be created.

3.3.2 Hierarchy of Geometry Managers and Kiosks

The geometry managers can be structured into a hierarchy or a flat space. While the flat approach diminishes the need to traverse a hierarchy, objects in such a space are regarded as having similar spatial magnitudes, and can be therefore viewed from a fixed scale. In other words, all objects are seen with the same level of detail, and the only way to select objects is a spatial contour. This may prove to be extremely limited with regard to the types of application the space can support. Thus, the hierarchical approach is chosen.

Geometry managers and kiosks are structured hierarchically into a tree with the geometry managers as internal nodes and the kiosks as leaves, as shown in Figure 3.2. In the diagram, there are five geometry managers g_1, g_2, \dots, g_5 and nine

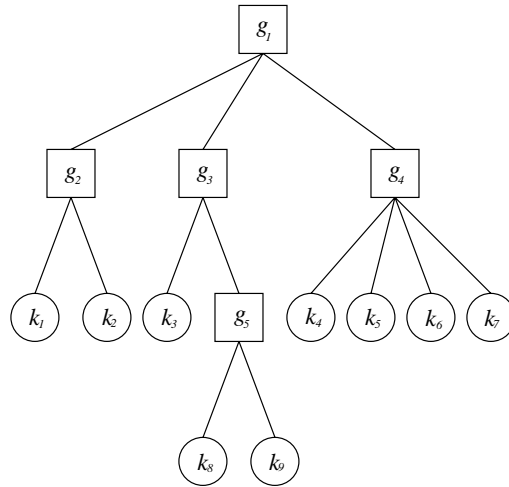


Figure 3.2: Geometry Managers and Kiosks

kiosks k_1, k_2, \dots, k_9 . A tree is chosen over a general graph for its simplicity, and it fits naturally into the spatial information of various scales stored within the structure. Each geometry manager keeps references to its parent (if any) and children. These references are shown as edges between nodes in Figure 3.2. For any geometry manager, all points inside the service area of a child node are also inside its own service area. For example, all points inside the service areas of k_3 and g_5 are also inside the service area of g_3 . Similarly, each geometry manager may have a number of children and any point inside the service area of a child is always inside the service area of the parent.

A geometry manager maps an area of interest into a list of geometry managers and kiosks capable of providing further information about the area. Essentially, the geometry manager compares the area of interest with its service area, and calculates

which of the adjacent nodes are responsible for processing the area of interest further. Such processing can be either spatial (in the case of a child geometry manager) or non-spatial (in the case of a child kiosk). Thus, a geometry manager does not necessarily have all geometric knowledge about its service area, and can delegate some portions of that responsibility to its children. This prevents the system from overloading any one geometry manager with excessive amounts of geometric data and processing. Therefore, a child geometry manager can be viewed as a more detailed view of a portion of the service area, whereas a child kiosk can be viewed as the non-spatial view of an indivisible portion of the service area. In replying to a query, a geometry manager is answering the question: “Which geometry managers and kiosks should be queried in order to process the area of interest?” An entity is said to be relevant if the service area of the entity intersects the area of interest. Therefore, a geometry manager can apply its spatial knowledge in determining which entities should be included and excluded in processing the query, depending on the area of interest. In fact, the only responsibility of geometry managers is to ultimately map an area of interest into a set of kiosks so that the attribute data of interest can be retrieved from these kiosks.

3.3.3 Approximations to Areas

Each area of interest is ultimately mapped into a list of kiosks, from which the non-spatial data of interest can eventually be retrieved. This process introduces some inaccuracy, as shown in Figure 3.3. The top-left diagram shows an area partitioned by nine kiosks and the top-right diagram is a hypothetical area of interest. Assuming that all nine kiosks share the same geometry manager as the parent, then this

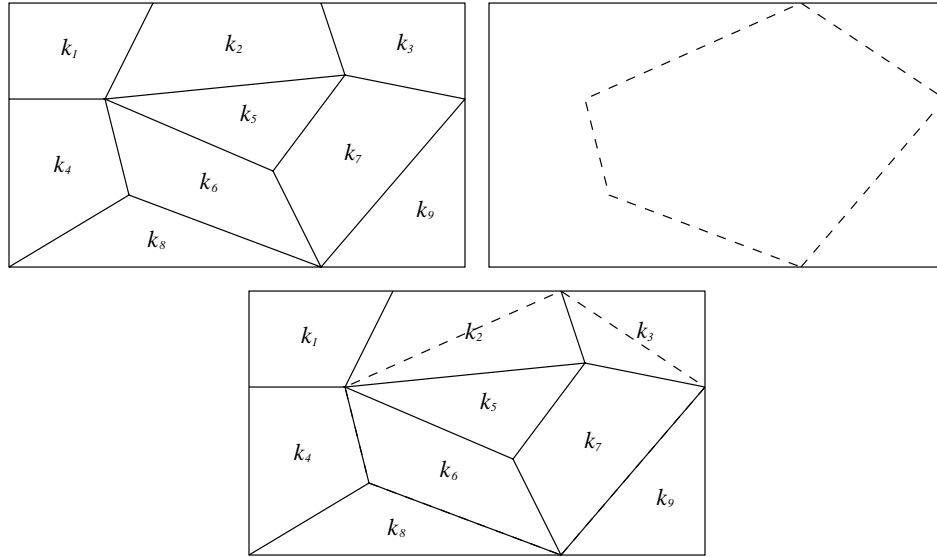


Figure 3.3: Approximation of an Area of Interest

geometry manager will map the query area into kiosks k_2 , k_3 , k_5 , k_6 , and k_7 . This result is inaccurate because clearly only parts of k_2 and k_3 should be included. Consequently, the query results will include extraneous entries outside the area of interest. Obviously, the query results will be an approximation to the original result with the property that the approximation is always a superset of the original.

There are two justifications for this inaccuracy. First, the size of the service area of each kiosk is arbitrary and adjustable. Therefore, it is possible to make each service area sufficiently small so that the corresponding accuracy is acceptable for most users of the system. For example, if each kiosk in Figure 3.3 represents a street block, then the overall accuracy is probably acceptable. A better approximation can be achieved by using small service areas, if desired. Second, even if spatially accurate query results can be obtained somehow, their utility is questionable. For instance, imagine travelling in a strange city and wishing to find a restaurant for supper. Suppose one wishes to choose from all restaurants within 500 metres. However, the 500-metre

requirement is not meant to be exact. That is, what you really want is all restaurants within *about* 500 metres, and if a restaurant 501 metres away offers an excellent entrée for a reasonable price, you would not mind walking the extra metre to get there. Many queries in daily life possess this fuzzy nature. If the spatial constraints were processed exactly, the restaurant 501 metres away would be sadly excluded. In other words, an approximation with some extra information is not only acceptable, but also preferred in many ways.

Alternatively, if spatial precision is desired, spatial information must somehow be attached to each service stored at the kiosk. For instance, a fourth column containing spatial information might be added to the `Kiosk` table as a spatial correction for the service represented by each row. It is assumed that some degree of inaccuracy in the architecture is acceptable for most applications, leaving the understanding and achievement of spatial accuracy as possible future work, as outlined in Section 6.1.

3.4 Resolvers

In general, multiple kiosks and geometry managers are required in order for a query to be answered. Therefore, multiple network connections need to be used, assuming the geometry managers and kiosks do not co-locate. Hence, there is a need for an entity to communicate with the geometry managers and kiosks, and this task is performed by resolvers in the architecture.

There are at least two reasons for establishing resolvers. First, resolvers serve as gateways into the virtual server answering mixed spatial and non-spatial constraints, and can therefore be used to hide the distributed nature of the overall architecture from applications. This practice shields applications from unnecessary details of de-

signing and implementing algorithms to access the spatial and non-spatial data by providing an easy and clean communication interface. Thus, for most applications, how queries are performed within the architecture is usually less important than the fact that the queries are answered in a reasonably efficient fashion via a simple protocol. If the performance of this protocol is not satisfactory for some reason, it is also possible to bypass resolvers and communicate with geometry managers and kiosks directly. This offers a viable option for extending the architecture incrementally. Second, it is likely more desirable to perform the queries at the resolvers, as they may have more processing power, memory, network bandwidth, and other resources than regular client machines, especially in the case of mobile and wireless devices. Hence, it makes sense to free the client machines from some of the heavy duties.

Essentially, the resolver receives queries with mixed spatial and non-spatial constraints from an application, decouples the constraints, forwards them to geometry managers and kiosks, and returns the combined results to the application. The spatial constraint is recursively forwarded to all geometry managers relevant to the query, and consequently, the resolver eventually receives a complete list of the kiosks from which the spatial data describing the area of interest can be retrieved. The resolver then queries these kiosks with the non-spatial constraint to retrieve the non-spatial data. Finally, the combined result is forwarded back to the application. Thus, a resolver communicates with applications, geometry managers, and kiosks.

3.4.1 Resolver-Application Communication

A resolver listens on a well-known port for an incoming query, containing a string of keywords, the first row requested (inclusive), the last row requested (exclusive), the

minimum scale (inclusive), the maximum scale (inclusive), and an area of interest. While the first and last row are used to limit the number of results, the minimum and maximum scales are used to specify the levels of detail.

Each resolver is configured to have a default geometry manager to which all spatial constraints are forwarded. The administrative authority of a resolver can choose an arbitrary geometry manager to be the default, and can therefore take factors such as mobility into consideration. For instance, a resolver can be configured to attach to the geometry manager of a target department, so that queries involving the target department can be handled without too much traversal of the hierarchy. Conceptually, a resolver performs the following tasks upon receiving a query:

- Forward the minimum scale, the maximum scale, and the area of interest to the default geometry manager, and receive a list of nodes relevant to the query.
- For each node, do the following:
 - If this node is a geometry manager, forward the minimum scale, the maximum scale, and the area of interest to the geometry manager, and append any nodes received to the end of the list.
 - If this node is a kiosk, forward the string of keywords to the kiosk, and save any results returned.
 - If the number of results meets the requirement specified by the first and last rows requested, or if all nodes are visited, return the results.

Essentially, the algorithm recursively creates a spanning tree rooted at the default geometry manager, covering all geometry managers and kiosks relevant to a query.

While the geometry managers decide which geometry managers and kiosks are relevant to the query based on the spatial aspect, kiosks determine which services are desired based on the non-spatial aspect. It is also worth noting that neither the constraints nor the results are interpreted by the resolver, the mere middleman between the client- and server-side entities.

3.4.2 Resolver-Geometry Manager Communication

The resolver initiates a spatial query on behalf of an application by forwarding the minimum scale, the maximum scale, and the area of interest to the default geometry manager. Upon receiving this information, the geometry manager determines the nodes relevant to the query as follows:

- For each child node of the geometry manager, if there is an intersection of the service area of this node and the area of interest, append the URI, type, and description of the node to the result list.
- If the area of interest is not included within the service area of the geometry manager, append the URI, type, and description of the parent to the result list.

In essence, the algorithm returns a list of adjacent nodes in the hierarchy of geometry managers and kiosks, each of which is potentially capable of processing a part of the area of interest. As a result, the resolver receives a list of relevant nodes, with which the query can be further processed.

3.4.3 Resolver-Kiosk Communication

Once a resolver has sufficient information about a kiosk, it forwards a non-spatial query to the kiosk. Since each kiosk is a normal SQL database, any existing database connectivity technology can be used to perform this task. Therefore, the expressive power of relational algebra determines the types of queries that can be constructed. For instance, predicates can be constructed using $=$, \neq , $<$, \leq , $>$, and \geq , augmented with logical operators \vee and \wedge . Therefore, assuming the `Keywords` attribute is a string of keywords describing services, SQL is capable of handling most queries, and refining this facility is a subject for future work.

Chapter 4

Implementation

Similar to the issues involved in designing the architecture, many decisions are made in the process of implementing it. Resolvers and geometry managers are implemented in Java (Standard Edition, version 1.3.0), and kiosks are simply MySQL databases (version 3.23.22-beta). All entities in the architecture execute on a server running Red Hat Linux (version 7.0). Java is an excellent choice as a prototyping language for its simplified programming interfaces and powerful built-in library, whereas MySQL and Linux are chosen because they are easy to configure and readily available free of charge. No modification has been made to the database engine, which illustrates the ease of turning an existing SQL database engine into a kiosk without disrupting its current functions. This means that no dedicated hardware or software is necessary, and a kiosk can be set up simply by creating a database on an engine willing to spare a small amount of its resources. However, two major implementation issues remain to be addressed:

- How is spatial data encoded at geometry managers?

- What protocols and message formats are used in communication?

In the remaining of this chapter, these issues are discussed.

4.1 Encoding of Spatial Data

Spatial data are encoded using the Drawing Web Format (DWF), a 2-dimensional vector-plot file format developed by Autodesk, Inc., and this format is chosen for a number of reasons.

First, DWF is a vector-plot file format, which allows operations such as zoom to be carried out with little loss of precision. This is a tremendous advantage over the raster counterpart. As a result, the relationships between various geographic entities are maintained correctly in the DWF files.

Second, DWF files retain more information than conventional plot files. In the process of creating the conventional plot files, the invisible information contained in the original drawing files is usually purged, because it does not produce any visual effects. The invisible information may include layers, symbols, names, and URIs. Therefore, plot files by nature contain much less information than original drawings. DWF files are quite different from conventional plot files, as they in fact save a portion of the hidden information such as layers and URIs.

Third, the DWF file format is compact. This feature stems from three factors. The first and crucial factor is that much of the unwanted information from original drawing files is dropped due to the plot-file nature of DWF. The second factor is the use of relative coordinates. The coordinates in a DWF file are 32 bits in size. However, by using a combination of absolute and relative coordinates encoded in two or four

bytes, DWF files offer a 32-bit coordinate space using virtually the amount of space taken up by a 16-bit coordinate space. The third factor is the use of compression. The DWF file format supports the ZLIB compression scheme using a efficient dictionary highly customized for DWF files.

Fourth, the DWF format is used in AutoCAD products. Many maps, plans, and drawings come in AutoCAD formats, making DWF an ideal format, because file conversions can largely be avoided. This can effectively eliminate unwanted loss of data and precision in the process of conversion.

DWF can express crucial aspects of spatial data, including layers, URIs, and transformation matrices between application coordinates and DWF coordinates. Layers are essential for their well-known role of grouping geometric objects into meaningful units, which can be used to increase the processing efficiency by limiting the number of geometric objects involved. URIs are also important for their unique ability to refer to objects on the Internet. However, the most vital piece of information surviving in DWF is the transformation matrix between application coordinates and DWF coordinates.

Application coordinates correspond to the coordinate space in which the original drawing is created. This application-specific coordinate space is in general uniquely defined in each drawing file. On the other hand, a discrete 32-bit coordinate space is used as the DWF coordinate space. The mapping between the two coordinate spaces is achieved by defining a 4×4 transformation matrix of the form:

$$X = \begin{bmatrix} A & \mathbf{p} \\ \mathbf{q}^T & r \end{bmatrix}$$

where A is a 3×3 matrix that produces a linear transformation such as scaling, \mathbf{p} is a translation vector, \mathbf{q} is a vector associated with a perspective transformation, and r is a scalar that usually has the value of 1. Let \mathbf{u} and \mathbf{v} be the application and DWF homogeneous 3D coordinates of a point, respectively. If the above 4×4 matrix X is the transformation matrix, then the mapping between \mathbf{u} and \mathbf{v} can be expressed as:

$$X\mathbf{u} = \mathbf{v}$$

Thus, the valuable information about the application coordinate space is maintained within the DWF file.

The content of the simplest DWF file is shown in Figure 4.1. While this particular file is in plain-text, it is also possible to use a much more compact, compressed binary format. The first line indicates the file type is DWF with a version number of 00.42, followed by other miscellaneous information such as the author and creator program. The first useful piece of information is provided by the `Units` command, by which the transformation matrix is defined as four \mathbb{R}^4 column vectors. In the example, the transformation matrix is:

$$\begin{bmatrix} 1185.3835191 & 0 & 0 & 2147396623 \\ 0 & 1185.3835191 & 0 & 28106 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This matrix implies that the DWF coordinates are obtained by scaling the corresponding application coordinates by a factor 1185.3835191, plus some translation. In general, the transformation matrix takes the form:

```

(DWF V00.42)
(Author q2li)
(Creator 'Genuine AutoCAD 2000 (15.0) - Autodesk(R) Design Your World (tm)')
(Created 995124867 '07/14/2001 11:34:27' '{74555CA2-E171-4150-857D-F07CBB52B7F7}')
(Modified 995124867 '07/14/2001 11:34:27' '{74555CA2-E171-4150-857D-F07CBB52B7F7}')
(SourceFilename square.dwg)
(SourceCreated 995059726 '07/13/2001 17:28:46' '{FDEAD576-A652-11D2-9A35-0060089B3A3F}')
(SourceModified 995064834 '07/13/2001 18:53:54' '{6181E639-AB92-46E4-8BE0-CAE9873F52B4}+')
(Units '')
((1185.3835191 0 0 0)
(0 1185.3835191 0 0)
(0 0 1 0)
(2147396623 28106 0 1))
(Embed 'image/vnd.dwg;' 'Genuine AutoCAD 2000 (15.0) - Autodesk(R) Design Your World (tm)' 'square.dwg' '')
(NamedView 2147364978,0 2147483647,75336 INITIAL)
(View 2147364978,0 2147483647,75336)
(PlotInfo hide 90 in 10.99999976 8.4999997597 0.70054181917 0.22810038619 10.299458391 8.2718994862
((8.055731376e-005 0 0)
(0 8.055731376e-005 0)
(0 0 1)))
(Background 225)
C 10
(LineWeight 122)
(LineStyle
(AdaptPatterns true)
(LineJoin bevel))
(Layer 1 polygon)
(Viewport '')
(Contour 1 4 2147483647,0 2147483647,75337 2147364978,75337 2147364978,0)
(Units '')
((1185.3835191 0 0 0)
(0 1185.3835191 0 0)
(0 0 1 0)
(2147396623 28106 0 1)))
P 5 2147385954,38774 2147407291,38774 2147407291,17438 2147385954,17438
2147385954,38774
(EndOfDWF)

```

Figure 4.1: A Simple DWF File

$$\begin{bmatrix} a & 0 & 0 & b \\ 0 & c & 0 & d \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where a , b , c , and d are non-zero values and the application coordinates (x, y) can be calculated as:

$$\begin{aligned} x &= \frac{x' - b}{a} \\ y &= \frac{y' - d}{c} \end{aligned}$$

Notice that the same matrix is defined again by the `Viewport` command later in the file. There are two other pieces of useful information: the layer and the DWF coordinates forming the polygon. The layer information is defined using the `Layer` command and in the case of the example, a layer called “polygon” is defined. Finally, the polygon is defined using the `P` command with the DWF coordinates of four points. The application coordinates can be calculated as follows:

$$\begin{aligned} x &= \frac{2147385954 - 2147396623}{1185.3835191} \\ &\approx -9.000462574 \\ &\approx -9 \\ y &= \frac{38774 - 28106}{1185.3835191} \\ &\approx 8.999618966 \\ &\approx 9 \end{aligned}$$

Thus, the first vertex of the polygon is $(9, 9)$. Similar calculations can be carried out for other vertices, and the resulting polygon is a square as shown in Figure 4.2. If the

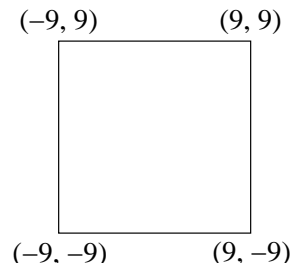


Figure 4.2: Square

ASCII representation is used as shown in Figure 4.1, the DWF file is 1,408 bytes in size. In contrast, the compressed binary representation of the same square is merely 604 bytes. Assuming latitude and longitude are used as the world coordinate system, each object is expressed as a polygon whose vertices are described with a latitude-longitude pair. Then, the application coordinates are translated into the correction DWF coordinates, described by the transformation matrix.

Although the DWF format appears to be an attractive choice for encoding spatial data, precision losses do occur in the process of plotting. A number of factors can potentially contribute to precision losses. For example, the complexity and capability of the plot device can lead to precision losses. The built-in plotter of AutoCAD 2000 automatically breaks more complex drawing objects (*e.g.*, a Bézier curve) into simpler objects (*e.g.*, a series of line segments), which inevitably introduces some degree of loss of precision. Another source is the resolution at which a DWF file is plotted. Finally, the discrepancy between the continuous coordinate space of applications and the discrete coordinate space of DWF files leads to inevitable precision losses. Despite these precision losses, the quality of properly plotted DWF files is in general sufficient.

4.2 Protocols and Message Formats

There are three communication links used in the architecture: resolver-application, resolver-geometry manager, and resolver-kiosk, and this section discusses the protocols and message formats being used on these links.

Over the resolver-geometry manager and resolver-kiosk links, the Hypertext Transfer Protocol (HTTP) is used due to the tremendous popularity and success the protocol has enjoyed on the Internet ever since the very birth of the World-Wide Web. In particular, the `POST` method of HTTP/1.0 [3] is used to upload query criteria as a well-defined Internet media type, `multipart/form-data` [18]. This media type is well-suited for the task, because query criteria consist of multiple types of data. In the case of the resolver-application link, the criteria are an SQL query, the first row requested (inclusive), the last row requested (exclusive), the minimum scale (inclusive), the maximum scale (inclusive), and the area of interest, as can be seen in Figure 4.3. The result of the request is a table of services satisfying the query criteria. In the case of the resolver-geometry manager link, only the last three elements are included, as geometry managers are only capable of handling spatial data. The result of a query to the geometry manager is a table of nodes with their node type, URI, and description.

Over the last link, JDBC is naturally used in conjunction with the Java programming language. JDBC is a data access API that provides the Java programming language with a universal way of accessing virtually any kind of tabular data, including traditional database tables, spreadsheets, and plain text files. Although it is often thought of as the acronym for “Java Database Connectivity,” JDBC is actually a trademark. JDBC allows its user to send SQL statements to a database easily. Thus, the SQL query shown in Figure 4.3 can be fed directly to each kiosk involved,

```

POST http://www.search.shoshin.uwaterloo.ca/ HTTP/1.0
Accept: */*
Accept-Language: en-ca
Content-Type: multipart/form-data; boundary=LGRKSMDVGIKULZYRLHMFSEBDPUHDMODISMPOK
Pragma: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
Host: www.search.shoshin.uwaterloo.ca
Content-Length: 1757
Proxy-Connection: Keep-Alive

--LGRKSMDVGIKULZYRLHMFSEBDPUHDMODISMPOK
Content-Disposition: form-data; name="query"

SELECT Keywords, URI FROM Kiosk WHERE Keywords LIKE '%chinese%' AND Keywords LIKE '%food%';
--LGRKSMDVGIKULZYRLHMFSEBDPUHDMODISMPOK
Content-Disposition: form-data; name="firstrow"

0
--LGRKSMDVGIKULZYRLHMFSEBDPUHDMODISMPOK
Content-Disposition: form-data; name="lastrow"

10
--LGRKSMDVGIKULZYRLHMFSEBDPUHDMODISMPOK
Content-Disposition: form-data; name="minscale"

2000
--LGRKSMDVGIKULZYRLHMFSEBDPUHDMODISMPOK
Content-Disposition: form-data; name="maxscale"

2000
--LGRKSMDVGIKULZYRLHMFSEBDPUHDMODISMPOK
Content-Disposition: form-data; name="aoi"

(DWF V00.42)...
--LGRKSMDVGIKULZYRLHMFSEBDPUHDMODISMPOK--

```

Figure 4.3: Query Criteria Uploaded as multipart/form-data

and as a result, the rows selected by the query are returned.

There are a number of reasons for choosing JDBC over other protocols such as ODBC. First, the fact that Java is used to code all parts of the system makes JDBC a natural choice for database access. Both the API and drivers are available from various sources including Sun Microsystems, Inc. Second, the JDBC API is a technology that intends to function as the base for developing other APIs and protocols. In this process, ODBC, among other things, is made one of the APIs available on top of JDBC so that ODBC calls are translated into JDBC calls. In other words, ODBC becomes a subprotocol of JDBC. Furthermore, existing ODBC drivers can be used to cope with the temporary unavailability of pure Java drivers for particular databases. Third, more general protocols such as XML require other components to be added into the system, which introduces unnecessary complexity. The trade-off between complexity and flexibility seems not to be justified.

4.3 Concurrency Control

Multiple threads are used in the implementation of resolvers to reduce the negative effect of communication links on the performance. When a query arrives at a resolver, a list of geometry managers and a list of kiosks are created, with the former containing only the default geometry manager and the latter being empty. A thread is spawned for each unvisited geometry manager or kiosk, and the resulting geometry managers and kiosks are appended at the ends of the appropriate lists. While this algorithm does not control the number of threads spawned, maximal concurrency is attempted. If desired, the maximum number of threads can be controlled by creating an upper limit on a pool of threads, where the maximum number can be determined by the

amount of resources available at a particular resolver.

One problem arises due the concurrent nature of resolvers. That is, repeating a query will yield the same set of results, but not necessarily in the same order as the previous query. This problem can be avoided by returning the two lists to an application as the state information about a query. Therefore, if an application wishes to maintain a consistent view between two queries, this state information should be kept at the application, and sent to the resolver along with the queries. In addition, this small piece of information allows the resolver to continue its work from where it was suspended, and thereby reduces the amount of unnecessary work at the resolver and the delay observed by the application.

Chapter 5

Case Studies

In an attempt to validate the architecture, three case studies have been developed. An application developer can employ the architecture, and attach desired semantics to it. This flexibility is demonstrated through the development of the three cases. In particular, the first case covers the topic of surfing the World-Wide Web using a geographically-oriented search engine. While the second case is about locating objects in the physical world, it is in fact a special case of the first case. Finally, the last case explores the possibility of service discovery on the Internet using geographic criteria as a generalization of the first two cases, leading to a more insightful view of the architecture. All three cases demonstrate the generality, flexibility, and power of the architecture, and emphasize the fact that the creation of an innovative application can be achieved through ingenious attaching of semantics to the architecture.

5.1 A Geographically-Oriented Web

The first and the most involved case is a geographically-oriented World-Wide Web search engine capable of indexing Web pages using spatial and non-spatial constraints. In this case, some semantics are given to the architecture, *i.e.*, each service is assumed to be a Web page providing information to a Web user. This assumption enables application developers to employ the architecture to tag spatial information to Web pages, making the pages geographically searchable with a minimal amount of effort. As a result, information can be retrieved from the Web using spatial constraints augmented with non-spatial constraints, and there are certainly many motivations for this approach to information retrieval on the Web.

Motivation

The World-Wide Web has been the most popular source of information, particularly among the younger generation, and yet the expressive power of Web query languages and the searchability along the spatial dimension are extremely limited in any existing search engines. Usually, a search query is initiated by a Web user specifying a few keywords augmented with logic operators at a Web-search engine, and the engine processes the query to yield a list of Web pages it considers relevant to the keywords.

This approach is dependent on the correct understanding of the semantics of keywords in natural languages. However, due to the inherent ambiguity of any natural language, the true meaning of a word is never entirely clear. For instance, the word “Waterloo” can mean a university (as in “the University of Waterloo”), a battle (as in “the Battle of Waterloo”), or a city (as in “Waterloo, Ontario”). Although additional words can be given to make a query more specific, there is no standard

way of extracting the semantics from phrases of natural languages. Furthermore, in the case of spatial information, it is not always possible to name every place on the planet using words, simply because many places are nameless. The limitation of the existing Web query language suggests the need for a language for expressing spatial information.

Knowing that existing search engines cannot decide the spatial extent of a query, the next problem many surfers face is that each query might essentially be answered for the entire spatial extent of the entire Web. That is, the query is conducted based on content, not location, and consequently, query results can easily become too general and too many to be useful if the location information is crucial to the query in question. For instance, it is likely very difficult to obtain the information only about book sellers in Waterloo, Ontario, as it is hard to specify such a geographic constraint precisely in any existing search engine. Some search engines use additional criteria to limit the spatial extent of a query to a domain or a host. While this practice in fact allows a Web user to retrieve information based on the physical location of a Web server, it does not provide any support for locating a Web page logically based on its content. For instance, a server providing information to residents of one city may be physically located in another city or even another country. Thus, using a DNS domain as spatial information is insufficient.

Another interesting phenomenon is that many commercial Web sites offer what is often referred to as a “store locator” with which a Web user can locate a store using geographic constraints such as a province, a city, or a postal code. This practice presumably allows potential customers to spend hours browsing through various products on the Web without any human assistance, and to commit actual transac-

tions in the physical world with very little extra time and effort. Some Web sites push this functionality even further to the extent that the availability of the merchandise of interest is combined with geographic constraints, so that only stores that have the merchandise of interest and are located within a geographic area are listed. At the same time, the efficiency and availability of the Web and the vast amount of retrievable information do make the Web irresistible. One problem in this solution is that a Web user is allowed to choose only from a set of spatial entities pre-defined by a Web page, eliminating the possibility for the user of choosing a spatially flexible area of interest.

All of these problems can be addressed by converting the current content-oriented Web into a geographically-oriented Web, which can be done by developing a search engine and a browser capable of using geographic information properly. Components of this geographically-oriented Web are shown in Figure 5.1.

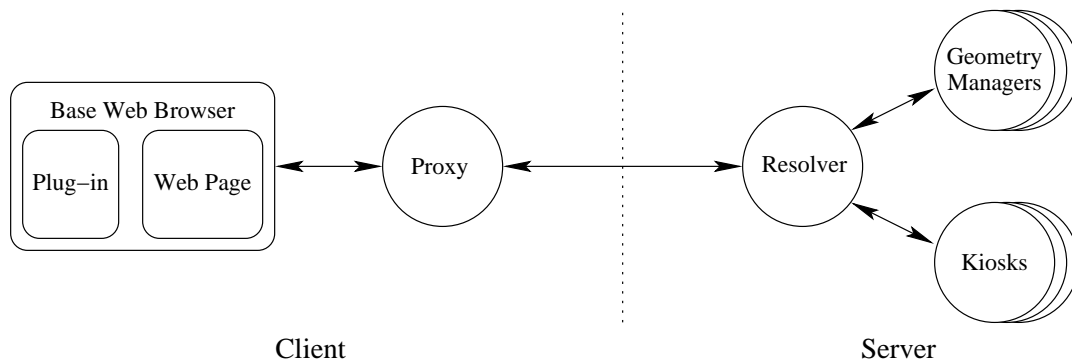


Figure 5.1: Components of a Geographically-Oriented Web

A Geographically-Oriented Search Engine

A geographically-oriented search engine can easily be created using the proposed architecture. This requires the owner of a Web page to define a string of keywords to describe the content of the page, and a service area for which the content is intended. For instance, a restaurant capable of selling and delivering pizza should probably set the **Keywords** to “pizza,” with a service area equal to the geographic region to which deliveries can be made. Obviously, the spatial data (the service area) is stored at a geometry manager, whereas the non-spatial data (the keywords and the URI of the Web page of the restaurant) is stored at a kiosk. In other words, each Web page has a piece of spatial information attached to it, and becomes both spatially and non-spatially searchable.

The services used in this case study are some of the food services available on the campus of the University of Waterloo. The non-spatial data have been manually harvested from the existing university Web pages to populate kiosk databases. For

```
mysql> CREATE DATABASE DC;
mysql> USE DC;
mysql> CREATE TABLE Kiosk
-> (ID BIGINT UNSIGNED NOT NULL PRIMARY KEY,
-> Keywords TEXT NOT NULL, INDEX KeywordsIndex (Keywords (100)),
-> URI TEXT NOT NULL);
mysql> INSERT INTO Kiosk (ID, Keywords, URI) VALUES
-> (10736906272839274627,
-> 'Bon Appetit: Chinese food, hamburger, fry, chicken burger, grill, sub',
-> 'http://www.foodservices.uwaterloo.ca/eateries.html#bonappetit'),
-> (10289729176372819736,
-> 'Tim Horton's: coffee, tea, cappuccino, doughnut, cookie, muffin, bagel',
-> 'http://www.foodservices.uwaterloo.ca/eateries.html#tims');
mysql> USE mysql;
mysql> INSERT INTO db (Host, Db, Select_priv) VALUES ('%', 'DC', 'Y');
```

Figure 5.2: Population of a Kiosk Database

example, Figure 5.2 shows how a kiosk associated with building DC can easily be created, populated, and granted proper privileges. Simplifying assumptions have been made about the service area of each service provider, so that the required service areas could be supplied to geometry managers. In particular, it has been assumed that the service area of each food service is the same as the service building. The spatial data are loaded into geometry managers from server maps in DWF format, each of which has been plotted from an original AutoCAD drawing containing the geometric definitions of neighbouring nodes, their URIs, and a short description. All three pieces of information are preserved in the process of plotting, and are loaded into geometry managers at run-time.

A Geographically-Oriented Browser

A specialized Web browser has also been developed to interact with the engine. As can be seen in Figure 5.3, the search engine allows a Web user to specify both spatial constraints (as an area of interest on the map) and non-spatial constraints (as keywords). As a Web user opens the browser, an initial page is loaded automatically, with a map displayed. If the user wishes to view a map of a larger or smaller scale, clicking on the appropriate “zoom-in” or “zoom-out” button will show the map with the next available scale. The user can first enter some keywords describing the desired content of the Web pages, then draw an area of interest limiting the spatial extent in which the pages are made available, click on the “search” button to send the criteria, and finally retrieve a list of links to the pages satisfying both the spatial and non-spatial requirements. Therefore, there is no ambiguity between spatial and non-spatial requirements, as they are specified separately. For instance, a user wish-

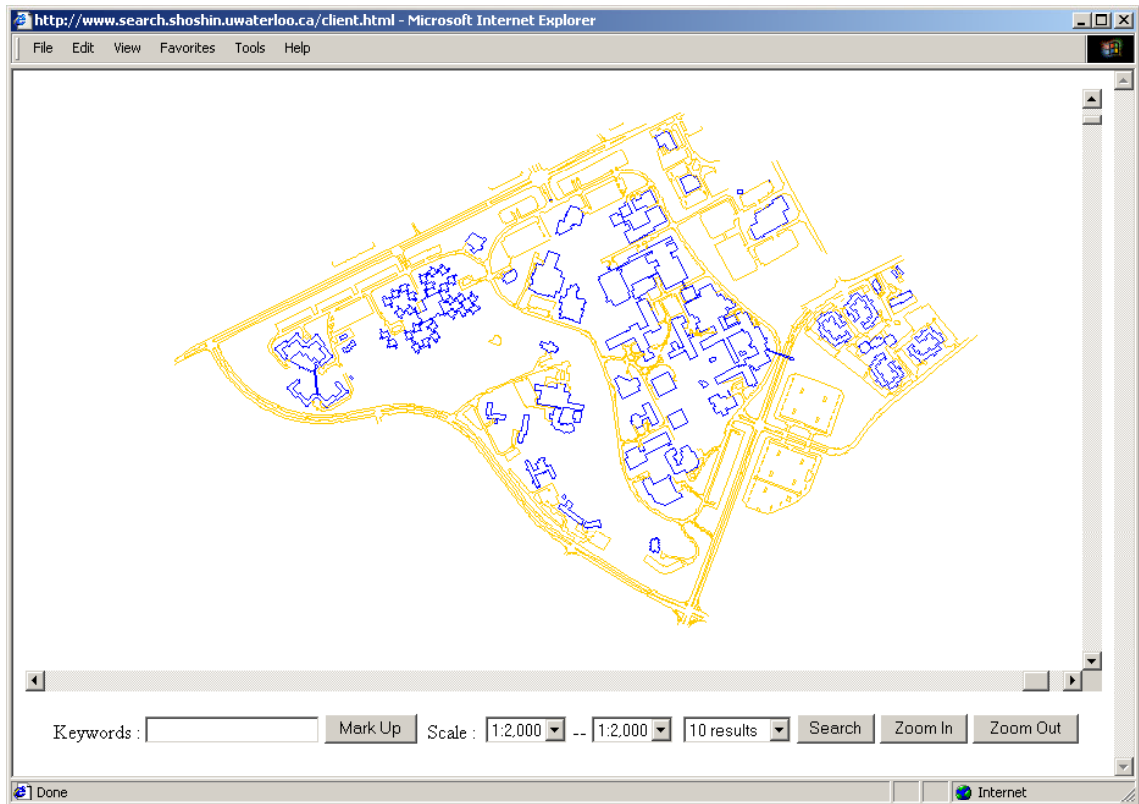


Figure 5.3: Geographically-Oriented Web Browser

ing to buy some Chinese food for lunch within one of three buildings can specify the query shown in Figure 5.4. Assuming that each service provider correctly specifies the spatial and non-spatial properties of its service, the specification of spatial and non-spatial requirements by the user ensures the retrieval of the Web pages with the desired spatial and non-spatial properties.

There are four components in this specialized browser: a base Web browser, a plug-in, a proxy, and a client Web page. The relationships among these components are shown in the left half of Figure 5.1.

Just as the resolver is the entry point into the architecture, the proxy is the entry point into the application. Upon receiving a request from the browser, the

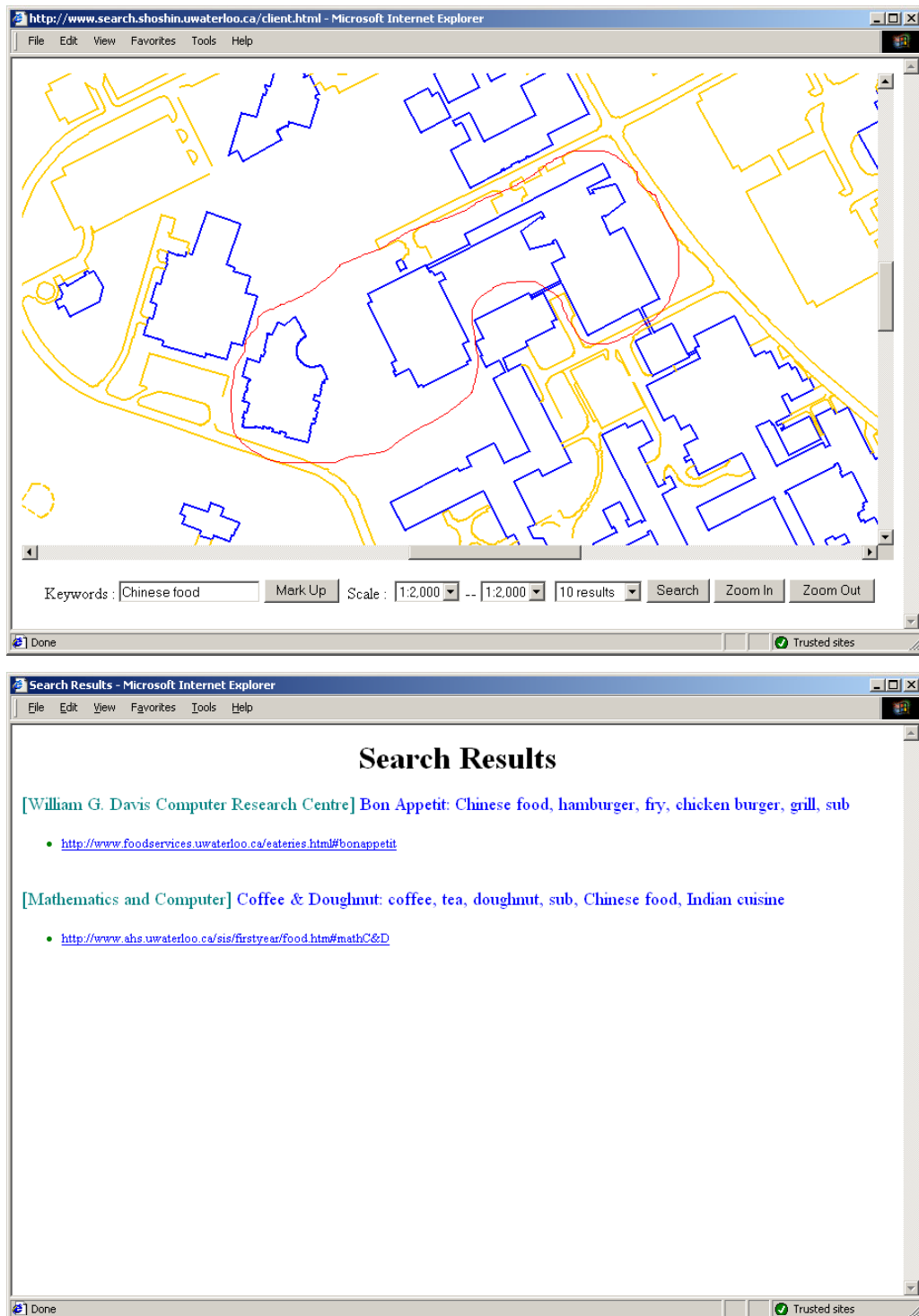


Figure 5.4: Chinese Food in Three Buildings

proxy first checks the host name of the target URI. Once an HTTP GET request is received at the proxy, the destination host name is checked against the trigger host name (`www.search.shoshin.uwaterloo.ca`), which is used in order for the proxy to identify queries with spatial and non-spatial constraints without creating a new protocol. If there is no match, the proxy fetches the requested page from the Web, and forwards the results directly to the base browser. If there is a match, an SQL query is created from the string of keywords specified by a user. Exactly how keywords should be structured and interpreted is beyond the scope of this thesis. Thus, in this particular implementation, a single white space is used as the delimiter between keywords, and the semantics of a set of keywords are assumed to be “discover all pages whose `Keywords` column contains all of the keywords, in any order.” For instance, the SQL query for selecting pages with the keywords “`Chinese_food`” is shown in Figure 5.5, returning all rows with their `Keywords` column containing the words `chinese` and `food` in any order. Once the query is constructed, the query

```
SELECT Keywords, URI FROM Kiosk
WHERE Keywords LIKE '%chinese%' AND Keywords LIKE '%food%';
```

Figure 5.5: SQL Query for the Keywords “`Chinese_food`”

and other spatial requirements are forwarded to the default resolver of the proxy in the format specified in Figure 4.3. In other words, the trigger host name signals an HTTP request containing both spatial and non-spatial constraints, which means that this request should be processed by the architecture. Once the query result reaches the proxy from the resolver, it is returned to the base browser. Needless to say, the base browser needs configuring to use the proxy as its HTTP proxy server for this to work.

The most crucial component is the Volo View Express plug-in, the browser plug-in capable of viewing, real-time panning and zooming, and marking up a DWF drawing. This freely available plug-in has been chosen for three reasons. First, the plug-in uses DWF as its native format, eliminating the need for any format conversion. Second, the viewing, real-time panning and zooming, and markup operations appear to have been implemented professionally. Finally and most importantly, its markup ability enables a Web user to mark up a DWF drawing to specify an area of interest graphically and intuitively. Although the plug-in is available for both Microsoft Internet Explorer and Netscape® Communicator, the version for Microsoft Internet Explorer has been chosen for its additional ability of using VBScript to control the plug-in. After a Web user draws an area of interest on the original map, the markup needs to be transferred to the proxy, and the only way to do this is to save the markup as a file on the local file system. Therefore, the plug-in and the base browser require full access to the local file system on the client machine, raising a serious security concern on the client machine. Fortunately, Internet Explorer allows four different classes of security settings, and by adding the trigger host name `www.shoshin.uwaterloo.ca` to the list of trusted sites, the required privilege is automatically granted. Furthermore, Internet Explorer must also be configured to allow unsafe ActiveX controls in order to eliminate a warning message after the “search” button is clicked.

A Web page is written in HTML and VBScript. This page fetches a map in DWF format, and provides users with a few control buttons. Once the page is loaded and displayed, a user can first navigate to choose a desired part of a map by panning, zooming in and out, and specifying an appropriate scale range. Then, the user can specify a highly customized area of interest by drawing a polygon on the map. Clearly,

the user can specify the area according to need, preference, or other constraints. For instance, a user with a car is more likely to specify a larger area than a user on foot. Finally, the user can specify how many results should be returned by the server. This entry is again highly customized. A user with a device having very limited bandwidth may choose to get the first 10 results, whereas a server with broadband connectivity may choose to download as many results as possible. The VBScript code is used to retrieve the markup from the local file system, and is also used to construct a multipart form for upload with input from the user.

Advantage

The application allows Web users to discover information on the Web using an approach other than the current content-based approach, making spatial information a separate type of search criteria. This separation eliminates any ambiguity between spatial and non-spatial requirements, and thereby ensures the correct interpretation of the requirements.

Disadvantage

Since the logical service area of each Web page is dependent on the content and nature of the Web page, a human operator must understand and define the service area of a Web page, and register with the architecture correctly. Therefore, this task cannot be automated in general. Unfortunately, the success of various search engines hints at the very need for this type of automation, *i.e.*, a search engine to obtain information automatically. Being so used to publishing only a Web page, Web page designers are likely to have trouble adapting to the scheme of providing one extra

piece of information.

In this case study, the proposed architecture is used to transform the existing World-Wide Web into a geographically-searchable Web where queries with spatial and non-spatial constraints can be processed. A geographically-oriented search engine is realized using the architecture, and a geographically-oriented Web browser is constructed using a proxy and a base browser with a plug-in. It is assumed that each kiosk contains the descriptions of Web pages, *i.e.*, some semantics have been given to the architecture. In the next case study, more specific semantics are attached to the architecture, resulting in a special case of this case study.

5.2 Service Discovery in the Physical World

The semantics given to the architecture determines the nature of an application. In this case study, hardware and software configurations are identical to the previous case, with an additional assumption that the service area of each Web page represents the physical location of the object in question in the physical world. That is, objects in the physical world can be mapped into Web pages, which in turn can be represented by URIs. This means physical objects now have a Web presence, and can therefore be indexed and searched just like regular Web pages, as in the previous case study. Furthermore, discovering a Web page using geographically-oriented constraints under this assumption becomes equivalent to locating an object in the physical world. Thus, the architecture can locate objects in the physical world by solving a special case of the problem of information retrieval on the World-Wide Web, and the problem of resource discovery in the physical world can be transformed and generalized into a problem of information retrieval on the Web.

Motivation

There are many different resources available in the physical world, and people rely on these resources to obtain products and services. Gas stations, post offices, and banking machines all qualify as resources. However, the subtle gap between the availability of resources and the awareness of people often renders wanted but unnoticed resources inaccessible. That is, people cannot access resources to obtain products or to receive services simply because the very existence of the resources is not brought to their attention. This gap will endure as long as people can find themselves in unfamiliar environments. On the one hand, newcomers, visitors, and tourists naturally belong to this category. On the other hand, anyone can fall victim to an unfamiliar environment occasionally.

While in unfamiliar environments, people may ask many different questions, yet one type of question comes most frequently. An example is “Where can I get some cash around here?” As many services in the physical world require physical presence of the service requesters at the service providers in order for the service to take place, it is often useful to locate the service providers. Traditionally, paper maps are the tool to help people find resources. The locations of service providers are drawn on a map as symbols according to their types, and service requesters identify the symbols corresponding to potentially interesting resources on the same map. Both providers and requesters also perform the mapping between a map and the physical world, and achieve a form of communication via the map.

Paper maps are not flexible enough in at least three ways. First, the searchability of paper maps is quite limited. For instance, imagine trying to locate a small street in a busy downtown area of a city. One would have to first look up the street name

from the street name index, turn to the correct page, locate the correct grid on the page, and look visually for the street in the grid. Each search in this process except the last one can be viewed essentially as a “human” binary search with the last search having no better algorithm than “skimming.” All of these searches are performed by a human map user and therefore tend to be slow and error-prone, compared to their automated counterparts. Second, generality and clarity are conflicting requirements for paper maps. While generality requires maps to contain as much information as possible, clarity implies the necessity for keeping them simple to reduce the effort demanded of users for a search. Finally and most importantly, the conflict between the dynamic nature of the world and the static nature of paper maps often results in the maps being out of date very quickly. All three limitations are intrinsic to the media, paper, and they lead to the development of digital maps.

Digital maps have overcome many limitations inherent in paper maps. A digital map almost always comes with a search function, which greatly reduces the amount of effort required from a human map user. Map data is often structured into layers so that unwanted layers can be disabled at run-time and become invisible to a map viewer. Therefore, generality and clarity are no longer conflicting requirements. However, digital maps are still neither open nor dynamic. They should be open so that information can be added and removed with ease. For example, the information about hotels should be able to be added to a map of restaurants. Currently, this task is administrated centrally by a map producer, which is becoming time-consuming and unscalable, especially as map scales become larger and larger. Furthermore, dynamic information which requires updating on a regular or real-time basis cannot be kept conveniently in such a static media.

The architecture can be used to address some of the unsolved problems, and the data used in this case study are printers in a hallway.

Data Source

A printer is usually considered a type of local resource, to which physical access is often critical for users to receive printouts. Therefore, the physical location of a printer becomes essential, and vicinity supersedes functionality. For instance, the laser printer next door offers a printing resolution of 300 DPI and another laser printer in the next building offers a printing resolution of 600 DPI. Unless high-quality printing is required, *e.g.*, when a graphic printout is to be produced, it is unlikely for one to spend the extra time walking to the other building to achieve the better quality. The importance of the physical locations makes printers an excellent choice for this case study.

The location of each printer is taken directly from the DNS resource records available from the local DNS server. Unlike many other domains, the DNS server of the University of Waterloo domain maintains extra information in these records, which can be seen from Figure 5.6. From this record, useful information can be obtained.

lj5m-sho.uwaterloo.ca	86400	IN	TXT	"ORGUNIT Math,CS,Shoshin"
lj5m-sho.uwaterloo.ca	86400	IN	TXT	"ADMIN jpblack"
lj5m-sho.uwaterloo.ca	86400	IN	TXT	"CONTACT pasward"
lj5m-sho.uwaterloo.ca	86400	IN	TXT	"LOCATION DC,3552D,,"
lj5m-sho.uwaterloo.ca	86400	IN	TXT	"DATE 2001-03-16"
lj5m-sho.uwaterloo.ca	86400	IN	HINFO	HP LaserJet-5M Printer
lj5m-sho.uwaterloo.ca	86400	IN	A	129.97.105.254

Figure 5.6: DNS Resource Record of `lj5m-sho.uwaterloo.ca`

This includes the IP address, the faculty, department and group it belongs to, and

administrative and contact information. In addition, there are two pieces of the most important information for this case study: the location (`LOCATION MC,3552D,,`) and the host information (`HP LaserJet-5M Printer`). These entries indicate there is a Hewlett-Packard LaserJet 5M printer physically located in Davis Centre, room 3552D, providing the spatial and non-spatial data needed to populate geometry managers and kiosks. A granularity of room is created by setting up a kiosk for each room of the hallway. For instance, as shown in Figure 5.7, the kiosk for room 3552D is created and populated with the non-spatial data for the printer. The URI used in the kiosk

```
mysql> CREATE DATABASE DC3552D;
mysql> USE DC3552D;
mysql> CREATE TABLE Kiosk
-> (ID BIGINT UNSIGNED NOT NULL PRIMARY KEY,
-> Keywords TEXT NOT NULL, INDEX KeywordsIndex (Keywords (100)),
-> URI TEXT NOT NULL);
mysql> INSERT INTO Kiosk (ID, Keywords, URI) VALUES
-> (10003872725389293728,
-> 'LaserJet 5M printer',
-> 'http://www.hp.com/cposupport/prodhome/lj5m.html');
mysql> USE mysql;
mysql> INSERT INTO db (Host, Db, Select_priv) VALUES
-> ('%', 'DC3552D', 'Y');
```

Figure 5.7: Database corresponding to Kiosk DC 3552D

provides information only about the properties of the printer, not about the printing services being offered, since the information is supposed to be used only to locate the printer physically, but not to connect to the printer electronically.

The process of gathering spatial and non-spatial data can be automated for all IP-enabled printers on the university campus, and the data can be extracted easily from the well-structured DNS records. However, a human operator is needed to transfer the spatial data manually into an AutoCAD drawing, from which DWF files are

produced. Even though the data insertion for each printer only requires a few clicks, a large number of printers results in a fair amount of manual work. Since this case study is used as a proof-of-concept, only three printers are used, and the dynamic update of spatial data is considered to be possible future work.

Assuming most Web pages are maintained independently, this case illustrates that the problem of service discovery in the physical world can be solved using the proposed architecture with a dynamic and open solution. The dynamic and open properties of the solution come directly from the dynamic and open nature of the World-Wide Web and of the Internet as a whole. Furthermore, service providers can minimize stale data by updating their service entries at each kiosk on their own. This reduces the management effort on any central authority, as is the case for digital maps.

5.3 Service Discovery on the Internet

The proposed architecture is intended for multiple types of service to co-exist without interference among themselves. Unlike the first two case studies, this one is only possible when no additional semantics are assumed. That is, a service listed at a kiosk may be a printing service, an information service, or an unknown service yet to be defined. This case study therefore becomes the most general case of all, covering the first two cases. More importantly, this case study demonstrates a possible solution for service discovery on the Internet.

Motivation

The problem of service discovery has most often been addressed in the context of local-area networks and local domains, but rarely in that of wide-area networks, because it

is technically possible but realistically impractical to perform resource discovery in a wide-area context: mechanisms needed for resource discovery, such as broadcast and multicast, are simply not feasible.

Broadcast on an IP network is effectively restricted to an IP subnet. In the current version of IP, datagrams destined for the limited broadcast address 255.255.255.255, are never forwarded outside a network or subnet, but are discarded silently at the router. Furthermore, routers can be configured to forward or to discard datagrams destined for network-directed and subnet-directed broadcast addresses. In practice, a broadcast datagram has a very small chance of leaving its originating network or subnet.

The limited coverage of multicast over the Internet makes the usefulness of multicast in a wide-area context questionable. For example, Jini[™] Network Technology multicasts UDP datagrams for service requesters to discover Jini lookup services, and for service providers to announce their presence. Although it is in theory possible for the multicast datagrams to leave the originating networks, this approach is considered a mechanism for locating resources on a local network and in local domains, presumably under the assumption that multicast datagrams are unlikely to propagate to remote networks and domains in reality.

It can be concluded that multicast- and broadcast-based resource discovery mechanisms can only be employed on local networks and in local domains, which suggests the need for an alternative mechanism for locating resources across multiple domains on wide-area networks. The inseparable nature of the wide-area Internet and the earth inevitably suggests using spatial information as the major search index. Therefore, the proposed architecture can be used to implement a service discovery mechanism

using geographic constraints on the wide-area Internet.

Design

This case has not been implemented, but the idea behind it is outlined briefly here. On the one hand, spatial data are stored and used the same way as specified in Chapter 3, *i.e.*, service providers define service areas whereas service requesters specify areas of interest. On the other hand, non-spatial data are stored a bit differently.

The most crucial issue in the storage and retrieval of non-spatial data is how to structure the **Keywords** column to describe service properties. That is, a syntax is required to describe services at each kiosk. While only one type of attribute (**URI**) is required in the architecture, applications are free to choose any syntax for all other non-spatial attributes (**Keywords**). Exactly what scheme is used is irrelevant, and is intentionally left unspecified so that various applications will have enough room to manoeuver. For instance, one possible way of describing a printer capable of printing at 300 DPI or 600 DPI on letter paper using the Postscript language or Printer Command Language is shown in Figure 5.8. Based on this syntax, one

```
(Service=Printing);  
(Resolution=300 DPI,600 DPI);  
(Paper Size=Letter);  
(Language=Postscript,Printer Command Language)
```

Figure 5.8: Properties of a Printer

possible SQL query to select a Postscript printer is shown in Figure 5.9. It is worth noting that the choice of the syntax does have performance impacts on the database engines. However, since each kiosk is assumed to be a database holding a relatively

```
SELECT Keywords, URI FROM Kiosk
WHERE Keywords LIKE '%(service=printing)%' AND
      Keywords LIKE '%(language=%postscript)%';
```

Figure 5.9: Selection of a Postscript Printer

small amount of non-spatial data, these impacts are assumed negligible, and therefore beyond the scope of this thesis.

Thus, while a combination of `Keywords` and `URI` allows users to locate services with desired properties, spatial constraints enable the users to narrow the choices in the spatial extent. In other words, the architecture can be used to discover services with the correct spatial and non-spatial properties.

Spatial constraints are specified at the client using a different approach. Rather than marking up on a map, the spatial constraints can be constructed automatically by a client. For instance, the client can obtain its current location from a GPS receiver, and construct an area of interest using this current location, bypassing any user interaction.

In the most general case, no semantics are given to the architecture, and the application is therefore the most general of the three. The first two cases can be treated as special cases of the last.

Chapter 6

Conclusions

This thesis explores the possibility of discovering services on the Internet using geographic criteria. It is a more natural approach to wide-area service discovery than approaches based on the logical structure of the Internet. An architecture is proposed to realize this idea within the existing Internet infrastructure. As such, the architecture can be deployed incrementally on the Internet. Users can query the system using both spatial and non-spatial constraints, and retrieve a list of URIs pointing to resources satisfying the constraints. This architecture can be used to create a mapping between the physical world and the Internet, so that people can explore the Internet and the Web using geographic constructs. Conversely, people can become more aware of their environments by obtaining services and information from the Internet.

The geometry managers and resolvers of the proposed architecture have been implemented in Java, whereas the kiosks are simply normal MySQL databases. Different semantics can be attached to the proposed architecture to develop useful applications, and three case studies are developed to illustrate this idea, ranging from the most general case to a very specific case. In particular, the first case realizes a

geographically-oriented Web by implementing a search engine and a browser, both capable of using spatial and non-spatial data. The second case attempts the problem of service discovery in the physical world by reusing the software from the previous case with an additional assumption that the service area of each service is in fact the physical location of that service. It follows that the problem of service discovery in the physical world can be solved as a special case of a geographically-oriented search on the world-Wide Web. Finally, the unimplemented last case generalizes the first two cases by eliminating all assumptions on the semantics, making the architecture capable of holding data about any service. Not surprisingly, despite the work done in this thesis, there is still room for improvement.

6.1 Future work

Although the architecture has been designed to be general, flexible, and open, extensions can be added to it to enhance its ability to meet even more demanding requirements of applications. These include the syntax of **Keywords** to store different types of service efficiently, enhanced spatial accuracy of queries, and overall performance of the system.

6.1.1 Syntax of Keywords

The syntax of **Keywords** determines how non-spatial data is stored at the kiosks, which also raises the question whether multiple types of service can co-exist efficiently without interference among themselves. For example, Section 5.3 suggests an SLP-like syntax in order for multiple types of services to co-exist. However, this

approach requires that the same syntax must be used among application developers, and therefore the standardization of the syntax is likely needed to ensure that there is no conflict among various service types. It is not clear what performance impact this syntax may have, or what a more efficient scheme is like. Thus, one possible study is to examine the performance impacts of various syntaxes on a database engine, and to determine the best syntax for optimal performance.

6.1.2 Enhanced Spatial Accuracy

As mentioned in Section 3.3, the proposed architecture processes spatial data with the granularity of the size of each kiosk. That is, no spatial data is processed further than within a kiosk, implying that there is no way of differentiating one service from another spatially within the service area of a kiosk. Under certain circumstances, such a distinction is needed. For instance, it may be desirable to distinguish one machine from another within the same research lab, but at the same time, it is not desirable to create multiple kiosks for multiple machines. Thus, extra spatial data are required for each service.

One possible extension is to modify the current schema of a kiosk table to add a fourth column to store an optional spatial correction term for each service within the kiosk. For instance, if a service area covers only half of a kiosk, the correction term will indicate the discrepancy between the service area of the kiosk and the service area of that service. This modification potentially requires each kiosk to store some spatial data, and therefore breaks the separation of spatial and non-spatial data. However, this problem can be solved partially by each kiosk storing correction terms as non-spatial data, thereby avoiding the requirement of any special software at each

kiosk. Instead, the correction terms can be processed at each resolver, where spatial and non-spatial query results are combined. Clearly, this will add complexity to the resolvers, and in particular, the resolvers will need to understand the query results. The trade-off between an enhanced spatial accuracy and additional complexity would require further study.

6.1.3 Performance

Due to the distributed nature of the proposed architecture, the extensive use of communication channels is likely to introduce significant latency into the process of answering a query. While concurrency has been used to reduce the impact of this latency, other techniques are likely required to improve the performance further. For instance, a caching scheme can be developed to reduce the frequency of consulting a remote geometry manager, which is similar to the way DNS servers and slaves cache domain information from remote domains.

6.1.4 Data Integration

Geographic Information Systems (GIS) have gained in popularity over the past few years with their ability to combine and analyze the integrated form of multiple types of data. Although GIS offers the integration of spatial data and non-spatial data, it is not designed to scale to the size of the Internet in general. However, data stored in GIS can be a valuable asset to the proposed architecture, and the integration of GIS into the architecture and the interaction between them would be a subject for further study.

Bibliography

- [1] Gregory D. Abowd, Christopher G. Atkeson, Jason Hong, Sue Long, Rob Kooper, and Mike Pinkerton. Cyberguide: A Mobile Context-Aware Tour Guide. *Wireless Networks*, 3(5):421–433, October 1999.
- [2] Gregory D. Abowd, Christopher G. Atkeson, Jason Hong, Sue Long, Rob Kooper, and Mike Pinkerton. Cyberguide: A Mobile Context-Aware Tour Guide. Technical Report GIT-GVU-96-27, Graphics, Visualization and Usability Center, Georgia Institute of Technology, September 1996.
- [3] T. Berners-Lee, R. Fielding, and H. Frystyk. Request for Comments: 1945, Hypertext Transfer Protocol – HTTP/1.0, May 1996.
- [4] Guanling Chen and David Kotz. A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Department of Computer Science, Dartmouth College, 2000.
- [5] Keith Cheverst, Nigel Davies, Keith Mitchell, and Adrian Friday. The Role of Connectivity in Supporting Context-Sensitive Applications. In *Proceedings of the First International Symposium on Handheld and Ubiquitous Computing*, pages 193–207, Karlsruhe, Germany, September 27–29, 1999.

- [6] Keith Cheverst, Nigel Davies, Keith Mitchell, and Adrian Friday. Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking*, pages 20–31, Boston, Massachusetts, United States, August 6–11, 2000.
- [7] Keith Cheverst, Nigel Davies, Keith Mitchell, Adrian Friday, and Christos Efstathiou. Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences. In *Proceedings of the CHI 2000 Conference on Human Factors in Computing Systems*, pages 17–24, The Hague, Netherlands, April 1–6, 2000.
- [8] Keith Cheverst, Keith Mitchell, and Nigel Davies. Design of an Object Model for a Context Sensitive Tourist GUIDE. *Computer and Graphics*, 23(6):883–891, December 1999.
- [9] Nigel Davies, Keith Cheverst, Keith Mitchell, and Alon Efrat. Using and Determining Location in a Context-Sensitive Tour Guide. *Computer*, 34(8):35–41, August 2001.
- [10] Nigel Davies, Keith Cheverst, Keith Mitchell, and Adrian Friday. ‘Caches in the Air’: Disseminating Information in the Guide System. In *Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications*, pages 11–19, New Orleans, Louisiana, United States, February 25–26, 1999.
- [11] Svetlana Domnitcheva. Location Modeling: State of the Art and Challenges. In *Proceedings of the Workshop on Location Modeling for Ubiquitous Computing*, pages 13–19, Atlanta, Georgia, United States, September 30, 2001.

- [12] Samir Goel and Tomasz Imielinski. DataSpace – Querying and Monitoring Deeply Networked Collections in Physical Space, Part II – Protocol Details. Technical Report TR2000-400, Department of Computer Science, Rutgers University, 1999.
- [13] E. Guttman, C. Perkins, J. Veizades, and M. Day. Request for Comments: 2608, Service Location Protocol, Version 2, June 1999.
- [14] Tomasz Imielinski and Samir Goel. DataSpace – Querying and Monitoring Deeply Networked Collections in Physical Space, Part I – Concepts and Architecture. Technical Report TR2000-381, Department of Computer Science, Rutgers University, 1999.
- [15] Tomasz Imielinski and Samir Goel. DataSpace: Querying and Monitoring Deeply Networked Collections in Physical Space. *IEEE Personal Communications*, 7(5):4–9, October 2000.
- [16] Rui José and Nigel Davies. Scalable and Flexible Location-Based Services for Ubiquitous Information Access. In *Proceedings of the First International Symposium on Handheld and Ubiquitous Computing*, pages 52–66, Karlsruhe, Germany, September 27-29, 1999.
- [17] Sue Long, Rob Kooper, Gregory D. Abowd, and Christopher G. Atkeson. Rapid Prototyping of Mobile Context-Aware Applications: The Cyberguide Case Study. In *Proceedings of the Second Annual International Conference on Mobile Computing and Networking*, pages 97–107, Rye, New York, United States, November 11–12 1996.

- [18] L. Masinter. Request for Comments: 2388, Returning Values from Forms: multipart/form-data, August 1998.
- [19] Julio C. Navas and Tomasz Imielinski. GeoCast – Geographic Addressing and Routing. In *Proceedings of the Third Annual International Conference on Mobile Computing and Networking*, pages 66–76, Budapest, Hungary, September 26–30 1997.
- [20] Sun Microsystems, Inc. Jini Architectural Overview, 1999.
- [21] J. Veizades, E. Guttman, C. Perkins, and S. Kaplan. Request for Comments: 2165, Service Location Protocol, June 1997.
- [22] Roy Want, Andy Hopper, Veronica Falcão, and Jonathan Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems*, 10(1):91–102, January 1992.
- [23] Randy Jay Yarger, George Reese, and Tim King. *MySQL and mSQL*. O’Reilly & Associates, Inc., 101 Morris Street, Sebastopol, CA 95472, U.S.A., July 1999.