Maintaining Quality of Service for Adaptive Mobile Map Clients

by

Wegdan Ahmad Elsay Fouad Abdelsalam

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, 2001

©Wegdan Abdelsalam, 2001

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

Mobile devices must deal with limited and dynamically varying resources, in particular, the network quality of service (QoS). In addition, wireless devices have other constraints such as limited memory, battery power, and physical dimensions. Applications that execute in such environments need to adapt to the dynamic operating conditions in order to preserve an acceptable level of service as close to 100% of the time as possible.

Viewing and downloading digital spatial data on mobile devices has become more popular, especially with the availability of “location-aware” applications that exploit GPS (Global Positioning System) receivers integrated in many of today’s mobile devices. Map client applications face many challenges in accessing data across a wireless network. Vector spatial data files tend to be large, and file sizes tend to increase unpredictably depending on the complexity of feature geometry. Due to the limited size of the mobile device display, viewing all the details of the map could cause unreasonable clutter and render the map useless. Even if it is feasible to transmit all the details from a QoS standpoint, this could pose a problem from a usability standpoint.

This research effort aims to tackle the issues of QoS and usability on mobile devices through a client-proxy-server model where clients are on mobile devices. The proxy performs two functions. First, it supplies the client with vital data about the status of the system that allows the client to take adaptive decisions aimed at maintaining the QoS. Second, it performs the adaptive actions requested by the client. There are two types of adaptive actions performed by the proxy, activating and deactivating filters. When filters are activated, the amount of data transmitted from the server to the client is reduced. The client may decide to activate one or more filters either to maintain QoS or to limit clutter on the screen and enhance usability.

The map client-server application and the proxy were developed in Java^{tm}, and a number of experiments and real-life scenarios were designed to determine the effectiveness and feasibility of the proposed adaptation model and to evaluate the performance of the proxy.

Acknowledgements

I must first thank my supervisor Professor Jay Black. Both his technical advice and guidance were indispensable in completing this thesis. I am also grateful for his extreme patience with me, his great support and encouragement for my studies here at Waterloo.

I am grateful to my readers, Professor David Taylor and Professor Ric Holt for reviewing my thesis. I appreciate both the time they have taken and their valuable suggestions.

I would like also to thank all members of the Shoshin research group at the University of Waterloo. They provided the necessary research environment to motivate me.

Above all, I also thank my husband Yasser Ebrahim for his great support, love, patience and encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Mobile devices, ranging from handheld computers to web-enabled cellular phones and pagers, are becoming more popular these days as they have many advantages over traditional notebook computers. Size is an obvious advantage. Another advantage is the friendly user interface these devices have compared to the interface to the operating system running on the average notebook. Cost is another important factor. With prices in the low to mid hundreds of dollars, Personal Digital Assistants (PDAs) are much cheaper than notebook computers that typically cost well over a thousand dollars.

Mobile devices must deal with limited and dynamically varying resources; in particular, the network quality of service (QoS). This QoS includes network availability, network bandwidth, latency and cost. Having a cheap, reliable, low-latency, high-bandwidth connection will be the exception rather than the rule, and the system will most probably use networks that operate intermittently, have low bandwidth and high latency, and are expensive. In addition to the network, wireless devices have other constraints like limited memory and battery power, and physical dimensions [20]. Applications that execute in such environments need to adapt to the dynamic operating conditions in order to preserve an acceptable level of service as close to 100% of the time as possible.

Viewing and downloading digital spatial data on mobile devices has become more popular, especially with the availability of “location-aware” applications that exploit GPS (Global Positioning System) receivers integrated in many of today’s mobile devices. Map-client applications face many challenges in accessing data across a wireless network. Vector spatial data files tend to be large, and file sizes tend to increase unpredictably depending on the complexity of feature geometry. Users requesting vector data often require a less detailed version than the original to “look right” on the mobile device display; their first operation upon receipt is to simplify or generalize the vector data. It would be ideal to transmit vector data at the resolution at which a user requests it and save the client time, sending first a very coarse version for browsing, and subsequently sending more details when the user zooms in.

1.1 Adaptation

The need for application adaptation in mobile and wireless environments is well established [4, 16, 20, 27, 43, 44]. Many approaches to adaptation have been proposed before, and many taxonomies of adaptation are possible. We focus here on the types of adaptation policies as well as the location of the implemented adaptation.

Adaptation policies can be grouped into two types: data and control [14]. Data adaptations transform the application data. For instance, they transform the images in a document into a lower-resolution format or display a map with less detail. Control adaptations modify the application control flow (i.e., its behavior). For instance, a control adaptation could cause an application, which has a networked mode and a stand-alone mode, to change its mode depending on network availability.

Based on the location of the implemented adaptation, we recognize a spectrum of possibilities with two extremes: system-based adaptation [21, 22, 25] and application-based adaptation [29, 36]. With system-based adaptation, the system performs all adaptation by interposing itself between the application and the data. System-based adaptation does not require modification of the applications, and provides centralized control, allowing the system to adapt several applications according to a system-wide policy. With application-based adaptation, only the application is changed; the system is unaware of any adaptation. Application-based adaptation allows both data and control adaptation, while system-based adaptation is limited to data adaptation.

Another approach that tries to strike a middle ground between system- and application-based adaptation is application-aware adaptation [38-40]. Here, the system provides some common adaptation facilities, and serves as a centralized locus of control for the adaptation of all applications. The applications are modified to implement control adaptations and to perform calls to an adaptation API provided by the system.

A novel approach to adaptation is called component-based adaptation [14]. It enables application-specific control and data adaptation policies without requiring modifications to the application. It does so by using the exposed APIs of component-based applications and the structured nature of the documents they manipulate to implement application-specific control adaptation policies. Component-based adaptation attempts to bring together the benefits of system-based and application-based adaptation, namely to implement application-specific policies, but without modifying the applications. Since adaptation is done in the system, component-based adaptation retains the advantage of providing a centralized locus of control for adaptation of multiple applications.

In this research, we are concerned with application-based adaptation where the application is modified to perform calls to the Communication Manager for Mobile Applications (Comma) [28]. Comma is a tool to help adaptive client applications monitor the environment and alter computation and communication behavior accordingly. The monitoring is handled by one Comma component, the Execution Environment Monitor (EEM) and the altering of the stream is handled by a second component, the Stream Filtering Proxy (SFP).

1.2 Data Fidelity

The data available to a mobile client from a remote server should be identical to that available to the application if it were run on a system reliably connected to the server. This state is difficult to achieve in a mobile system, especially when network characteristics vary so widely. Using application-based adaptation, some degradation in the data presented to the application is expected. This motivates the notion of fidelity [37], defined as the degree to which data presented at a client matches the reference data at the server.

Fidelity is a type-specific notion; different kinds of data are degraded in different ways. For example, video may be degraded by dropping frames, reducing image size, or reducing the image quality of individual frames. Maps can be degraded by either coarsening the level of detail or removing certain classes of features; for example, one might only look at roads, ignoring buildings.

This notion of fidelity requires data to be relatively rich in structure; an untyped stream of data cannot be degraded in any meaningful way [37]. That is why we chose to work on a client-server mapping application, which by its nature deals with maps, each of which contains a large data set of objects of varying types.

1.3 The Problem

Viewing and downloading digital spatial data via the Internet has become more popular especially with the availability of “location-aware” applications that exploit GPS (Global Positioning System) receivers integrated in many of today’s mobile devices. As users expect more frequent data updates, demands for Internet access to geographic spatial information should continue to increase.

Map-client applications face many challenges in accessing data across a wireless network because a mobile client has to be compact and lightweight, it is typically resource-poor relative to desktop clients. Network connectivity, especially via wireless media over a large area, tends to vary considerably in bandwidth, latency, reliability and cost. Limited storage space on PDAs and handheld computers is also a major constraint, especially with the lack of secondary storage devices on most of these machines.

The issues in Internet transmission of vector spatial data remain a challenge. Amongst these factors, bandwidth continues to be the primary limiting factor for mobile computing. While the typical bandwidth available nowadays may be sufficient for text-based communication, it may be not be sufficient for graphics-intensive mapping purposes [47].

Vector files tend to be large, and file sizes tend to increase unpredictably depending on the complexity of feature geometry [6].

Even quite small mapping applications use an enormous amount of data. For example, the data representing a 7,500,000 : 1 map of Canada, with an accuracy of about 1 km, showing merely the major waterways, a few cities and roads, plus individual place names as points, takes over 10 Mbytes [1]. If the user needs access to more than one map at a time, the problem is aggravated even further.

This conflict between the limited resources available to PDAs on wireless networks and the sheer size of data that needs to be handled by mapping applications creates a situation where it could be difficult to meet the users' needs in a timely fashion, if at all. In this research effort we explore ways to make it possible for the user to have some degree of service that varies with the availability of resources

1.4 Contributions

This effort is aimed at showing how application-based adaptation could be used to handle the scarcity of resources in wireless environments.

Adaptation is an approach to handling limited resources. Instead of an all-or-nothing approach, an adaptation-based system assumes that the user is not willing to wait for the required resources to become available and is instead willing to accept degraded service. By reducing the degree to which the application uses the resource, it will be better able to cope with the application requirements and provide some level of service.

For a mapping application to adapt, it may have to reduce the data fidelity of the map. Users requesting vector data often require a less detailed version than the original if the map is to "look right" on the mobile device display [5]. Therefore it would be ideal to transmit vector data at the low resolution the user requests which will also improve the response time. This can be achieved by first sending a very coarse version for browsing while details can be subsequently sent as the user zooms in.

We lower data fidelity by using map generalization techniques. Map generalization is the process of reducing detail on a map. Fidelity can be lowered in two ways: coarsening and straining. Coarsening means reducing the level of detail in the map, which could be done by filtering out objects that are small relative to the window size. Straining removes certain classes of features from the map. For example, one might only look at roads and rivers, ignoring buildings. This results in a reduction in the amount of data that the network has to transmit from the server to the client, which translates into a faster completion of the transmission operation at the expense of the quality of the map.

Using these map generalization techniques, we devised a set of filters. These filters were designed with the goal of limiting the amount of data traveling over the network from the server to the portable client. The filters have parameters that are used to determine the degree of filtering achieved. This allows for finer grained adaptation which makes for a better balance between QoS and map quality.

Sometimes the reduction in the quality of the map resulting from a lower fidelity level could be desirable. One such situation is when displaying all details of the map means an over-crowded display that is too cluttered to be useable. We use the same set of filters to actually enhance the usability of the map by lowering the amount of detail displayed as the user zooms out, and increasing it as the user zooms in. This dynamic adaptation to the zooming factor enables the user to have enough details displayed at any time, without unnecessary detail that might actually hinder the usability of the map.

For the main purpose of activating and deactivating filters, we developed a Stream Filtering Proxy (SFP). As the client takes adaptive decisions, it asks the SFP to carry out these decisions by activating or deactivating filters.

One important characteristic of adaptation-based systems is their ability to adapt dynamically to changes in the environment at runtime. For this to be possible, constant monitoring of the environment variables is required. Whenever one or more of these variables passes a predetermined threshold, adaptive action is taken to remedy the situation.

One responsibility of the proxy, besides lowering data fidelity, is to inform the map client of the changes in the environment, to allow the client to adapt by lowering or increasing data fidelity.

We conducted a series of experiments to verify the effectiveness and feasibility of the filtering techniques used and their implementations. We tested individual filters and combinations of filters. From the test results, we developed some heuristics related to filters, their combinations, and their parameters to be used to achieve our goals of maintaining network QoS and map usability in different situations.

While we did not test the dynamic adaptation aspect of the system due to time limitations, we provided a model for such adaptation in respect to mapping applications running on mobile devices.

A map client-server application and a proxy were developed in Java^{tm}. A number of experiments and real-life scenarios were designed to determine the effectiveness and feasibility of the proposed adaptation model and to evaluate the performance of the proxy.

1.5 Chapter Organization

Chapter 2 gives background on wireless computing and geographic information systems. Chapter 3 describes in detail the architecture design and implementation. Chapter 4 shows the experimental results. The results are summarized in Chapter 5 along with suggested future work.

Chapter 2

Background

This research concentrates on mapping applications and wireless communication. What follows is a brief overview of the two areas, plus a specific description of the EEM as an adaptation tool.

2.1 Mobile Computing and Wireless Networks

Mobile computers, such as laptops and PDAs, can communicate over a wireless channel. This allows a person to travel while maintaining connections with other machines, or to run networked applications in locations that do not have any wired network support (e.g. outdoors).

Figure 1 from [34] shows a typical mobile computing environment. The coverage area could vary widely: it could be a small room or a large city. The area contains several antennas which represent transceivers, known as base stations. The base stations are usually connected to a wired network, and communicate with mobile computers over a wireless link. Wireless communication can be carried over various wavelengths: wide-area networks generally use radio frequencies, while indoor LANs may use radio or infrared frequencies [46].

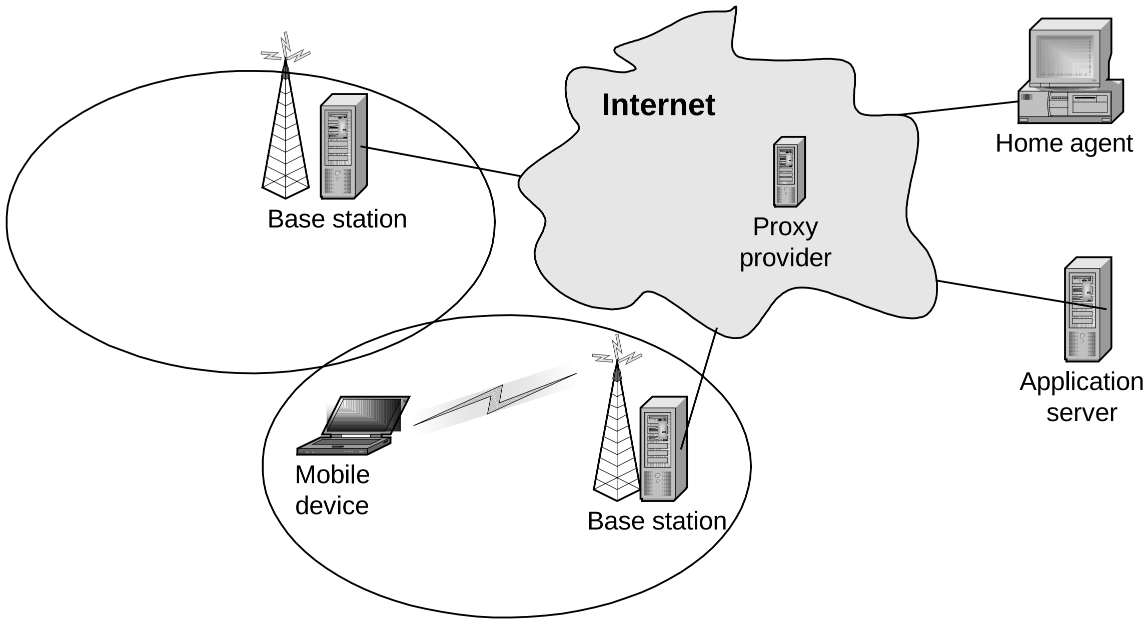

Figure 1. The wireless device connection to the Internet

The area over which a base station can transmit and receive signals is known as a cell. A mobile user may move out of the coverage area of one base station and into another cell. This situation, when a mobile unit moves to a new base station, is known as a hand-off. In this way, a mobile computer can maintain a connection despite a change in location.

The convenience of wireless computing comes at a price: there are a number of restrictions that limit mobile computing. Mobile constraints can be broken into two classes: constraints on the portable computer and constraints on the wireless network. Portable computers have more modest resources than stationary machines. For example, they have less disk space, limited memory, a smaller screen, and a finite energy source [20]. See Table 1 for a comparison between current typical consumer desktop computers and typical consumer wireless devices.

Device	**Intel PC**	**HP Jornada 820**	**Palm III**
CPU Speed	450 MHz	133 MHz	16 MHz
RAM or Program memory	64 MB	8 MB	96 KB
Hard disk or Storage memory	10 GB	8 MB	2-8 MB
Wired Network	100 Mbps	57.6 Kbps-10 Mbps	57.6 Kbps
Wireless Network	11 Mbps	1.2-14.4 Kbps	1.2-14.4 Kbps
Display	17” monitor, 1280x1024 pixels and 2^{32} colors	8.2” LCD 640 x 480 pixels 256 colors	3.25” LCD 160x160 pixels 4 grayscales
Input	104 Key keyboard, two-button mouse	touch-typeable keyboard with embedded numeric keypad	6 buttons, stylus for hand-printed input

Table 1. Comparison between consumer PC and wireless devices

A wireless network uses radio waves or infrared light to carry information. This signal can easily be blocked by buildings, corrupted by noise, and attenuated by distance. This leads to high error rates and frequent, sudden disconnections. Disconnections may cause a networked application to behave unpredictably, possibly causing it to stop running suddenly.

Furthermore, wireless networks have considerably less bandwidth than wired networks. For example, CDPD, a wide-area cellular data network, has a nominal rate of 19.2 kbps. This is substantially slower than conventional wired LANs, which can transfer data at rates between 10 and 100 Mbps [46].

Hand-offs can also cause communication problems. When a user changes base stations, information must be transferred to the new cell (for example, authentication information). Communication resources, such as an unused frequency, must also be obtained for the new user, which can be difficult if a cell contains many users. Thus, a hand-off can result in delays, short disconnections, high error rates, or even complete disconnection.

These limitations have a noticeable impact on client-server applications. Network speeds are slower, processing takes longer, and applications must use less memory than in traditional computing environments. Limited display technologies mean that graphics must be converted to smaller dimensions or fewer colors. Processing of complex structured documents leads to a noticeable increase in latency, as does transmitting or receiving large amounts of data of which only a portion is important to the user. Unless an application developer takes these limitations into account when designing an application and the protocols that it uses, the result will be slow or unusable, and use excessive amounts of the device's limited memory and battery power.

There is much research that tries to address these limitations in client-server applications [28, 33, 37, 38, 46]. One approach is to insert a new entity into the client-server model of computation.

The client-proxy-server model of communication has been used quite successfully in the past to support a variety of applications [28, 31, 34]. Instead of transmitting directly to a mobile device, a server can communicate with its proxy, which is running in the wired Internet and, if possible, close to the base station. This proxy can alter communication protocols and streams, and make the necessary adjustments to improve communication between the wired server and the wireless client. This whole scheme would be transparent from the server’s point of view, which would be unable to distinguish between a wired and a wireless client.

Comma, a proxy developed at the University of Waterloo, enables adaptive applications by providing methods for execution environment monitoring, protocol manipulation, and communication stream manipulation [34]. In the following section a more detailed description of Comma is given.

2.2 Comma, A Communication Manager For Mobile Applications

Comma consists of a monitoring module, the Execution Environment Monitor (EEM), and a communication stream modifier, the Service Proxy (SP). Because the SP was still under development, we chose to implement our own Stream Filtering Proxy (SFP) instead. The SFP provides a mechanism for manipulating, or filtering, data streams to or from a mobile host. SFP is capable of setting up concurrent connections between clients and servers, and activates and deactivate filters for each client-server connection. The EEM helps Adaptive Client Applications monitor the environment to alter computation and communication behavior accordingly.

Figure 2 gives an overall view of the architecture of the EEM. We will discuss the EEM architecture generally below.

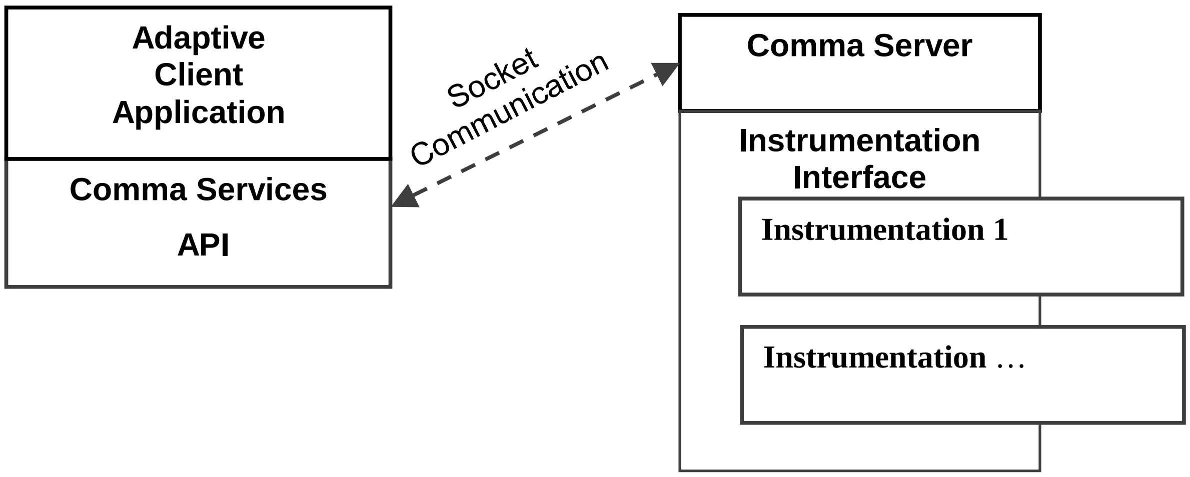

Figure 2. The EEM Architecture

The purpose of the EEM is to provide information to an adaptive client application about its environment. To benefit from the EEM, the client must instantiate a Comma Services API.

The Comma Server obtains information from instrumentation objects plugged into its Instrumentation Database. These objects share the same interface; they differ from one another in the set of Comma variables they provide. The instrumentation object does the actual measurement of the variables we are interested in. The Comma Server manages the data that is then provided to the adaptive client applications.

The client can get a Comma variable value on a one-time basis, or register interest in the variable to receive regular updates

When a client registers interest in a Comma variable, it specifies the range of values it is interested in and how often values should be reported [34].

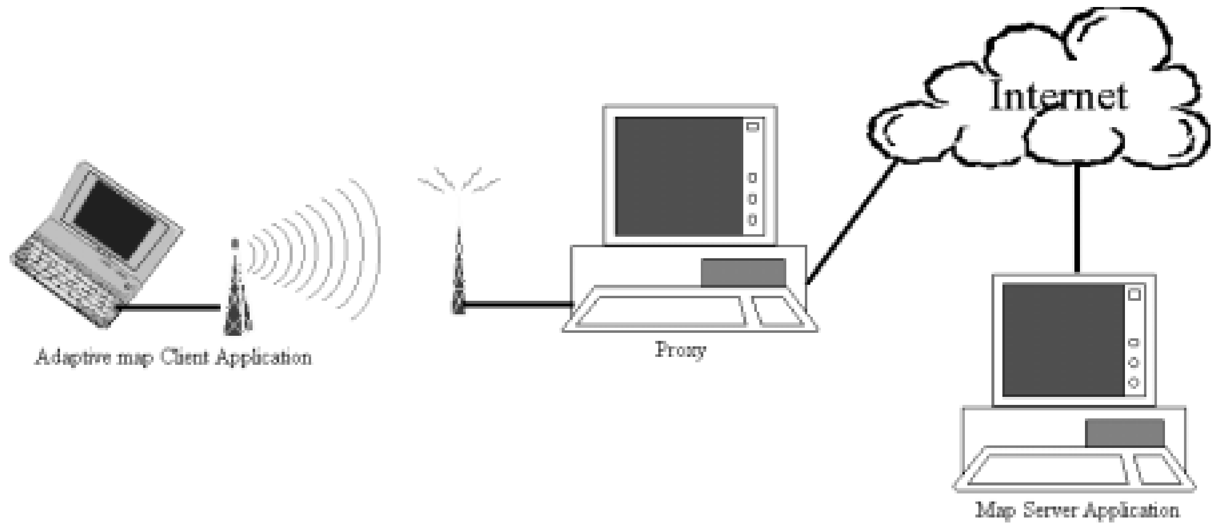

Figure 3. Client-proxy-server model

In our research, we propose that the client-proxy-server model be used by adaptive map client applications running on mobile devices requesting vector spatial data for their operation as shown in Figure 3. In the following section, we give the necessary background about geographic information systems (GIS) and the differences between the vector and raster models for representing spatial data.

2.3 Mapping Applications

Maps have been in use for thousands of years. While some people need maps to navigate the high seas, most of us need them to find our way across the country or across town. Although traditional paper maps did the job reasonably well for centuries, digital maps have a lot more to offer. Capabilities such as zooming and searching are some of the ways digital maps are superior to paper maps. The fact that digital maps do not wear out or need the same physical space adds to their appeal. All types of map users can enjoy these advantages of digital maps. This includes field workers who need maps to find routes to customer locations and users of car navigational systems.

There are two main alternatives when accessing digital maps. One is to have the map stored on a storage medium such as a hard disk or a CD-ROM. The other alternative is to have a connection to a remote server that can be queried for the desired map(s). While the first approach has many advantages (e.g. no need to connect to a network and instant availability), the second has some advantages of its own. Instead of having to dedicate physical storage space to maps that might not be needed all the time (assuming we have all the maps we need), only the map needed immediately is downloaded and deleted afterwards. Besides, in some cases (e.g. PDAs and handheld computers that have no secondary storage devices) this is the only option. One way online maps are clearly superior to stored maps is in the case of dynamic maps. Dynamic maps reflect changes such as traffic conditions in real time, providing the user with up-to-the-minute information [2].

Cost is also a factor when comparing these two alternatives. Clearly, the frequency of use and the data quantity have an important bearing on this factor. The cost of a daily one-minute data connection can be much less than buying a CD-ROM every time a map is needed. On the other hand, a daily 100-minute data connection could involve too much waiting and cost [1].

2.3.1 Geographic Information Systems (GIS)

Geographically referenced information was first included in digitally processed computer files in the early 1960s. The data, which were usually referenced to a grid overlaying a map of some sort, were stored and processed by the computer, and the results were output as line-printer maps or plotter-drawn grid cell maps. Modern vector- and raster-based geographical information systems (GISs) have evolved from these beginnings. GIS technology is now used by thousands of public and private organizations throughout the world, for a very wide variety of applications. The field has been growing at a rate about 25 to 40% per year [32].

GISs have been defined in various ways. Some of these sound more technical than others. For example, GISs have been defined as "computerized spatial data handling systems," that "automate the manipulation and integration of spatial data files" [13]. Though not impossible to understand, this statement may be confusing to someone not familiar with the terminology.

Consequently, we have chosen two definitions that seem to say what a GIS is, what it does, and what it is used for in a few, short sentences. They are as follows:

> A system of hardware, software, and procedures designed to support the capture, management, manipulation, analysis, modeling and display of spatially-referenced data for solving complex planning and management problems [12],

and

> An information system that is designed to work with data referenced by spatial or geographic coordinates. In other words, a GIS is both a database system with specific capabilities for spatially-referenced data, as well as set of operations for working (analysis) with the data [45].

Both definitions explain that a GIS works with "spatially-referenced" data or "spatial" data. This terminology may be perfectly intuitive to some, but to fully understand what a GIS analyzes and manipulates, it is necessary to break this concept down one step further. In addition, there is another important component of GIS data that the definitions do not mention, and that is "attribute" data. We will describe both of these concepts in more detail.

2.3.1.1 Spatial Data

Spatial data are "elements that can be stored in map form." These elements correspond to a uniquely defined location on the Earth's surface. Spatial data have also been described as "any data concerning phenomena that are distributed" in two or more dimensions [12].

There are three basic components to spatial data: points, lines and polygons [45]. Points are single locations, such as a dot representing a city on a map. Lines can be isolated, within a tree-structure, or elements of a network structure. Examples are river or road systems. Polygons can be isolated, adjacent, or nested, such as state boundaries or buildings on a map.

2.3.1.2 Attribute Data

Attribute data is the description of the spatial data seen on a map [45]. For example, a map of Waterloo, Ontario will have corresponding "attribute" data that describes it such as elevation numbers, land use designations and boundary information. This information is usually kept in tabular form, managed as a normal database. If something on that map changes, for example city zoning areas, the attribute data can be modified, and the map will reflect the changes.

The key with a GIS is that this map can be combined with other "layers" of information that the computer analyzes and manipulates to create a new map.

2.3.2 How does a GIS Work?

An important concept in GIS is layering. The combination of database and mapping functions creates overlays of different information, graphic and tabular, which one can easily manipulate to produce graphic results [18]. Using computers allows us to manipulate large data sets easily and analyze them in graphic form without labor-intensive manual processing. Figure 4 from [18] shows how a GIS stores information about the world as a collection of thematic layers.

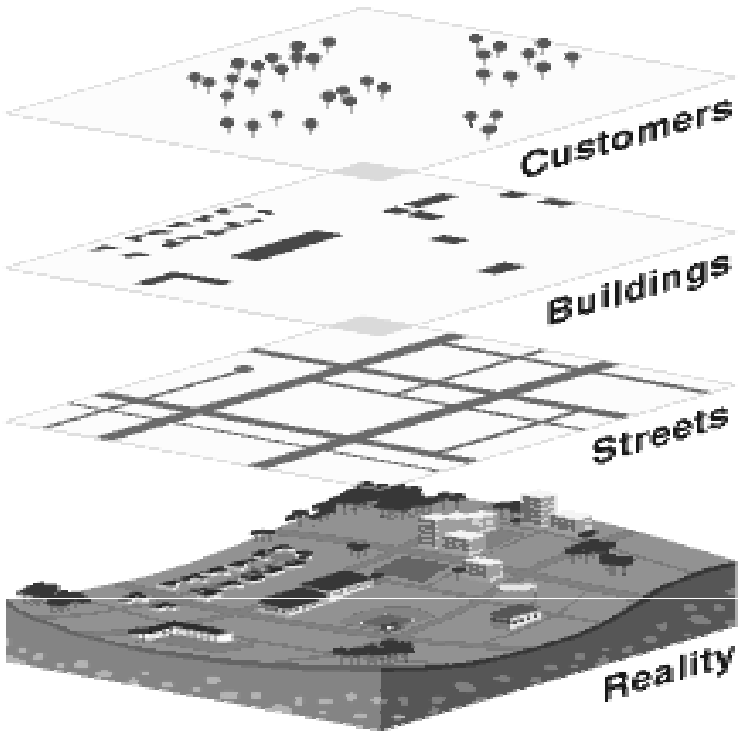

Figure 4. GIS stores information about the world as a collection of thematic layers

2.3.2.1 Geographic References

Geographic information contains either an explicit geographic reference, such as a latitude and longitude or national grid coordinate, or an implicit reference such as an address, postal code, census tract name, forest stand identifier, or road name [18]. An automated process is used to create explicit geographic references (multiple locations) from implicit references (descriptions such as addresses). These geographic references allow the user to locate features, such as a business or forest stand, and events, such as an earthquake, on the earth's surface for analysis.

2.3.2.2 Vector and Raster Models

GISs can store geographically referenced (spatial) data in a raster data structure or in an x,y coordinate (vector) reference-based data structure [18]. In the vector model, information about points, lines, and polygons is encoded and stored as a collection of x,y coordinates. The location of a point feature, such as a bore hole, can be described by a single x,y coordinate. Linear features, such as roads and rivers, can be stored as a collection of point coordinates. Polygonal features, such as sales territories and river catchments, can be stored as a closed loop of coordinates. The vector model is extremely useful for describing discrete features, but less useful for describing continuously varying features such as soil type. The raster model has evolved to model such continuous features. A raster image comprises a collection of grid cells rather like a scanned map or picture. Both the vector and raster models for storing geographic data have unique advantages and disadvantage [18]. One advantage of vector files over raster files lies with the fact that vector files are scalable; you can easily zoom in on the details in a large map, for instance [35].

Figure 5 from [18] shows a vector map that uses lines and polygons, rather than pixels, to describe an image. Vector maps allow images to be made with layers for ease of editing. Changing map elements (color, line width, type, scale) is accomplished easily. Image resolution is sharp even when the image is displayed or printed at an enlarged view. Raster images are commonly referred to as “bit mapped.” These graphics are typically created and used with programs such as Adobe Photoshop. They can be made smaller, but are not meant to be enlarged.

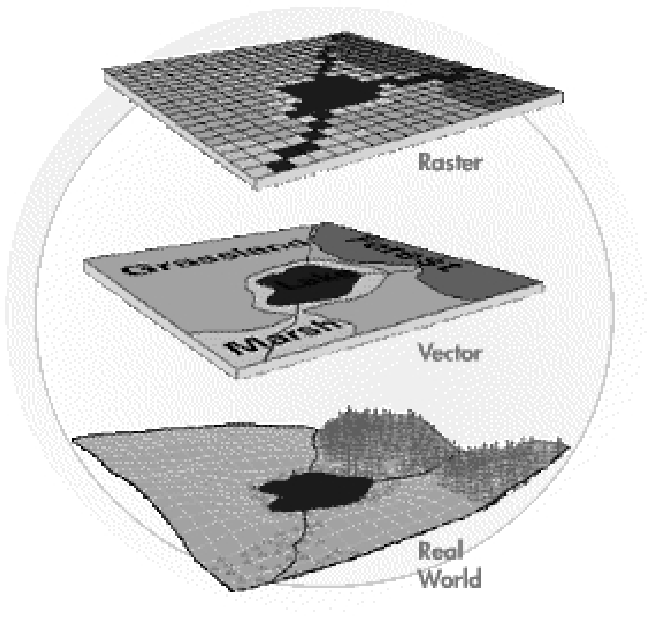

Figure 5. Vector map uses lines and polygons, rather than pixels

2.3.3 Accessing Vector Spatial Data on the Internet

Viewing and downloading digital spatial data via the Internet has become more popular. Improved access to networking technology and improved networking speeds reduce the cost of transmitting data electronically. As users expect more frequent data updates, demands for Internet access to geographic spatial information should continue to increase. At present, spatial data transmission over the Internet relies upon a raster data model [6]. The raster model proves efficient for a number of reasons. File size remains predictable for a given resolution, and constant regardless of feature complexity. Moreover, transmission of raster data can proceed incrementally by means of a simple algorithm. An initial (very coarse resolution) version consists of a subset of the raster file's rows and columns. Detail is “added” by transmitting additional rows and columns, in a randomized order. At the viewer's workstation, the Internet browser displays transmitted data with the visual effect of a blurred image whose details gradually sharpen. This method works well for digital images and satellite data products [6].

For some applications it makes sense to simply rasterize the vector image. For example, on the U.S. Geological Survey (USGS) web page (http://mapping.usgs.gov), index maps show the status and availability of digital data products. These indexes are transmitted to one's Web browser in raster format. Clicking on a specific map tile on the index sends an HTTP query back to the USGS server. By capturing outgoing HTTP requests on one's own server, it is easy to document that the query includes an argument containing a pixel value. Essentially, locations on the index map are hard-coded. The USGS server captures the index map pixel number, recovers the tile location, and responds with appropriate metadata. A similar algorithm is applied to zoom into and out of map windows on the map browser at Alexandria Digital Library at the University of California, and in the PARC (http://mapweb.parc.xerox.com/map) map viewer, a special-purpose web server running at the Xerox Palo Alto Research Center in California. For the purposes of map indexing, transmitting the rasterized version of a vector image works effectively. However, metric characteristics of distance, direction, shape, and other geographic attributes are corrupted or lost altogether in rasterizing a vector data set.

The problems for Internet transmission of vector spatial data remain a challenge. Vector files tend to be large, and file sizes tend to increase unpredictably depending on the complexity of feature geometry [6]. Users requesting vector data often require a less detailed version than the original; their first GIS operation upon receipt is to simplify or generalize the vector data [5]. It would be ideal to transmit vector data at the resolution at which a user requests it, sending first a very coarse version for browsing and subsequently amplifying details. Unfortunately, one cannot amplify details of a vector data set simply by transmitting additional rows and columns. The appearance and geometry of vector data change with resolution, and this aspect of scale-dependence must be accounted for in solving the problem of sharing vector data across the Internet.

Transmission of an entire vector data set (e.g., transportation, vegetation, …) can take hours, depending on the spatial footprint, number of features and attributes, and network speed [6]. It is possible to speed the transmission by sending compressed files, and GIS software provides export utilities to package features with attribute relational files. But even in compressed form, file size remains a function of data complexity and level of detail, and is difficult to predict. Thus the waiting time after initiating a download process is unknown.

Overcoming the size problem requires sending only as much vector data as the user needs. The principle is to transmit a very coarse version of the data, and progressively fill in the details. As stated above, since we cannot mimic the raster solution, we can use map generalization to reduce the resolution and therefore reduce the transmission data and time.

2.3.4 Map Generalization

Transformation between scales is always needed when dealing with spatial data [11]. Transformation is done by a process called generalization, which involves the selection of features to survive the scale change, the deletion of detail, and simplification of the data. Map generalization is the process of reducing detail on a map as a consequence of reducing the map scale. The process of generalization requires the selection of those features that are essential to the purpose of the map and the representation of them in a way that is clear and informative [24].

Generalization is a scale-dependent procedure. The smaller the scale, the greater the generalization required. As the scale is decreased, at some point, a scale plotting (area-proportional scaling of objects) is no longer possible. Less important objects must be eliminated or combined and more important objects may have to be exaggerated.

The two main types of generalization are semantic, based on the initial choice of the relevant information to be presented on the map, and geometric, based on manipulation of graphic characteristics of the objects.

2.3.4.1 Semantic generalization

Semantic generalization is closely related to two types of processes [24]: classification and aggregation. For both of these processes, the existence of a hierarchical data structure is very useful. Classification helps to organize objects into groups that are represented by the same symbol. Individual objects that are grouped this way are, in other words, generalized. Aggregation allows for combining objects across the hierarchical structure as the scale becomes smaller. For example, city districts may be combined together to form an urban zone.

Like geometric generalization, semantic generalization has a main objective of simplifying the presented data, so that at a given (small) scale, the complexity of the map does not make it impossible to read. Semantic generalization normally takes place before geometric generalization.

2.3.4.2 Geometric generalization

Geometric generalization is a graphic extension of classified and aggregated objects on the map. The complexity of such objects may be still too great to show them clearly, especially as the scale of the map becomes smaller. When conducting geometric generalization, the objective is to create a map that will have good visual communication characteristics. The types of symbols and the level of geometric generalization should preserve important parts of the data and eliminate or simplify the less important ones.

There are several procedures within geometric generalization: elimination of point, line and area geometry; reduction in the detail of lines, areas and surfaces; enhancement of the appearance of lines, areas, and surfaces; amalgamation of lines and of areas; collapse of areas to lines and points; enlargement or exaggeration of area and line objects; typification of line, area and surface objects; and displacement of points, lines and areas. See Figure 6 from [24] for an illustration of geometric generalization procedures.

Due to time constraints we had to choose two techniques to examine and implement, elimination and reduction, which are described in more detail below.

To decide the techniques we are interested in we had two criteria in mind. First, the implementation of the technique has to be simple enough to lower overhead required by using a filter that implements the technique. Second, the technique’s main strength has to be its ability to reduce the amount of data to be transmitted rather than to enhance the appearance of the map. Applying these two criteria to the set of geometric generalization operations in Figure 6, we can see how many of these techniques could be disqualified.

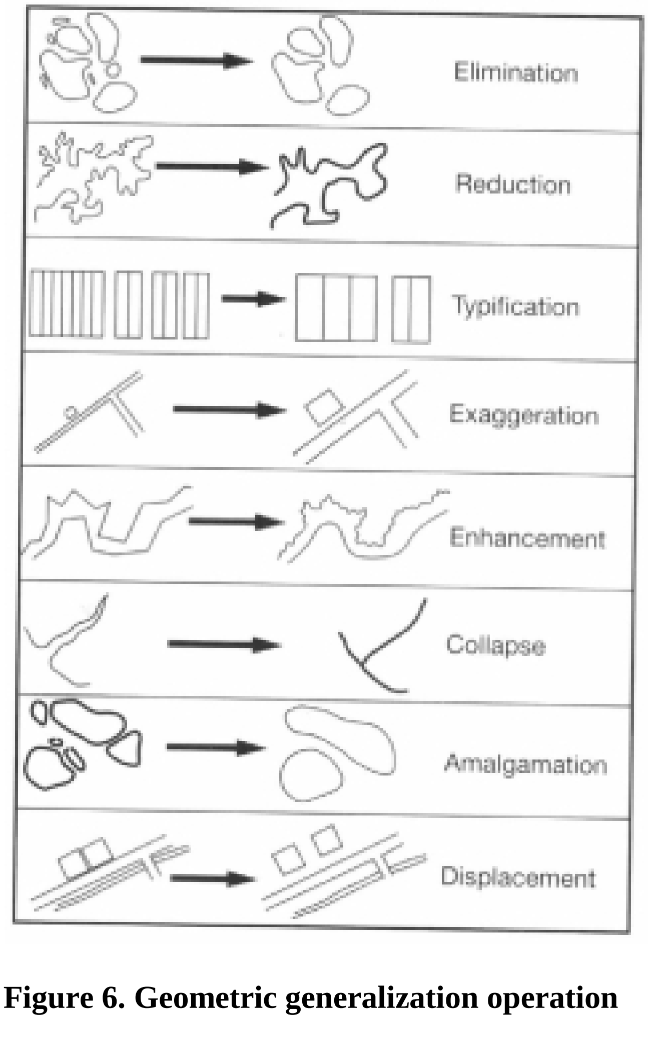

Figure 6. Geometric generalization operation

For example, Exaggeration and Displacement seem to be aimed more at enhancing the appearance of the map than doing any meaningful reduction in the size of the data. Enhancement is obviously doing exactly the opposite of what we want. While Collapse and Amalgamation seem very promising, they are very complicated to implement, which means a high overhead that could actually defeat the purpose of using filters to increase the QoS. This leaves us with Elimination and Reduction that satisfy both of our criteria.

Elimination

Elimination consists of dropping map objects as the scale decreases. If the scale of the required map is expected to result in poor visualization, then individual objects might be eliminated if this would enhance the visualization. Normally, the small objects are the ones that are removed. For areal features, elimination depends upon a combination of falling below a given area and a given maximum dimension. For example, river tributaries shorter than a given length and lakes smaller than a given area might be eliminated [24].

Reduction

The process of reduction is most common for linear and areal objects. Several methods can be used in line reduction. The following are the most common reductions in GIS applications.

Arbitrary Point Selection

To reduce the line detail, only every n^{th} node (or vertex) is preserved, and all others are removed. This method, though very simple and efficient, does not take into account the relative importance of points, which may result in cartographically erroneous outcomes.

Local Direction and Distance Processing

This includes several different techniques. One is distance-dependent, where only points lying above a certain distance from one another along the line are maintained, so if two neighboring points are closer than this threshold value, one of them is removed. A second is angle-dependent, where if the angle at the point where the line changes direction is lower than a tolerance value, the point is removed. Lastly, a mixed approach combines the distance and angle techniques.

Local Tolerance Band Processing

In its basic form, a band is created around the line beginning at the first point. The band has the direction of the line between the first and second points. When this band is crossed by one of the subsequent line segments, a second band is created in the direction of the line between the last point within the first band and the first point outside that band. The process is repeated along the entire line. All points that lie between the first and the last point in each band are removed. As shown in Figure 7, the large dots represent points selected using local band processing in the manner of the Reumann and Witkam algorithm [24].

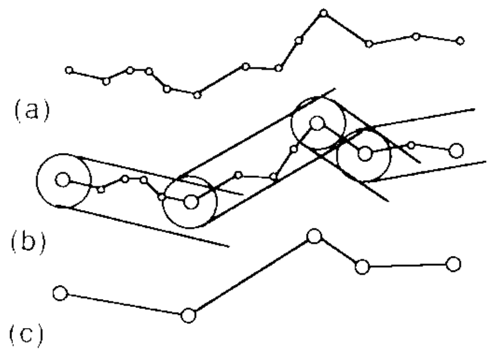

Figure 7. Line reduction using local-tolerance-band processing

Global Band Processing

The most common global algorithm is that of Douglas and Peucker [15]. In the linear object that needs to be reduced, a line is created between its starting (anchor) and ending (floater) points. A distance tolerance level is also defined. Perpendicular distances are calculated from the created straight line to all points of the linear object. The most distant point (from the straight line) along the linear object between the anchor and the floater, that lies further away than the tolerance value, becomes a new floater and the process is repeated until all points on the linear objects lie closer to the straight line than the tolerance value. These points are then removed. Figure 8 shows the stages in line reduction.

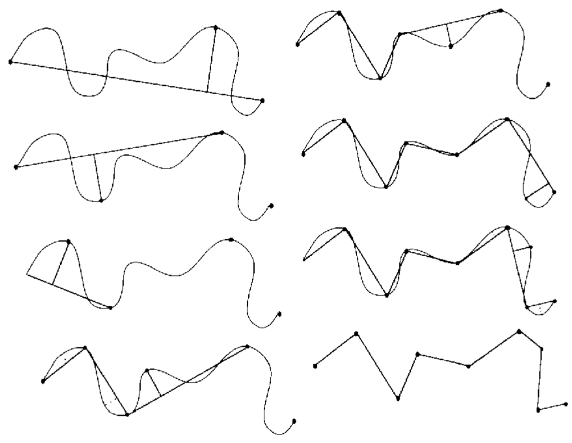

Figure 8. Stages in line reduction by the Douglas-Peucker algorithm

The accuracy and level of detail of stored geometric data can vary greatly, from large-scale plans of the boundaries of properties to small-scale, highly generalized representations of entire countries. For some purposes, the ideal GIS database might be one that could store data at the highest level of accuracy available, but enable users to access the data at different levels of detail and generalization according to the required purpose [24]. However, the pragmatic solution most commonly adopted is to store several versions of the data, derived from different-scale map resources.

We are proposing in this research to insert a proxy between the map client and the server. The proxy applies different generalizing algorithms to the spatial data, without changing the server, and using spatial data stored at the highest level of accuracy available. In this way, users can access the data at different levels of detail. Generalization can be done according to the characteristics of the client requests and environment.

In this research, we implement two filters to reduce the level of detail by applying elimination and reduction to remove small objects relative to the window size and to reduce the number of points in line or polygon objects. While the idea of elimination and reduction is not new, the way we used it is. Our main contribution regarding the use of these two techniques is that we adapted them to be used on streams of data while being transmitted from the server to the client. We also conducted a series of tests to measure the effectiveness and feasibility of each of these techniques. To the best of our knowledge, these tests are the first to be made in such a context.

2.3.5 Acquisition of Spatial Data

The acquisition of spatial data in a GIS is normally motivated by a desire to use the data to solve a problem or to make decisions relating to a particular application. Many GISs include data modeling and statistical functions that may be quite specific to their application. However, there is a common function to most applications, and it is a relatively straightforward spatial analysis to find those features, or parts of features, that lie within a given region. At its most trivial, the region may simply be a rectangular window defined by four coordinates, or two opposite corners. This may be regarded as a basic spatial database query rather than an analysis. What raises it above the level of a conventional database query is that it requires the capacity to clip spatial objects consisting of line and polygon components at the boundary of the spatial window, a task that cannot be performed using a relational database query language such as SQL, without additional specialized processing [24].

We implement this functionality and allow the user to define a rectangular window by two opposite corners, which we refer to as bounds.

As mentioned above, spatial data are stored in layers. Retrieving certain features from the GIS requires logical Boolean operations (such as AND, OR, …). Boolean operators are appropriate if each feature under consideration is of equal importance. In practice, certain features may be much more important than others to the user. The relative levels of importance of the different types of data can be taken into account by attaching numeric weights to each of the layers in an overlay operation [24].

A problem that arises with weighting of the layers is the selection of appropriate weights. The process may be a subjective one in that weights are intended to reflect the preferences of the user [24]. The selection of weights can be assisted by an interactive user interface designed to solicit weights and to ensure that they become appropriately normalized. A well-established procedure for assigning weights to a set of layers is Saaty's Analytical Hierarchy Process (AHP), which builds a matrix of pair-wise comparisons (ratios) between the layers [24]. After creating the pairwise comparison matrix, the values can be normalized by calculating the principal eigenvector of the matrix. This is done by dividing each entry in a column by the sum of the ratio values in that column. This results in a matrix of values that range from 0 to 1 and which sum to 1 in each column.

In this research, a matrix is constructed at the client. Assuming there are *m* layers (e.g. roads, parks, schools, …), the matrix is constructed by eliciting, from the user, values for relative importance on a scale of 1 to 9, where 1 means that two layers are equally important, and 9 means that the second layer is absolutely more important than the first. If a layer is less important that another this is indicated by reciprocals of 1 to 9 values (i.e. 1/1 to 1/9). For consistency, the matrix should ensure that if the importance of an attribute *a* relative to *b* is *n*, then that of *b* compared to *a* should be 1/n. If the user exhibits a small amount of inconsistency in this regard, the value is modified automatically to force consistency. An example matrix is given in Table 2.

After creating matrix (a), it can be used to derive the individual normalized weights w_1 to w_I of the I layers. Given a matrix (a) of pairwise values, they can be normalized by calculating the principal eigenvector of the matrix. This can be approximated by dividing each entry in column *i* of A by the sum of the ratio values in the column. This results in a matrix of values a_{ij} that are in the range 0 to 1 and which sum to 1 in each column. Estimates of weights for each layer can then be obtained by averaging the values in each row 1 to I.

As mentioned before, the vector spatial data sets are usually large. In a scale-less and seamless map, objects can be retrieved at an arbitrary scale. As maps may contain thousands or even millions of objects, and with single geometric objects that could contain up to a thousand points, it is essential to minimize data accesses by just retrieving objects that can be displayed meaningfully and with a least number of points [8]. This argues for a need for data structures that are suited to the nature of geometric data.

(a)	Roads	Parks	Schools	Elevation
Roads	1	1/5	1/3	1/2
Parks	5	1	4	2
Schools	3	1/4	1	1
Elevation	2	1/2	1	1

a_{ij} = importance of *i* relative to *j*

(b)	Weights
Roads	0.087
Parks	0.519
Schools	0.192
Elevation	0.202

Table 2. Matrix of pairwise comparison of layers

2.3.6 Spatial Data Structures

Spatial data structures are needed to find all two and three-dimensional data objects within a given region as quickly as possible. Unfortunately, traditional indexing methods (B-trees, etc.) are only useful for one-dimensional data [30].

From the computer graphics point of view, an object is almost always a collection of polygons. For the spatial organization, the exact geometric form is of little relevance. Instead, bounding boxes are used to describe the outlines of objects. Therefore, the data structure only needs to deal with bounding boxes. Both R-trees and quadtrees [30, 42] have been found useful for this purpose.

Depending on the shape of the object, bounding boxes can be considerably larger than the objects they contain. Using geometrically more exact bounding volumes - e.g. circles, polygons would result in more exact query results. (The probability of retrieving an object that is not actually within the search region would get smaller.) At the same time, complex bounding volumes would impose a considerable computational overhead and slow down query processing. For this reason, bounding boxes aligned on the axes are used in most performance-critical applications [30].

2.3.6.1 R-trees

R-trees consist of overlapping rectangles that either contain geometrical objects or R-tree rectangles of the next deeper level [23]. Real-world objects can be organized easily using R-trees. Many 2D GIS applications rely on R-trees or on one of their variants, and many object-oriented or object relational databases offer R-tree extensions (e.g. O2, ObjectStore, Illustra/Informix) [30].

Figure 9 symbolizes a street with houses (A-L) on either side. A geometric object is represented by its minimum-bounding rectangle (MBR). The R-tree is used to provide a spatial organization of the rectangular bounding boxes (not shown) of the houses. The first level of the tree embraces the entire scene and contains links to the rectangles of the next level. Rectangles 1 to 4 already contain links to actual objects. For rectangle 2, a further splitting or subdivision into another R-tree level is indicated. Rectangles that are too large or contain too many objects are split or subdivided into smaller rectangles (in the next R-tree level). In Figure 9, objects were grouped according to the upper left point of their bounding boxes. The main innovation in the R-tree is that father nodes are allowed to overlap. This way, the R-tree can guarantee at least 50% space utilization and remain balanced. There are many splitting algorithms for how to build the tree. However the three well-known algorithms are the linear split, the quadratic split and the exponential split [26]. Their names come from their complexity; among the three, the quadratic split algorithm is the one that achieves the best trade-off between splitting time and search performance.

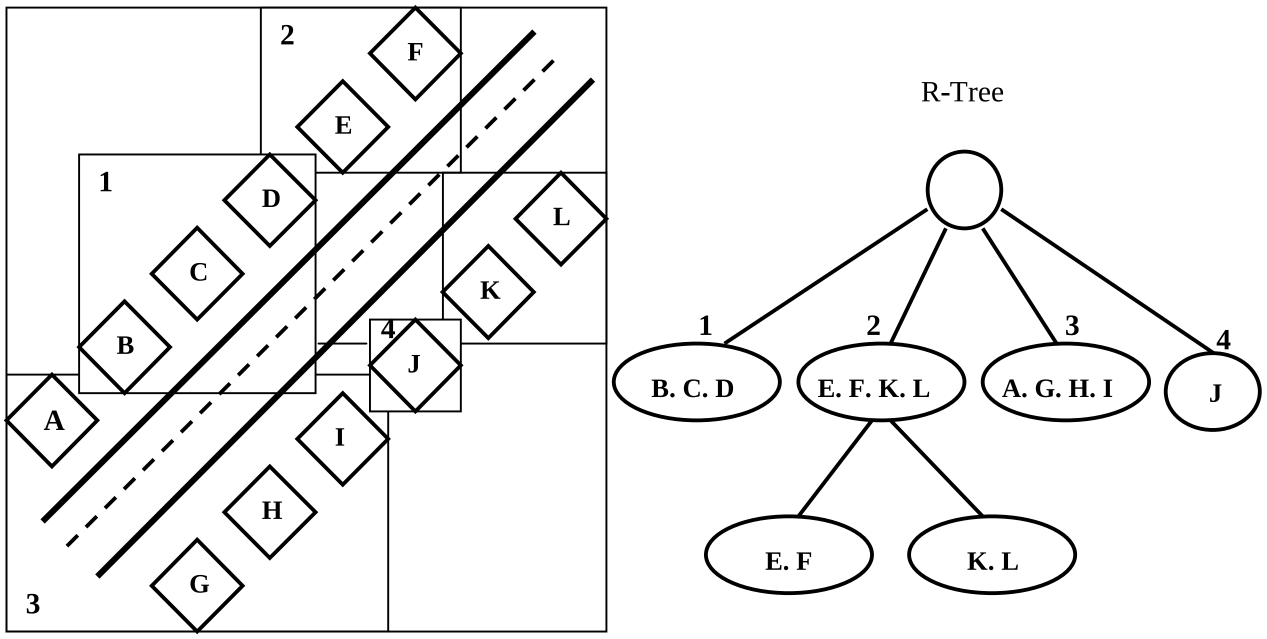

Figure 9. The concept of R-trees

A number of R-tree variants have been developed (such as R*-tree, R+-tree, and Hilbert R-tree). In this research, the map-server application maintains a Hilbert R-tree [26], one of the R-tree variants. As explained below, this is the best variant for our purposes.

2.3.6.2 Hilbert R-tree

The main idea behind Hilbert R-trees is the ability to facilitate the deferred splitting approach in R-trees. This is done by imposing an ordering on the R-tree nodes. This ordering is achieved by using a Hilbert space-filling curve to get the Hilbert value of each rectangle (object). This value is then used as the primary key when inserting the object into the tree [26].

A space-filling curve visits all the points in a k-dimensional grid exactly once and never crosses itself. It was shown experimentally that the Hilbert curve achieves the best clustering of several methods [26]. Figure 10 also shows the Hilbert curves of order 1, 2,and 3. The basic Hilbert curve is denoted by H_1. Curve H_1 has four vertices at the center of each quarter of the unit square. Curve H_2 has 16 vertices each at the center of a sixteenth of the unit square. To derive a curve of order i+1, connect four copies of H_i, each rotated as necessary.

The basic pattern is a curve which starts near the bottom left corner of a box and terminates near the bottom right corner (see Figure 10). When the order of the curve tends to infinity, the resulting curve is a fractal, with a fractal dimension of 2. The Hilbert curve can be generalized for higher dimensionalities.

To use a Hilbert curve for spatial data, consider the curve to have its lower-left vertex at the origin of a grid. The path of a space-filling curve imposes a linear ordering on the grid points. Figure 10 shows one such ordering for a 4 X 4 grid (see curve H_2). For example the point (0,0) on the H_2 curve has a Hilbert value of 0, while the point (1,1) has a Hilbert value of 2. The Hilbert value of a rectangle is defined as the Hilbert value closest to its center.

The advantage of using Hilbert's space-filling curve instead of other types of filling curves is that it groups similar data rectangles together, which leads to minimizing the area and perimeter of the resulting Minimum Bounding Rectangles (MBRs), which in turn leads to faster searching. Another reason for using Hilbert R-trees as opposed to regular R-trees or R*-trees is that Hilbert R-trees can achieve utilization levels close to 100% by adjusting the split policy used (2-to-3 or 3-to-4 …etc.). The R*-tree typical utilization is about 70% [26].

Each object is represented as a leaf node on the tree using its Hilbert value and its MBR coordinates. Each non-leaf node has the MBR coordinates that enclose its children. A non-leaf node also has the largest Hilbert value (LHV) among the data rectangles enclosed by the MBR (its children). The LHV is used to decide where a new object will be inserted in the tree.

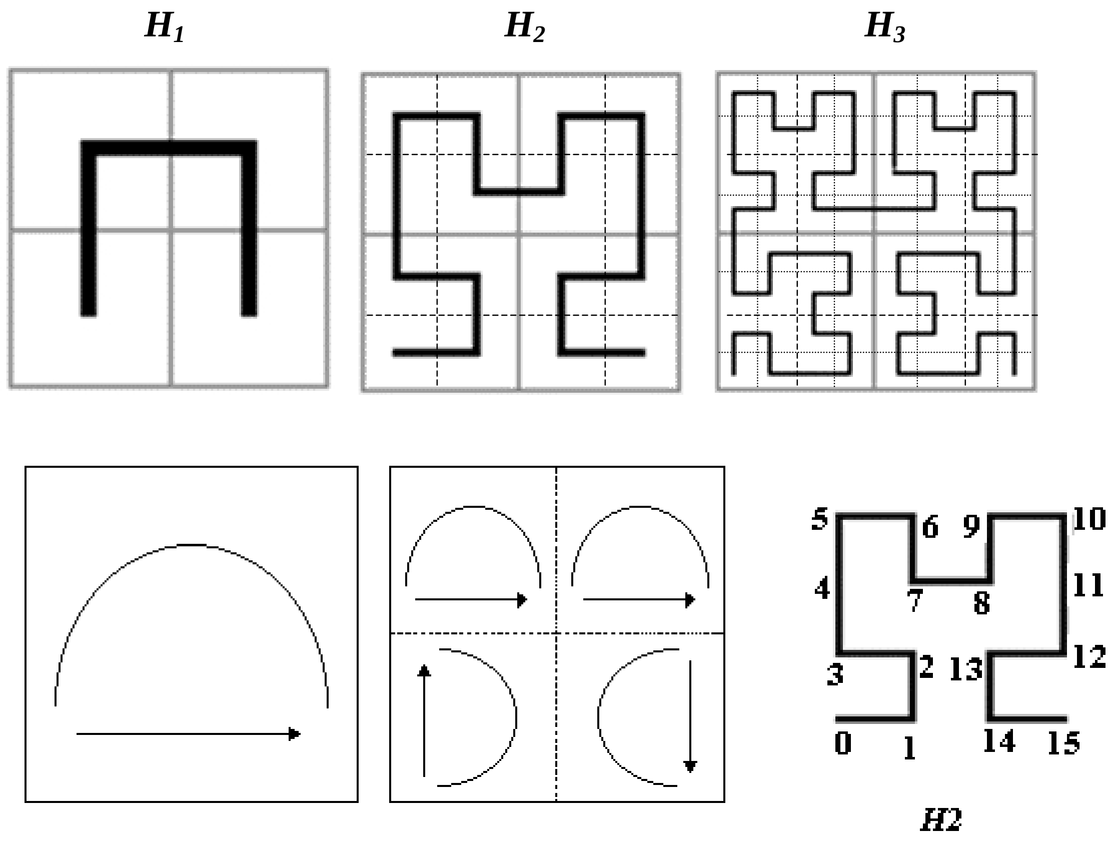

Figure 10. Hilbert curves

The Hilbert value of an object is used as the primary key when inserting objects, and the MBR value is used when searching for objects. Figure 11 illustrates some rectangles, organized in a Hilbert R-tree. The Hilbert values of the centers are the numbers by the ‘x’ symbols (shown only for the parent node ‘II’). The LHVs are in brackets. Figure 12 shows how the tree of Figure 11 is stored on the disk; the contents of the parent node ‘II’ are shown in more detail. Every data rectangle in node ‘I’ has Hilbert value ≤ 33; everything in node ‘II’ has Hilbert value > 33 and ≤ 107 etc.

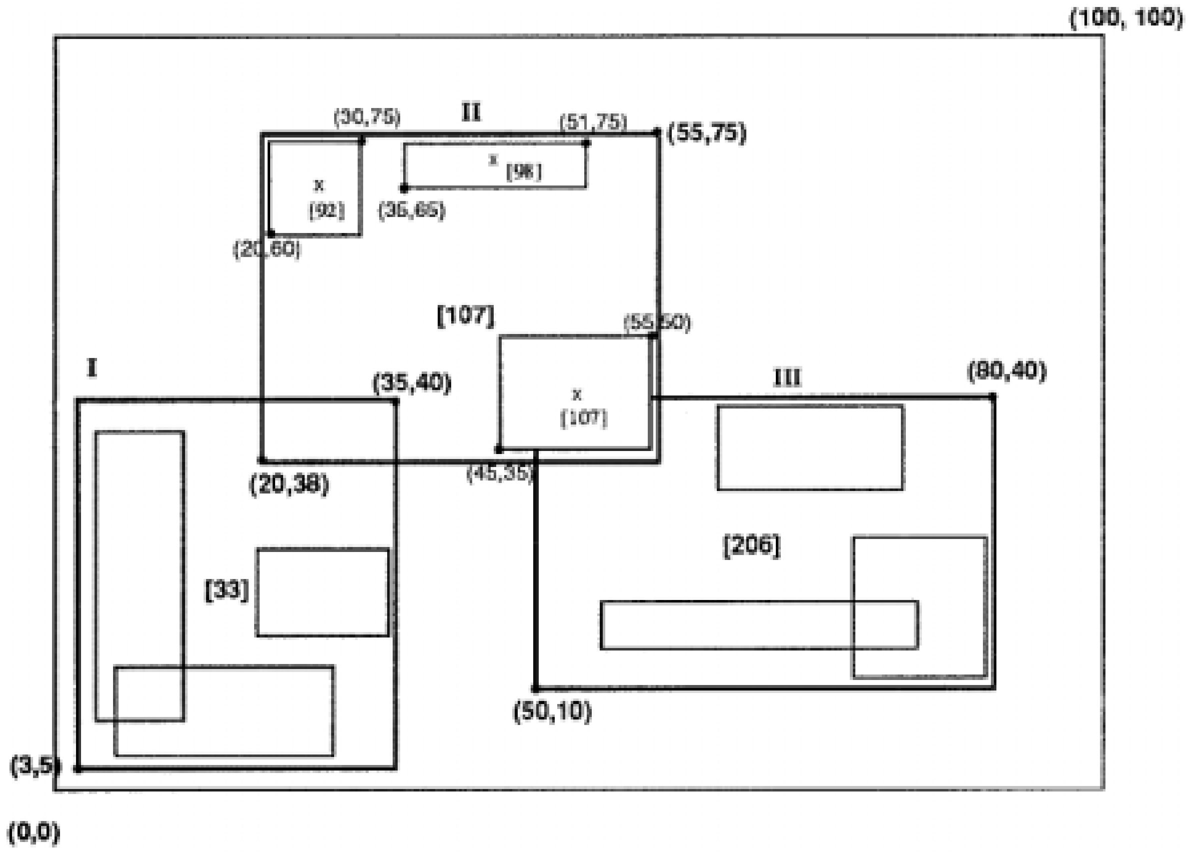

Figure 11. Data rectangles organized in a Hilbert R-tree

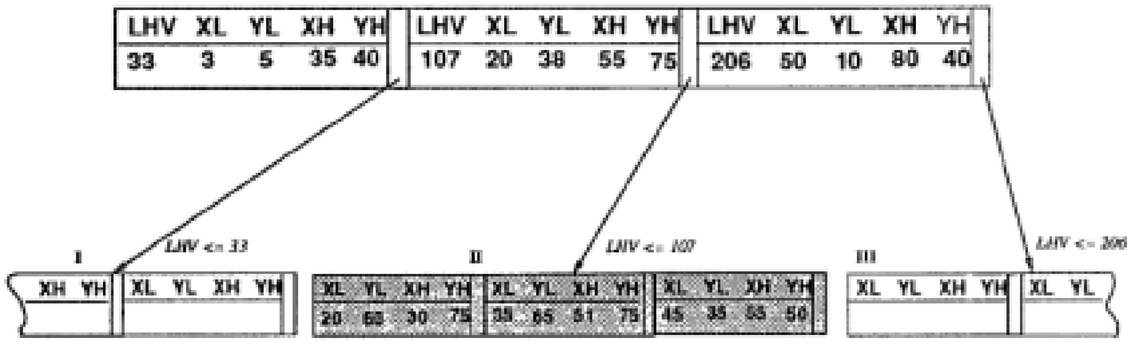

Figure 12. The file structure for the previous Hilbert R-tree

Now that background information has been provided, the proposed adaptation model for mapping applications executing in a wireless environment and detailed information about the design and implementation of the model are described in Chapter 3.

Chapter 3

Design and Implementation

3.1 Adaptation Model

Figure 13 shows the adaptation model we are using. In this model, adaptive map client applications register for environment variables with EEM. Constant monitoring of the environment variables is done by EEM, notifying the map client with the values for these variables. Whenever one or more of these variables passes a predetermined threshold, adaptive action is taken by the client to remedy the situation. The client fetches maps from a remote server via the Stream Filtering Proxy (SFP). Based on the adaptive decisions the map client makes, SFP filters the data stream by activating or deactivating filters.

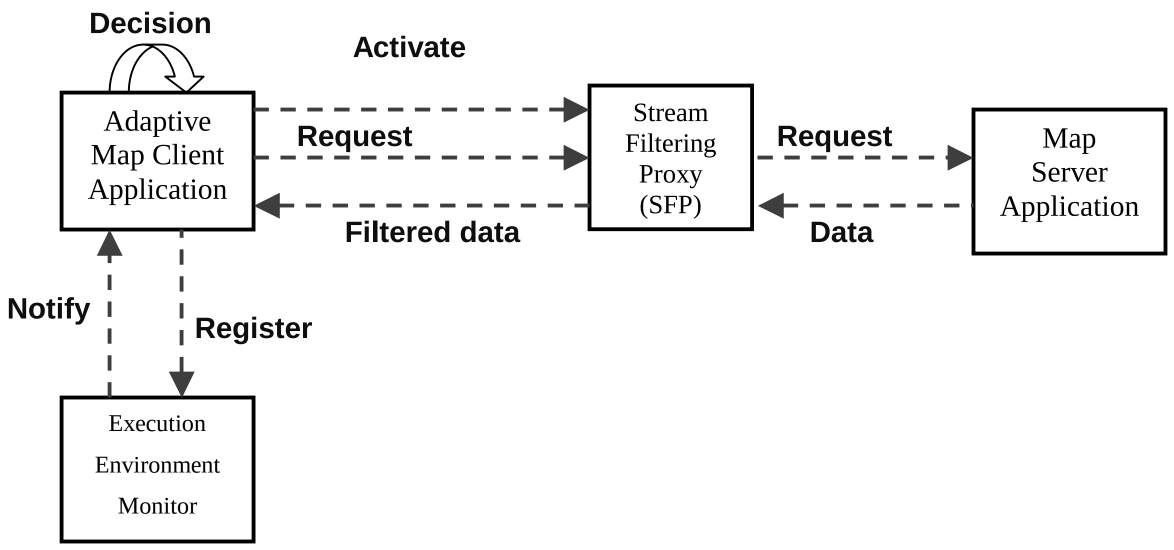
Decision
Activate
Adaptive Map Client Application
Request
Stream Filtering Proxy (SFP)
Request
Map Server Application
Filtered data
Data
Notify
Register
Execution Environment Monitor

Figure 13. **Adaptation model**

The correct adaptation decision depends on several things, not the least of which is user preference. In application-based adaptation, adaptation decisions can be handled on either the client side or the server side or both. In our adaptation model, we chose to let the client choose the kind of adaptation because each user prioritizes the objects in the map differently based on their needs. For example, buildings at crossing points of roads may be more important than other buildings in drivers' maps. That is why it is more appropriate for adaptation to take place on the client side, to allow each user to decide on the relative priority of each layer. For example, a user who is traveling across country most likely needs a map that shows the major highways (high priority) and is not interested in small roads in towns (low priority).

One important characteristic of adaptation-based systems is their ability to adapt dynamically to changes in the environment at runtime. Adaptation involves trading data quality for resource consumption. For example, a map application might fetch maps with less detail rather than suffering long transfer delays for full-quality maps. The adaptation decision is made when one or more of the performance measures passes a predetermined threshold for remedying the situation.

3.2 Design Methodology

Adaptive applications have been defined in various ways. One definition is that "An adaptive application is one in which the application changes its behavior according to the perceived constraints in the environment, so as to maintain the semantics of the application for the user" [33]. If we take this definition as our starting point, we can immediately break the problem of designing an adaptive application into two stages.

The first stage is defining the constraints in the environment, such as network quality of service or memory size. The second is defining the semantics of the application, that is, what the application is trying to achieve. This is user- and task-specific, and can vary from context to context [33].

To examine these two aspects clearly, we first need to decide on an application to be used as the subject of our experiment. We chose the mapping application for several reasons. Most of the adaptation experiments we came across were performed on multimedia applications; none were done on a mapping application. Client-server mapping applications have become very popular on PDAs these days, especially when integrated with a GPS (Global Positioning System) receiver. Lastly, mapping applications typically deal with rich data sets, which makes an excellent target for this kind of adaptation.

Our design goal was to ensure that the semantics of the application remain invariant across the adaptation process. In order to maintain the semantics, the changes in behavior of the map client should ensure that the application continues to work while it is adapting. If the behavior is changing to adapt to a restriction of some needed resource, such as bandwidth, the change should inflict the minimum damage upon the essential functionality of the application. This leads us to several design choices.

Since PDAs are typically small and not powerful, it makes sense to try to reduce the memory and processing burden on them. The way to do this is to split the functions of the system between the PDA and a stationary powerful server. The PDA is only responsible for the interface with the user, while most of the storage and processing is done on the server and communicated back to the client.

Having started with a client-server application as an adaptation example, we had three choices as to where the adaptation should take place, the client, the server, or an entity between them, namely the proxy.

While it is more suitable to take adaptation decisions on the client side for the reasons mentioned above, it may not be possible. For example, adaptation to the memory available on the client is possible on the client side, but adaptation to network conditions is not. It would be completely useless to wait until the data reaches the client to filter out unnecessary layers in the case of network congestion.

Taking adaptive action on the server is not a very attractive option either. Typically, servers are big programs that are difficult to change. We assumed that we were unable to change the server and looked for another means of network adaptation that did not require modifying the server. A third entity, the proxy, seemed like a good idea. The proxy stands as an interface between the server and the client. It is capable of filtering the data in each direction. In our case, we are only interested in filtering the data going from the server to the client to ease the burden on the network when necessary. The proxy does that by activating the proper filter(s) as instructed by the client.

When it comes to monitoring the resources, we have two options. One option is to have the client do the monitoring, since it is the party responsible for taking adaptation decisions. But since clients in our case are memory- and CPU-power-starved PDAs, having clients do the monitoring makes little sense. The other alternative was to use another program to do the resource monitoring. The readings of the desired resources are then communicated to the client periodically. EEM is the tool used for this purpose.

3.3 Architecture Overview

Figure 14 gives a more detailed view of the architecture of our system. The system consists of three parts, the client-server mapping application, the Stream Filtering Proxy (SFP), and the Execution Environment Monitor (EEM).

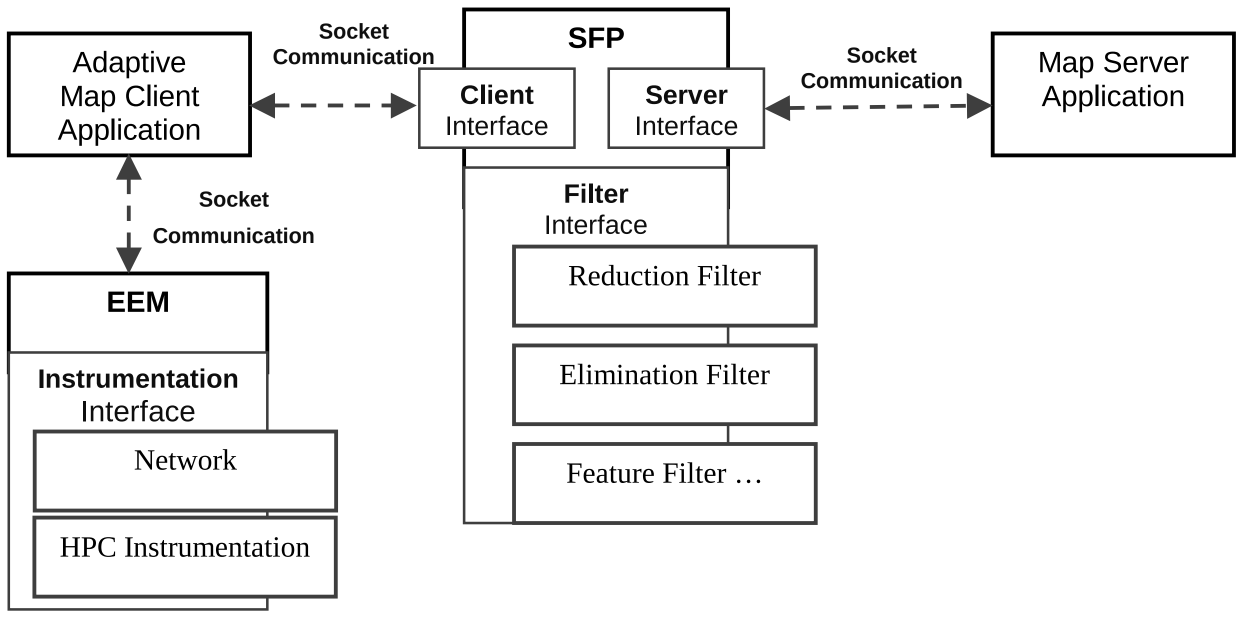

Figure 14. Architecture overview

3.3.1 The Client-Server Mapping Application

The client-server mapping application was developed in Java. The application consisted of two parts, client and server. The server is responsible for constructing a tree that contains the map objects, maintaining and traversing the tree, and retrieving the objects requested. The client is responsible for drawing the queried map objects. We will describe each part in detail.

This application was originally developed for a course project I attended a few semesters ago. While some major modifications were made to adapt it to the purposes of this effort, many of the decisions made in the original version (e.g. the use of Hilbert R-trees) still exist. I did not feel that changing these aspects of the application would be justifiable in terms of its utility to this research effort.

3.3.1.1 The Server

The Map Server (MS) maintains a Hilbert R-tree (see Section 2.3.6) for the spatial database. The tree is built from a flat file containing the coordinates of all the objects on the map. We generate this file using the Autodesk^{TM} MapGuide SDF Loader application [3] for files in Spatial Data Loader (SDL) format. SDL files are ASCII format, as follows.

```
<ObjType>, <ObjName>, <ObjKey>, <NumPts>
<LatitudeValue1>, <LongitudeValue1>
...
<LatitudeValueN>, <LongitudeValueN>
```

Where:

ObjType	Object type. **L** Polylines, **M** Points, **P** Polygons
ObjName	Object name
ObjKey	Object key, unique key that includes the layer name.
NumPts	Number of Lat-Lon points in this object’s definition.
LatitudeValue, LongitudeValue	Lat-Lon coordinate pairs (or arbitrary XY coordinates). One pair per line. Expressed in decimal degrees.

Once the tree is built, it can be searched to provide the coordinates of the objects in any particular area of the map. Objects in the tree can be points, polylines, and/or polygons. A point is defined by one x, y coordinate. Both polylines and polygons are defined by a sequence of x, y coordinates. Besides the x, y coordinate(s) of each object, the object has a layer field. The layer field specifies the layer to which this object belongs.

After constructing the tree with all the map objects, the server opens a socket and waits, listening for a client to make a connection request. The client sends the boundary of the map the user wishes. The boundary is defined with two corners of a rectangle in the map (see Section 2.3.5).

MS accepts the connection and creates a thread to deal with each client. After MS successfully establishes a connection with a client, it performs a range search in the tree and retrieves the map objects whose coordinates fall within the requested bounds. Stored map objects may lie within the range, i.e. be entirely inside, in which case they can be retrieved as a whole. Alternatively, they may overlap the range, in which case they are retrieved as a whole and sent, but only the area inside the boundary is drawn on the client window.

MS sends the objects to the client grouped in layers. The objects are sent as Java objects. After sending, the server sends a special message to the client indicating the end of the objects. The client either terminates the connection or sends a new request.

Here we would like to note that we decided to send the objects as data objects rather than ASCII text to simplify the code in the client, the server, and the proxy. We did not think that the extra time and effort spent on implementing a more efficient ASCII text-messaging format was justifiable since our adaptation model and our filtering techniques are not dependent on the format of the data sent from the server to the client. Furthermore, the sizes of the serialized Java objects are quite similar to the ASCII text file sizes, as shown by the experimental data in Chapter 4.

3.3.1.2 The Client

Map Viewer (MV), the map client, is responsible for visualization of the spatial data, giving the user the ability to select the map contents (i.e. what map, what layers, etc.) from known data sets. This means the user must have a good knowledge both of the types of map available on the server and of their layers.

MV is capable of storing all of the objects of the map locally if enough memory is available. In this case, the client can perform some zooming and panning operations without the need to query the server, resulting in savings in response time and bandwidth. The objects stored locally are organized in a Hilbert R-tree similar to the one on the server.

MV has a user-friendly map navigation system that allows the user to move around a map and zoom in and out of any part of it. Based on the user’s actions, the client queries the server for a certain area of the map. The server, in response, passes back the coordinates of the objects that reside in the area queried. MV performs the actual drawing of the map.

By using a pull-down list, users build a matrix of pairwise comparison of layers to record the relative importance of each layer. We used the AHP procedure for assigning weights. Less important layers are the ones to be dropped as the system adapts to the network bandwidth. The user has the ability to turn adaptation on and off by clicking on the *adaptation* button.

By pressing on the *connect* button, MV sends the requested bounds to the server. The initial bounds are defined when the user requests a certain map.

The user can zoom in either by clicking on the *zoom in* button or by using the mouse pointer to specify a certain area of the map. Zooming out is done by clicking on the *zoom out* button or on the *max pack* button to view the entire map. Each time the user performs a zooming or panning operation, the bounds of the drawing are changed to reflect the portion being viewed. The new bounds are used in future panning and zooming operations.

The user can query the server by using the mouse, specifying a rectangular area of the screen to zoom in. The bounds of the area selected become the drawing bounds, and are retrieved from either the tree or the server, depending on the availability of stored data locally.

The user can pan by using the scroll bars. Each time the user perform a panning operation, the MV will check if the edge of the drawing is reached, in which case a flag gets set and no more panning in this direction is permitted.

The client is responsible for taking adaptive action based on the readings it receives from EEM regarding the network throughput and the memory available at the client. To get this information, the client has to register with EEM specifying the information it needs. Based on the information provided by EEM, the client can activate filters to reduce the amount of traffic from the server to the client in case of low network throughput. It can also decide whether to build the local tree, based on the amount of memory available.

3.3.2 Stream Filtering Proxy (SFP)

Originally, we were planning to use the service proxy provided by Comma (AEGIS), but it is still under construction and there were still some stability issues to be resolved, so we decided to build a separate proxy. SFP is capable of setting up concurrent connections between clients and servers, and activates and deactivates filters for each client-server connection.

SFP will have a known port number for clients to use when they want to connect to a server. Whenever a client wants to communicate with a server, it requests a connection from the proxy indicating the port number of the server wanted. Once the proxy receives this message, it creates a new thread and sets a connection with the client. This thread then connects to the server using the port number provided by the client. The proxy will have two connections (streams) active for any client-server connection. The first is between the client and SFP; the other is between SFP and the server.

SFP is designed to receive commands to activate or deactivate filters from clients only. These commands are sent over the data connection. Until the client activates a filter, the SFP does nothing but put the messages it receives from one stream on the other in both directions. Whenever a filter is activated, all the data received from one direction is routed to the filter and its output, if any, is sent in the same direction. Filters can be activated and deactivated at any point during a connection. Filters are Java objects that can be instantiated every time a filter is activated.

Figure 15 outlines the relationships among the different components of the proxy, clients, and servers. Each client-server connection can have one or more filters activated at a time.

As mentioned above, we created a set of filters. We group them into two groups based on the effect on the data streams. The two groups are coarsening and straining. Coarsening filters reduce the level of detail in the map. Straining filters remove certain classes of features from the map.

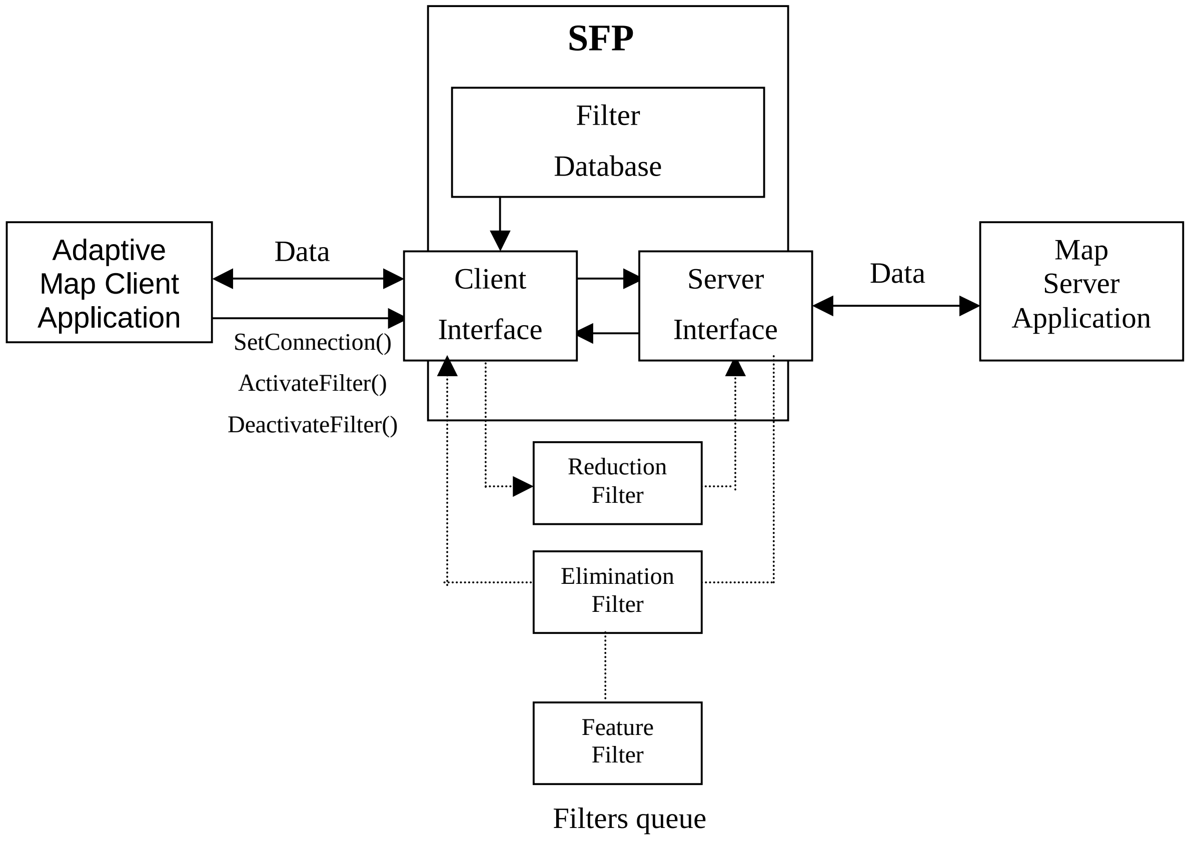

Figure 15. SFP architecture

We developed two coarsening filters for elimination and reduction based on map generalization techniques.

The Elimination Filter (EF) filters out objects that are too small to be useful. This is determined based on a comparison of the area of the bounding rectangle representing the object and the bounding rectangle representing the display. After calculating this ratio, we compare it to a threshold X. X is calculated by the following formula:

X = least number of pixels the user is interested in / the total number of pixels in the window

Any object whose area is smaller than *X* of the area of the display is discarded. *X* is calculated by the client and is sent as a parameter to the proxy when activating the EF filter. Notice that *X* will change as the user changes the minimum size of an object to be displayed (in pixels). This flexibility allows the user to determines what he/she considers to be “too small” which may vary from one user to another and from one map to another.

EF checks every message from the server and ignores the message if the object rectangle size relative to current bounds is smaller than the threshold *X*, otherwise it will send it to the client.

The Reduction Filter (RF) reduces the number of points in a line or polygon object respectively. A GIS stores a line (e.g. a lake shoreline) as a sequence of point locations, and draws it with the edges that join them. There is no limit to how many points can be stored, or how close together they may be. The amount of detail on line features should be limited, as it does not make sense to store points at intervals which are shorter than the accuracy of their locations. To lower the number of points we use the most common global line reduction algorithm by Douglas and Peucker [15] (see Section 2.3.4).

RF checks every message from the server and applies the reduction algorithm on map lines and polygons.

We developed one straining filter, Feature Filter (FF), used to filter out the less important layer(s) when the need arises. To be able to determine the relative importance (i.e. priority) of each layer, the client calculates the priority vector using Saaty’s Analytical Hierarchy Process (AHP). For the client to be able to calculate this vector, the user has to assign a value on a scale from 1 to 9 to each layer to indicate its relative importance to each of the other layers. A relative importance of 1 means that the two layers are equally important, and 9 means that the second layer is absolutely more important than the first.

After calculating the priority vector, the client sends it to the proxy when it first initiates a connection with it. Each value in the vector is a layer-priority pair. The vector is sorted in ascending order by priority. Every map has its own priority vector. When FF receives a request from the client to drop a layer, it receives as a parameter the least priority to be accepted. To determine which layer(s) are to be dropped, the FF searches the priority vector to determine the first layer with a priority higher than that it received from the client. All layers before this one in the priority vector will be dropped.

As the FF receives objects from the server, it checks the layer property of each object to determine if it belongs to one of the layers to be dropped. If so, it ignores the object. Otherwise, it passes it through to the client.

After developing the stream filtering algorithms, it is interesting to see how effective those filters are on real maps. To determine this, experiments have been devised and conducted; results are presented in the next chapter.

Chapter 4

Experimentation

In this chapter, a number of experiments were designed and executed to determine the effectiveness and feasibility of the proposed adaptation model and to evaluate the performance of the filters.

Experiments were conducted in two stages. In the first stage, tests were conducted on the performance of the filters by looking at how much data reduction was achieved on the data stream between the client and server. In the second stage, several scenarios were developed to imitate real situations and to show the effectiveness and feasibility of the proposed adaptation model.

In Section 4.1, the test data sets are described. In Section 4.2, the test results are presented and summarized.

4.1 Test Data

We used four actual street maps as our data sets in all our experiments. These maps were either provided by the Faculty of Environmental Studies of the University of Waterloo, or by a private surveying company, or have been downloaded from the Internet. Information about these maps is summarized in Table 3. Notice that "segments" in Table 3 means the edges of the graph representing a polyline or a polygon object. A smaller average number of segments per object indicate the existence of a high percentage of smaller, more fragmented objects in the map.

Map	**No. of layers**	**No. of objects**	**Ave. no. of segments per object**	**Total size of serialized object (in Kbytes)**	**Size as text file (in Kbytes)**
University of Waterloo in Waterloo, Canada [19]	9	4227	17.11	2,392	2,327
Cairo City in Egypt [7]	11	18612	12.4	8,910	7,880
Clearwater City in Florida, USA [17]	6	42782	7.13	14,354	10,027
South Tyrol City in Italy [10]	8	20436	11.95	9,066	7,985

Table 3. Test data set information

Layer Name	**Description**
FENCE	Fences
SIDEWALK	Sidewalks
BUILDING	Buildings
BRDGDECK/ BRIDGE	Bridges and bridge decks
TOWER	Tower buildings
TRAIL	Trails
DRAINAGE	Streams, creeks, and lakes.
DRIVE	Drives and private roads
ROADS	Highways, Main roads
SCHOOL	School buildings
CEMETRY	Cemetery sites and buildings
HOSPITAL	Hospital buildings
HISTRICL	Historical sites
HOTEL	Hotel buildings
RIVER / NILE / LAKES	Rivers and lakes
SHORLINE	Shoreline
RAILROAD	Railroad lines
INDUSTRE / GOVRNMNT	Government and Industrial buildings
VGTATION	Various types of vegetation

Table 4. Layer descriptions

The maps were chosen so that each map has distinct characteristics. The University of Waterloo map is relatively small and has the highest average number of segments, while the Clearwater city map is larger and has the lowest average number of segments. The Cairo and South Tyrol city maps are somewhere between the two extremes and have almost the same average number of segments. Each data set has a different number of layers, as shown in Table 4. In the following sections we will show in more detail how these distinct characteristics affect the experimental results.

4.2 Experimental Results

To assess the amount of reduction achievable by each of our filters, we first performed filter performance tests. Our second stage of testing included real-life situations to show the effectiveness and feasibility of the proposed adaptation model.

4.2.1 Filter Performance Tests

The aim of this set of tests is to assess the amount of reduction achievable by each filter. To come up with the amount of reduction achieved, we compared the size of the received serialized objects at the client machine with the original size of the serialized objects on the server side. The difference reflects the effectiveness of the filtering process.

Several experiments were performed using different combinations of filters. The sizes of the serialized objects were measured before and after activating the filters. The results and analysis of the experiments are presented and summarized in the following sections.

4.2.1.1 Feature filter experiment results

The Feature Filter (FF) is used to filter out the least important layer(s). The importance of a layer is reflected by its weight relative to the other layers. Table 5 shows the weights assigned to each layer in each of the maps used (see Section 2.3.5). We assumed that the most important layers are the roads and drives. This assumption is based on the fact that these maps are street maps and these layers are probably the most important for the user to find a route between two points.

University of Waterloo				**Cairo Map**			
#	Layer name	Serialized objects Size	Weight	#	Layer name	Serialized objects Size	Weight
1	FENCE	131820	0.012	*1*	SCHOOL	14691	0.011
2	SIDEWALK	821501	0.021	*2*	CEMETRY	253271	0.017
3	BUILDING	139442	0.037	*3*	GOVRNMNT	225072	0.023
4	BRDGDECK	1705	0.050	*4*	BUILDING	490460	0.028
5	TOWER	2728	0.070	*5*	HOSPITAL	4906714	0.045
6	TRAIL	5362	0.093	*6*	HISTRICL	175872	0.070
7	DRAINAGE	178916	0.102	*7*	HOTEL	15379	0.082
8	DRIVE	834845	0.189	*8*	TRAIL	8876	0.090
9	ROADS	333279	0.426	*9*	BRIDGE	11306	0.108
				10	NILE	6472	0.151
				11	ROAD	3016174	0.375
Clearwater Map				**S.Tyrol Map**			
#	Layer name	Serialized objects Size	Weight	#	Layer name	Serialized objects Size	Weight
1	TRAIL	22388	0.034	*1*	VGTATION	2246735	0.020
2	SHORLINE	1364047	0.084	*2*	BUILDING	3382751	0.045
3	LAKES	2092010	0.093	*3*	INDUSTRE	346656	0.050
4	RAILROAD	151174	0.102	*4*	RAILROAD	75960	0.075
5	DRIVE	3043210	0.198	*5*	RIVER	168840	0.093
6	ROAD	8026080	0.489	*6*	DRAINAGE	588283	0.102
				7	DRIVE	1801748	0.189
				8	ROAD	672546	0.426

Table 5. Information about the weight and size for each layer

Figure 16 shows the amount of data reduction that resulted from activating the FF filter. The total size of serialized objects is shown for each layer. Looking at the figures, we can see that the amount of reduction achieved using this filter varies from one map to the other based on the size of each layer in the map. For example, while dropping the first two layers in the

Clearwater map achieved less than 10% reduction, dropping the same number of layers in the S. Tyrol map achieved more than 60% reduction. Also, while dropping the first four layers in the Cairo map resulted in 65% reduction, it resulted in very little reduction in the Clearwater map.

We can also see that within the same map, the amount of reduction achieved differs from one layer to the other. For example, for the S. Tyrol map, the first two layers account for more than 60% of the reduction achieved by the FF filter. For the Cairo map, layer four accounts for more than 65% of the reduction achieved. The same is true for sixth layer of the Clearwater map.

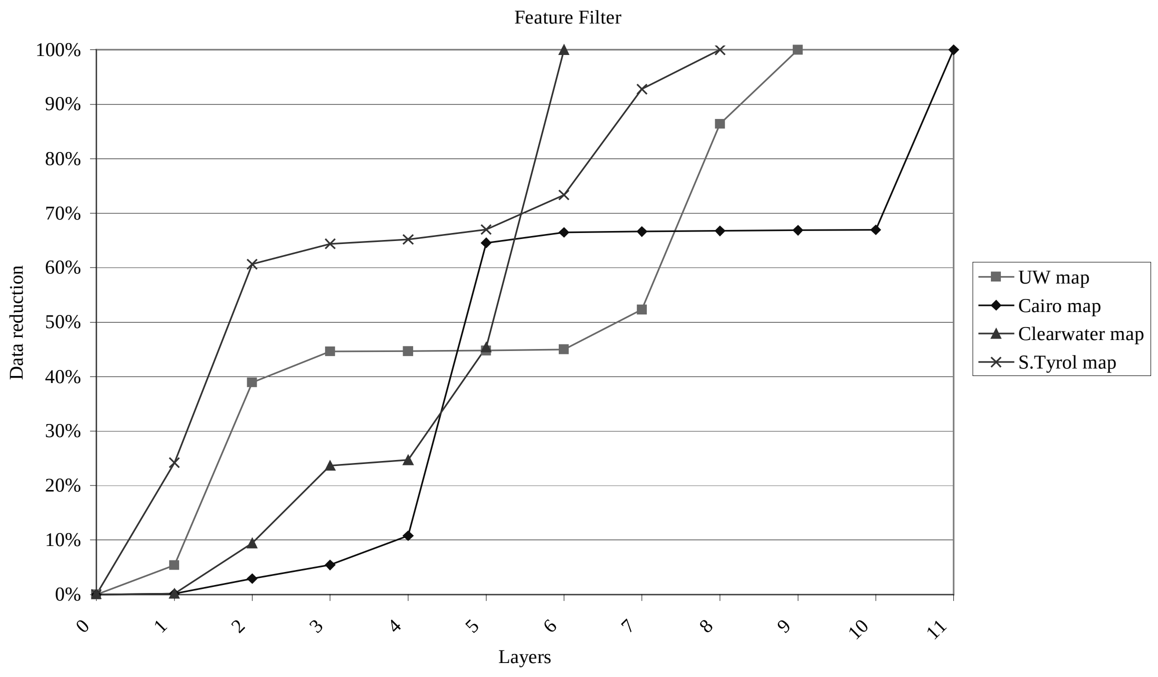

Figure 16. Feature filter experiment results

As a consequence of the previous two observations, we can see that the maximum amount of reduction possible (by dropping all layers but the most important one) differed from one map to the other. For example, while the maximum reduction achievable using the FF filter on the Clearwater map was 45%, it goes up to 93% for the S. Tyrol map.

Looking at Figure 16 again we can see that 45-65% reduction was achieved by dropping 3-4 layers or roughly 30% of the layers in the map. This observation holds true for all maps except for the Clearwater map which had a different pattern. For the rest of the maps, dropping 30% of the number of layers resulted in 45-65% reduction that corresponded to 50-90% of the maximum possible reduction. We can say that this amount of reduction is very significant which means that dropping further layers will result in much less average reduction per layer, which means that the reduction gained by dropping more layers might not be justifiable.

4.2.1.2 Elimination filter experiment results

The *Elimination Filter (EF)* filters out small objects relative to the client window size and resolution. Figure 17 shows the amount of reduction made to the maps when activating the elimination filter. For our test purposes we chose several values from one to 100 pixels for the maximum size of an object to be filtered out.

From Figure 17 we can see that except for the Clearwater map, all maps reacted somewhat similarly to this filter. The Clearwater map had an 80% reduction when objects of size one pixel were removed. This reduction could be attributed to the fact that the Clearwater map has the lowest average number of segments (see Table 7). This means that the lines and polygons are broken down into small segments, which meant that objects that should be large enough not to be removed by this filter were actually removed. Looking at the curves for the other maps we can see a similar correlation between the average number of segments and the reduction achieved using this filter. For example, the University of Waterloo map, which has the highest average number of segments, showed the lowest reduction with all object sizes tested except for one pixel. The Cairo map, with the second-highest average number of segments, achieved the second-lowest reduction. The same pattern holds for the other maps.

Looking at Figure 17, we can also notice that most of the reduction took place with small objects (up to 16 pixels). The amount of reduction diminished as the number of pixels increased to 36, 64, and 100. Actually, 70-98% of the reduction took place when 16 pixels were used as the object size. Only 1-15% extra reduction took place when the object size went from 16 to 100 pixels. This means that it might not be justifiable to drop objects of size more than 16 pixels since the reduction gained could be much less significant if compared with the loss of detail resulting from the elimination process as we can see from Figures 18, 19, and 20.

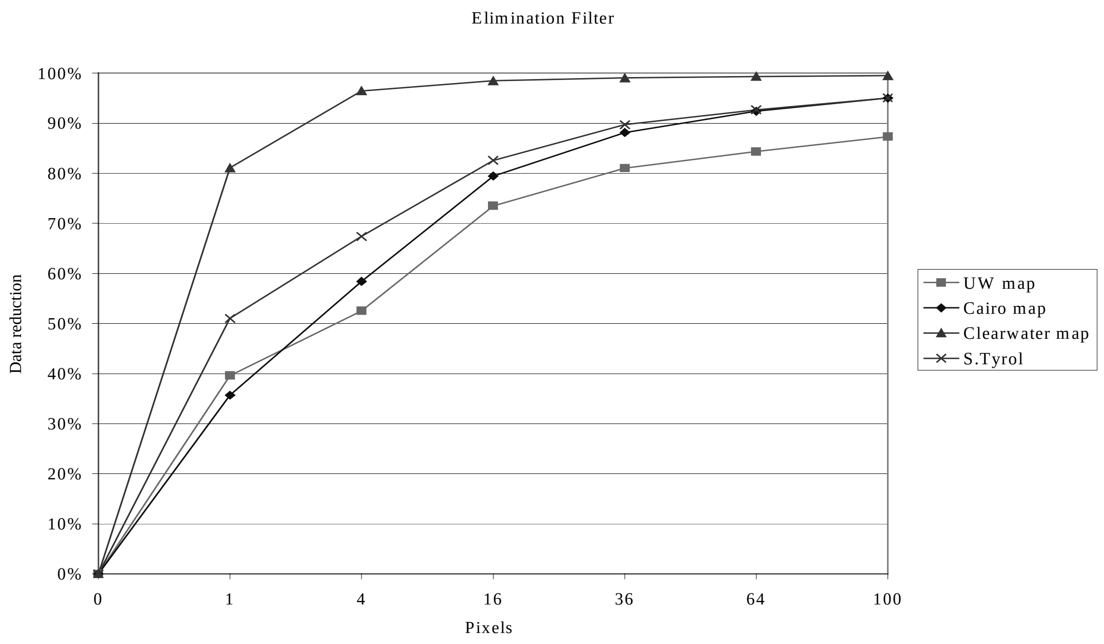

Figure 17. Elimination filter experiment results

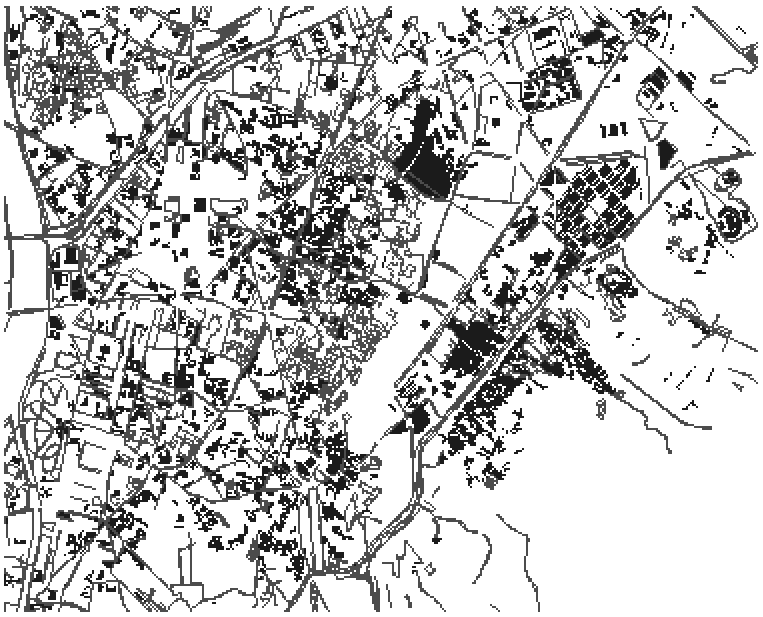

Figure 18. Cairo map, EF 16, size of serialized objects 1,829 Kbytes

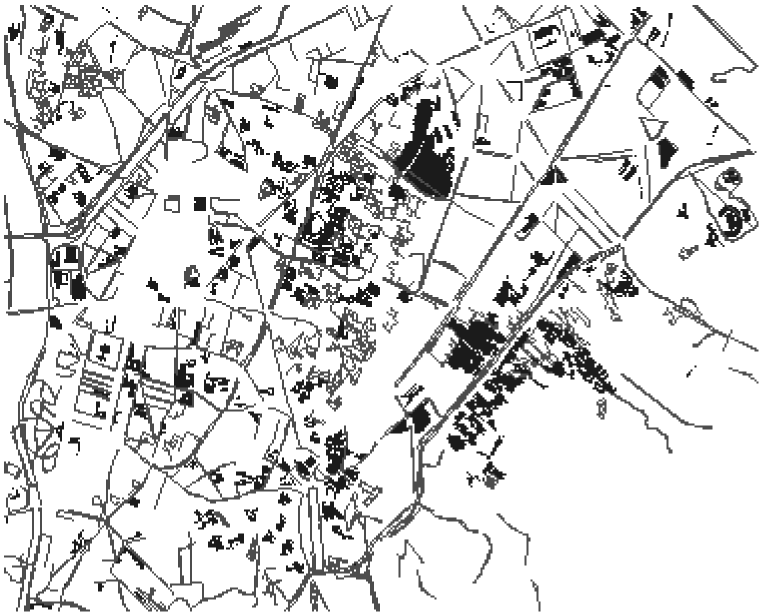

Figure 19. Cairo map, EF 36, size of serialized objects 1,056 Kbytes

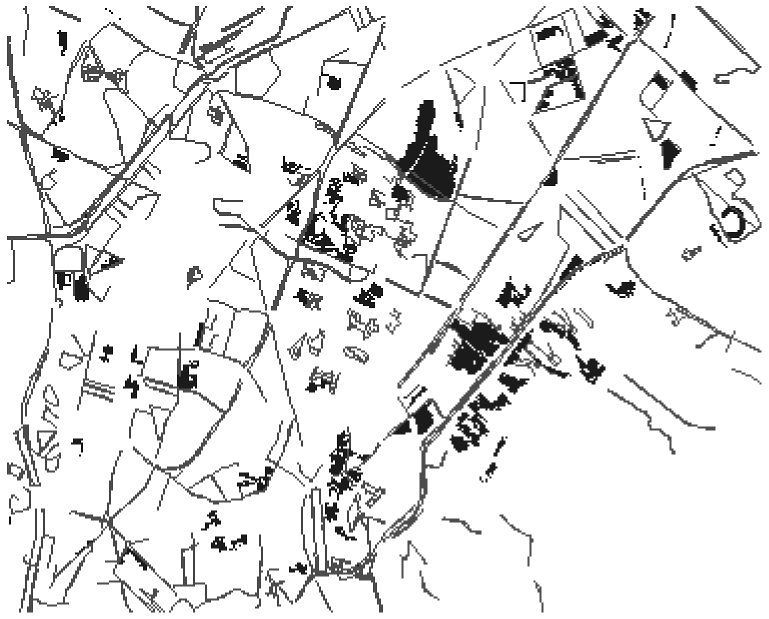

Figure 20. Cairo map, EF 64, size of serialized objects 675 Kbytes

4.2.1.3 Reduction filter experiment results

The *Reduction Filter (RF)* reduces the number of points in a line or polygon object. RF takes a target reduction percentage by which to reduce the number of points. Figure 21 shows the reduction achieved by using the filter. For our test purposes, we used 30, 70, 90, and 99% reduction percentages. (99% is the most reduction can be made on the data.) By inspecting Figure 21, we can see a strong correlation between the average number of segments per object in a map and the amount of reduction achieved using the RF filter. The University of Waterloo map, with the highest average number of segments per object, achieved the highest reduction with a maximum reduction of 57%. The Clearwater map, with the lowest average number of segments per object, achieved the lowest reduction with only 28%. The Cairo and S. Tyrol maps were similar (with about 40% maximum reduction) since their average numbers of segments per object were very close. This behavior is to be expected since this filter achieves reduction by lowering the number of segments in an object. As the number of segments in an object increases, so does the potential for reduction. This observation suggests that the amount of reduction to be expected from applying RF with a certain percentage could be approximated given the average number of segments per object in the map.

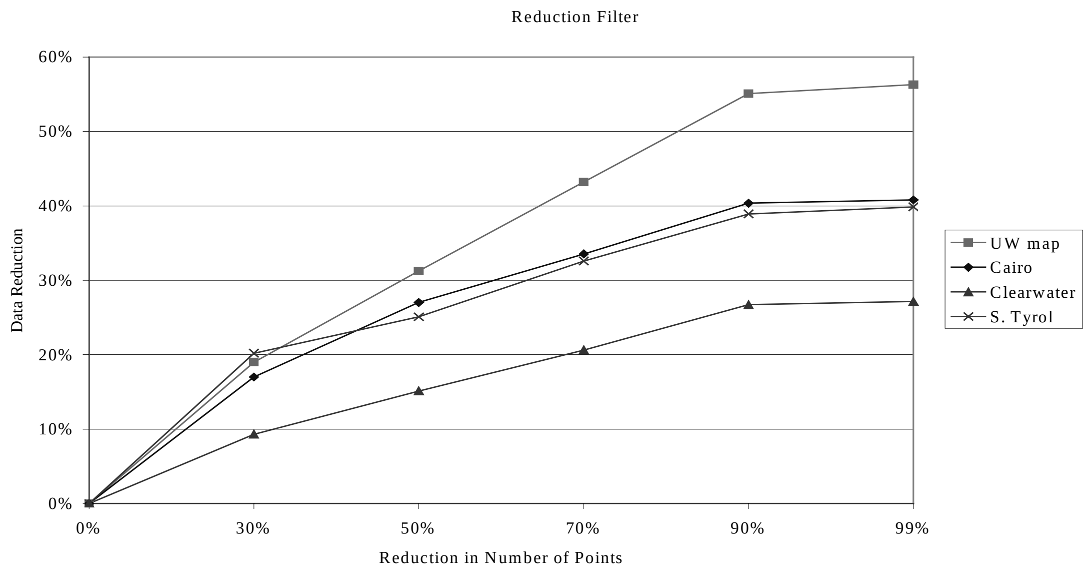

Figure 21. Reduction filter experiment results

4.2.1.4 Applying combinations of more than one filter

These tests were performed to explore the effects of combining more than one filter on data reduction.

First we attempted to see if the order of the filters has any effect on the data reduction. For example, would activating EF then RF produce the same results as RF then EF? From these tests we found that the results are almost the same. However, it makes more sense to activate EF before RF to save time, since the elimination will filter out some of the objects that the RF would otherwise spend time on.

Generally, there was extensive data reduction when running combinations of filters. Tables 6, 7 and 8 show the results of applying pairs of filters FF-RF, EF-RF, and FF-EF on our four maps. Notice that the number of layers dropped by the FF filter is not the same for all maps. That number can be determined by matching the map with the value after FF that corresponds to it. For example, in the first column in Table 6, the number of layers dropped by the FF filter for the UW map was 2 while it was 4 for the Cairo map and so on.

	FF 2,4,2,2 & RF 30%	**FF 8,8,4,5 & RF 30%**	**FF 2,4,2,2 & RF 99%**	**FF 8,8,4,5 & RF 99%**
UW Map	53%	90%	79%	95%
Cairo Map	73%	75%	85%	87%
Clearwater Map	51%	54%	63%	70%
S. Tyrol Map	77%	79%	82%	86%

Table 6. Applying FF & RF on all four maps

	EF 16 & RF 30%	**EF 64 & RF 30%**	**EF 16 & RF 99%**	**EF 64 & RF 99%**
UW Map	80%	88%	93%	97%
Cairo Map	85%	95%	95%	99%
Clearwater Map	98%	98%	99%	99%
S. Tyrol Map	88%	95%	97%	99%

Table 7. Applying EF & RF on all four maps

	FF 2,4,2,2 & EF 16	**FF 8,8,4,5 & EF 16**	**FF 2,4,2,2 & EF 64**	**FF 8,8,4,5 & EF 64**
UW Map	82%	96%	87%	97%
Cairo Map	91%	91%	95%	95%
Clearwater Map	99%	99%	99%	99%
S. Tyrol Map	92%	93%	96%	97%

Table 8. Applying FF & EF on all four maps

Notice that the number and values of parameters chosen to be used in testing with each filter were meant to give a cross section of the filter's performance while limiting the combinatorial effect of multiple values.

Looking at these figures we can make two observations. First, as expected, the amount of reduction achieved by combining two filters has a strong correlation with the amount of reduction achieved by each filter individually. For example, applying RF with 30% reduction and FF with four layers dropped achieved the lowest combined reduction for the Cairo map (about 73%). Using RF with 99% reduction and FF with eight layers dropped achieved about 87% reduction. Looking at the individual curves for these filters for the Cairo map, we can see that the individual filters behaved very similarly. The FF filter alone achieved about 65% reduction with four layers dropped and achieved about 67% reduction with eight layers dropped. Similarly, the RF filter achieved only 17% reduction with 30% reduction rate but achieved about 57% reduction with 99% reduction rate. Second, the combined reduction is usually less than the total of the individual reductions. This is to be expected since the same object could be filtered out by both filters. When filters are combined the object is counted only once.

We also notice that combining FF with EF results in more data reduction than using FF with RF. This is due to the fact that EF has a very high reduction rate, especially when the maximum object size to be eliminated is 16 pixels or more.

The question now is which combination of filters to choose. We believe that there is no single answer to this question. The combination to choose depends on the user needs and the characteristics of the map. For example, if the user needs all the layers to be visible, then the option of using the FF filter is not available. Now the question is what parameters should be used with each of the two filters to be used. Assuming that maximum reduction is desired while maintaining usability, we can see from Figures 22 and 23 that using 16 pixels with EF and 30% with RF makes for a good combination that satisfies these criteria. With an average reduction rate around 91% we believe that this combination with these parameters is very well balanced in terms of reduction and usability. If more reduction is desired, then again it depends on the user’s needs. If the user is interested more in details of polylines and polygons than the existence of smaller objects (e.g. using the map to find trails in a mountain area), then the number of pixels to be used with EF should be increased rather than increasing the percentage of reduction used with RF.

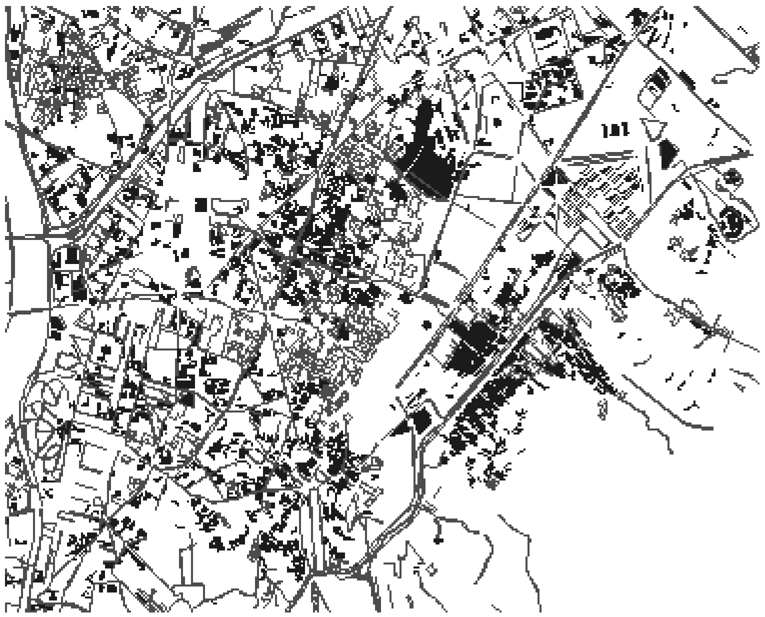

Figure 22. Cairo map, RF 30% & EF 16, size of serialized objects 1,151 Kbytes

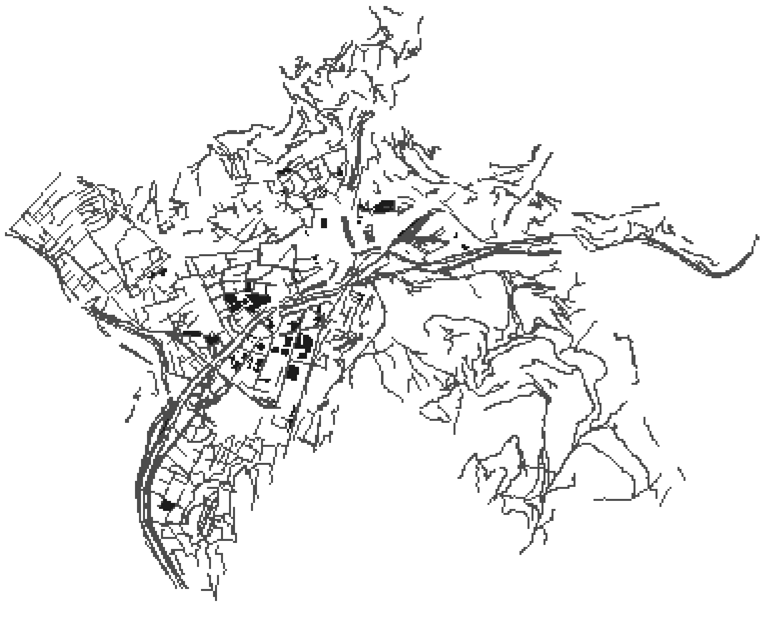

Figure 23. S. Tyrol, RF 30% & EF 16, size of serialized objects 1,071 Kbytes

Table 9 shows the results of combining the three filters. We notice that combining the three filters did not always increase the amount of reduction in a dramatic way compared to using only two filters.

	FF 2,4,2,2 & EF 16 & RF 30%	**FF 2,4,2,2 & EF 16 & RF 99%**	**FF 2,4,2,2 & EF 64 & RF 30%**	**FF 2,4,2,2 & EF 64 & RF 99%**	**FF 8,8,4,5 & EF 16 & RF 30%**	**FF 8,8,4,5 & EF 16 & RF 99%**	**FF 8,8,4,5 & EF 64 & RF 30%**	**FF 8,8,4,5 & EF 64 & RF 99%**
UW Map	86	95	91	97	96	98	98	99
Cairo Map	93	97	96	99	94	97	97	99
Clearwater Map	99	99	99	99	99	99	99	99
S. Tyrol Map	94	99	98	99	95	99	98	99

Table 9. Combining three filters on all four maps

For example, while adding the EF filter to the other two improved reduction noticeably (10-20% in the S. Tyrol map, for example), adding the FF filter to RF and EF did not result in a big improvement. This is to be expected given the high reduction achieved by RF and EF combined. Most of the objects to be filtered out by FF are probably filtered out by the other two. That is why adding FF did not make a big difference in the reduction achieved.

4.2.2 Scenarios

Several scenarios were developed to simulate real situations and to show the effectiveness and feasibility of the proposed adaptation model. We had the following concepts while designing our scenarios.

Maps are spatial media used for representing spatial, or more specifically, geographic knowledge. Considering spatial representation with respect to geographic maps in the first place means examining a representation medium such as a screen. This medium as a form of representation imposes certain constraints on the content encoded in it.

A cognitive question about visualizing maps is how much information should be presented on the screen. Access and processing time are related to the complexity and clutter on the map, and there is always tension between the goals of presenting complete information while not obscuring relevant objects with unnecessary detail. Most navigation maps are designed to err on the side of presenting complete information at the expense of accessibility because they are often the sole resource for navigation. While it may take considerable time and attention to extract needed spatial information, this is judged preferable to faster access.

Too much detail, however, can make such a display an annoying drain on the attention to interpret. That can limit its usefulness, or worse, take away needed attention from the user, such as in a car navigation system. For this reason, some kind of prediction must be made as to the appropriate map information to be displayed before it is actually rendered.

One way to overcome the “large volume” problem for transmitting vector files and viewing them on a crowded screen is to send only as much vector data as the user needs. The principle is to transmit a very coarse-resolution version of the data and gradually increase the resolution as the user requests more detail (i.e. zooms in).

One of the biggest limitations is the frequent need to keep map size small in order to facilitate downloading. This need in turn leads to the need to limit the amount of information on a map and to take great care with generalization. The content of the map depends also on its scale. In principle, maps on a screen have no fixed scale, since they can be enlarged or reduced at will by zooming in or out. This changes the relation between the displayed map distances and real-world distances, and thus the scale. There is, however, an ideal scale (or scale range) to display any particular map, depending on the density and accuracy of map detail. If a map is enlarged too much, very few details may be visible in the image window and the positional accuracy of the symbols may be much less than the user expects. If the map is reduced too much, it becomes illegible.

In the adaptation algorithm, one aspect that should be taken into account is the operational logic of activating the filters. For example, it would be logical and efficient to start with feature filtering or elimination followed by reduction filtering.

4.2.2.1 Scenario #1

The user makes an initial request for an entire map. The whole map is displayed for him to choose the area he would like to zoom in on. In this case, if we apply no filters, the map will be too cluttered to be useful. Figures 24, 25, and 26 show an overview of the Clearwater, S. Tyrol, and Cairo maps with no filters activated. We can see how the usefulness of these maps is very limited due to having too much detail in such a small area.

To overcome this problem, we suggest that one or more filters should be activated before the map is transferred from the server to the client. This will have two desirable effects. First, the user will get the map faster. Second, the map will have only enough detail for the user to decide which area he is interested in. Looking at Figure 27, for example, we can see a much clearer Cairo map with only the main roads and highways. This map is less than 7% of the size of the original map with no filters activated. This drastic reduction was achieved by activating three filters; FF with 4 layers dropped, RF with 30% reduction rate, and EF with 16 as the max object size to be dropped. Trying to reduce further will lead to a map with too little detail and this will defeat the purpose of trying to increase the clarity and usability of the map. Figure 28 is a good example of such a case where an extra 3.5% reduction could cause the map to become virtually useless.

We should note here that this particular combination of filters with these parameters is by no means an ideal combination that will work for all maps. For example, while the same combination for S. Tyrol worked reasonably well with about 94% reduction, using it with Clearwater resulted in more than a 99% reduction, rendering the map completely useless. Actually, we got no reasonable results when activating the three filters with Clearwater since the reduction achieved was always above 98%. From the previous numbers, we can say that we should aim at a reduction of about 90% for the whole map to achieve both clarity and usability at the same time.

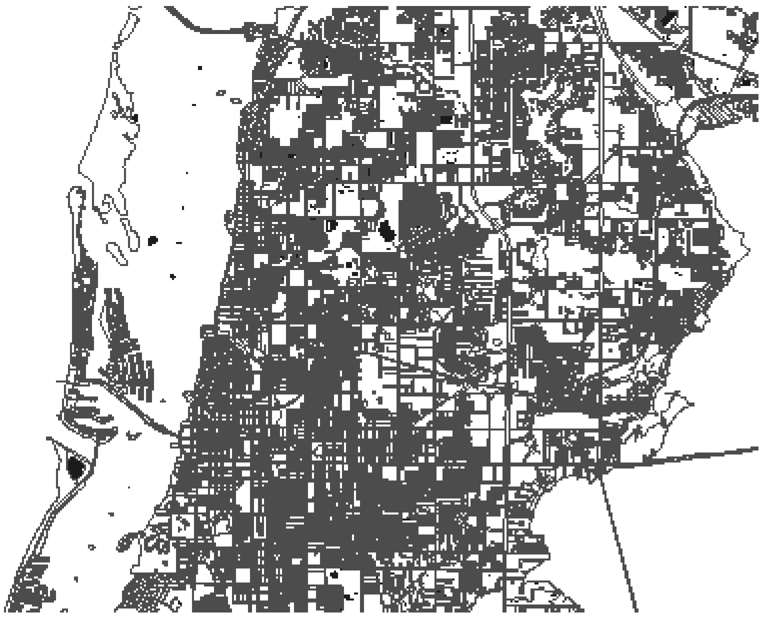

Figure 24. Clearwater map, no reduction, size of serialized objects 14,354 Kbytes

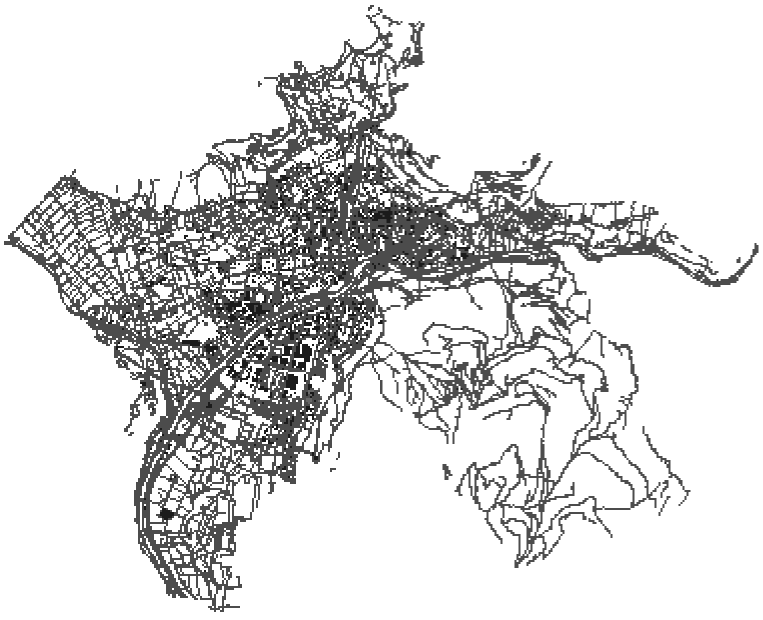

Figure 25. South Tyrol map, no reduction, size of serialized objects 9,066 Kbytes

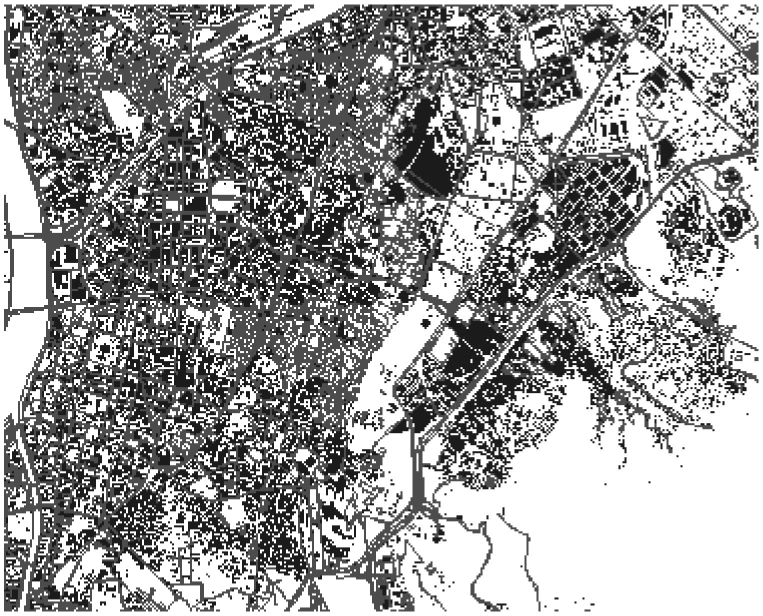

Figure 26. Cairo map, no reduction, size of serialized objects 8,910 Kbytes

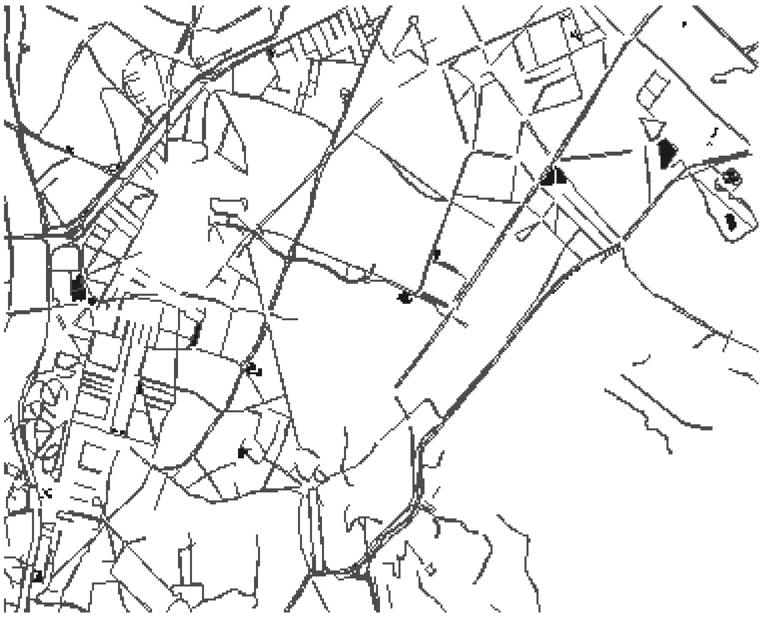

Figure 27. Cairo map, FF 4 & RF 30% & EF 16, size of serialized objects 599 Kbytes

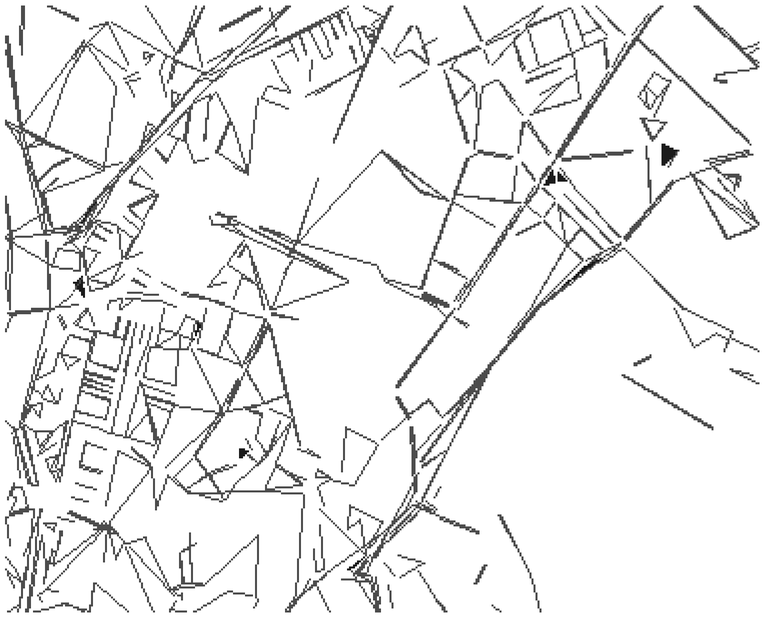

Figure 28. Cairo map, FF 8 & RF 99% & EF 16, size of serialized objects 231 Kbytes

4.2.2.2 Scenario #2

Now that the user has a clear, concise map to start with he will probably zoom in on the area of his interest. If we keep the same amount of filtering in place, the zooming process will not reveal much more detail than what is already on the screen. Zooming is supposed to allow the user to see more detail as he zooms in farther and farther. For this to happen, we should start relaxing our filtration process more and more as the user zooms in farther. This should not have too much of a negative impact on the response time since as the user zooms in, fewer objects are displayed, thus reducing the number of objects transferred from the server to the client. We can relax the filters gradually to allow the user to see more detail. The question then becomes in which order do we deactivate or relax our filters, assuming that we have more than one filter activated as in the Cairo map example above.

We believe that filters should be deactivated in the following sequence. First, the elimination filter. This will allow the user to view the objects that were invisible on the overall map like points, small polygons, etc. Second, the feature filter should be relaxed gradually as the user zooms in to show more and more of the layers that were hidden in the overall map. Finally, the reduction filter should be relaxed and eventually deactivated as the user reaches greater zoom depth. It makes sense to retain this filter to the end since it will probably not matter too much to the user how accurate the polygons and polylines are until they are large enough for the user to be able to tell the difference.

Applying this gradual deactivation of filters concept on the Cairo map in Figure 27, we can see in Figure 29 that after deactivating the elimination filter as the user zoomed in once, many more details became visible, the grid of the main roads is more complete, and some smaller streets become visible. The size of the serialized objects was under 1Mbyte, where if we transfer without activating any filter, the size will be over 3Mbytes with less visibility (see Figure 30). As the user zooms further, we can see in Figure 31 how the hidden layers become visible as the feature filter gets deactivated, allowing more details (e.g. buildings) to be added to the map. With further zooming in Figure 32, we can see how deactivating the reduction filter showed the buildings and roads with the original number of points in each object, which made them much more natural looking.

From this example we can see how deactivating filters as the user zooms in gradually gives the user more details without overcrowding the screen or inflating the file size.

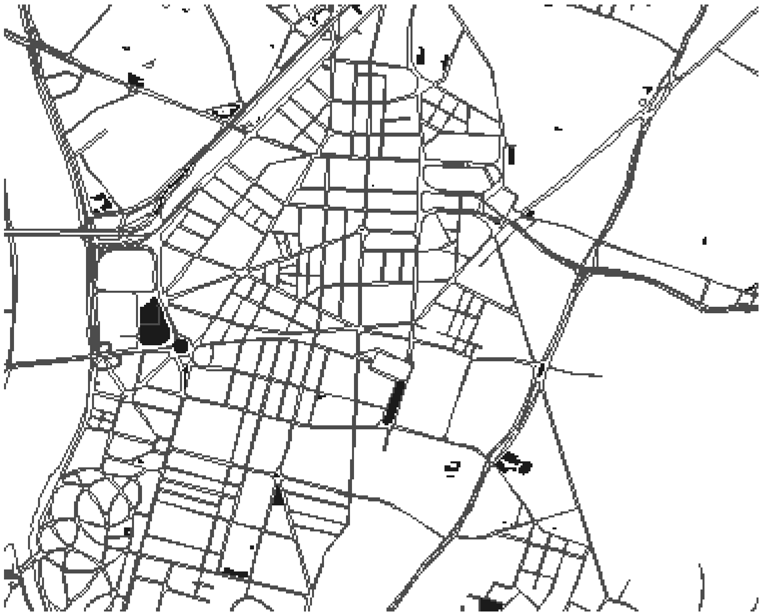

Figure 29. Zoom in Cairo map, FF 4& RF 30%, size of serialized objects 987Kbytes

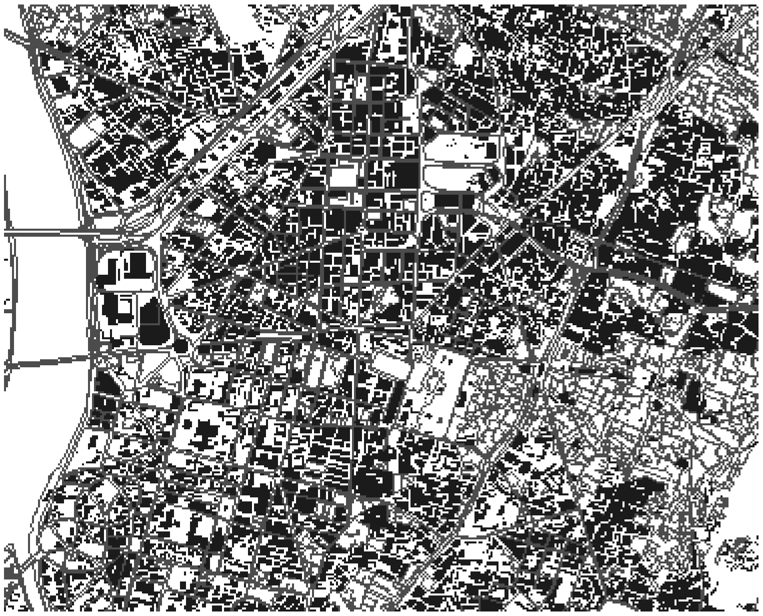

Figure 30. Same as in Figure 29, no reduction, size of serialized objects 3,056 Kbytes

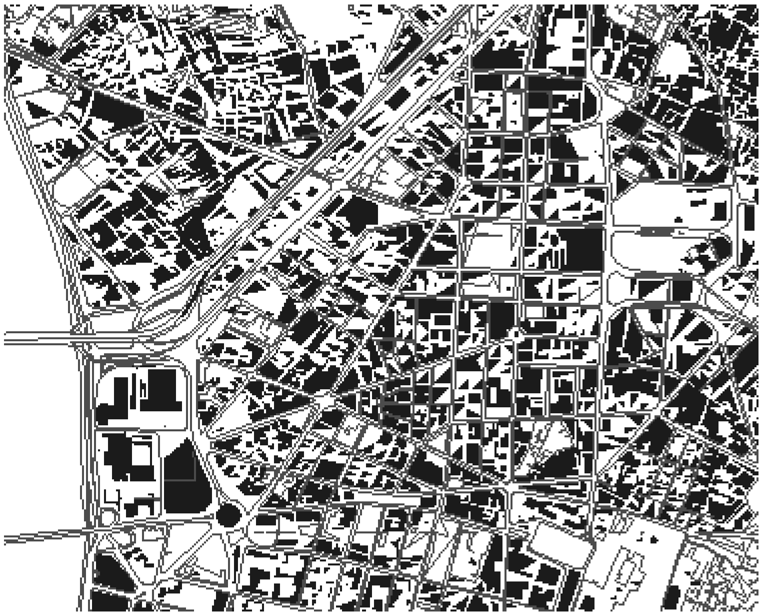

Figure 31. Zooming in Cairo map, RF 30%, size of serialized objects 997 Kbytes

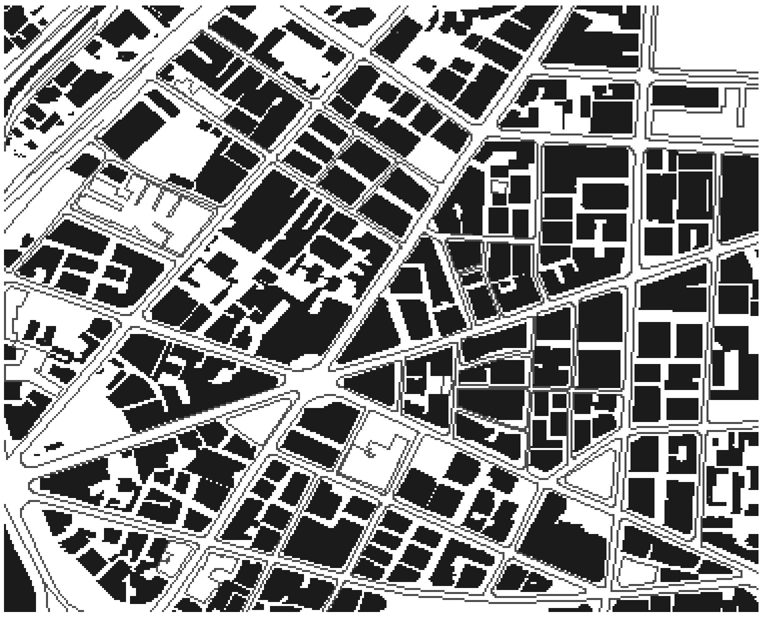

Figure 32. Downtown Cairo map, no reduction, size of serialized objects 366 Kbytes

4.2.2.3 Scenario #3

Besides zooming, the user could pan to view areas of the map that were beyond the boundaries of the display. In our implementation, panning results in neither the activation nor the deactivation of any filters. However, it could be the case that parts of the same map have different object density (e.g. a coastal city). In this case panning could view parts of the map that are drastically different in the density of objects that may call for the need to activate or deactivate filters. Although our implementation does not account for this situation, we see this as a possible point of interest for future research.

As we can see in Figures 33 and 34, panning left on the South Tyrol map increases the size of serialized objects from 512 Kbytes to 518 Kbytes. This is not much of a change but it could be a bigger change in a different map.

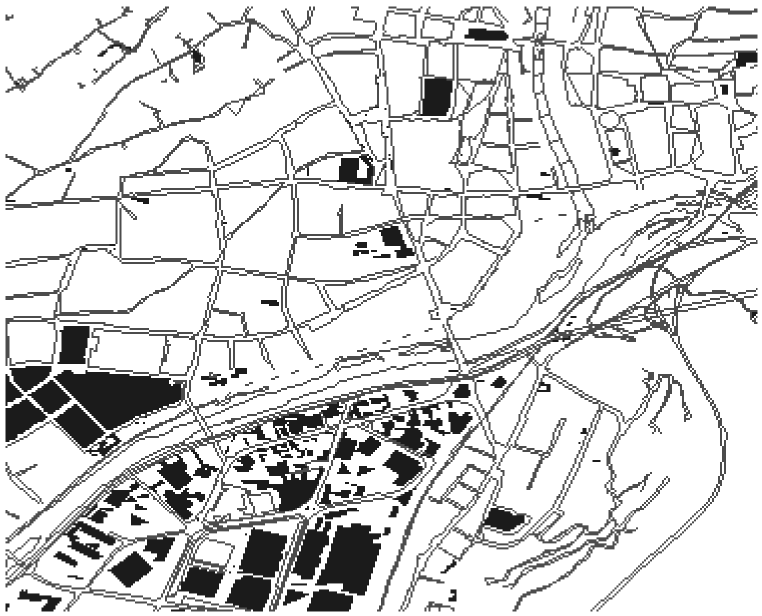

Figure 33. Zooming in S. Tyrol, FF 2& RF 30%, size of serialized objects 512 Kbytes

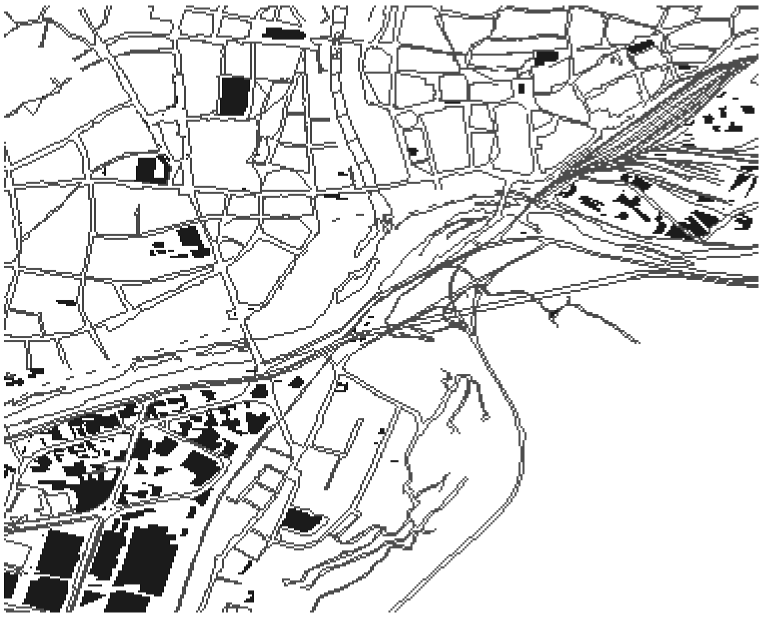

Figure 34. Pan left on S. Tyrol, FF 2& RF30%, size of serialized objects 518 Kbytes

As a result of these experiments and analysis of the results, we verified the effectiveness of the filters. In the next chapter, we will summarize this research and point out possible research directions for the future.

Chapter 5

Summary and Future Work

5.1 Summary

In this thesis we dealt with issues related to viewing and downloading digital spatial data to mobile devices. Map-client applications face many challenges in accessing data across a wireless network. Vector spatial data files tend to be large, and file sizes tend to increase unpredictably depending on the complexity of feature geometry.

Mobile devices must deal with limited and dynamically varying resources, in particular, the network quality of service (QoS). In addition, wireless devices have other constraints such as limited memory, battery power, and physical dimensions. Applications that execute in such environments need to adapt to the dynamic operating conditions in order to preserve an acceptable level of service.

Due to the limited size of the mobile device display, viewing all the details of the map could cause too much clutter and render the map virtually useless. Even if it is feasible to transmit all the details from a QoS standpoint, this could cause a problem from a usability standpoint.

This research effort aims to tackle the issues of QoS and usability on mobile devices by introducing a client-proxy-server model where clients are on mobile devices. The proxy performs two functions. First, it supplies the client with vital data about the status of the system that allows the client to take adaptive decisions aimed at maintaining the QoS. Second, it performs the adaptive actions requested by the client. There are two types of adaptive actions performed by the proxy, activating and deactivating filters. When filters are activated the amount of data transmitted from the server to the client is reduced. The client may decide to activate one or more filters to either maintain QoS or to limit clutter on the screen to enhance usability.

In this research effort, we used geometric generalization techniques to lower the data fidelity, developing an application-level set of data filters for vector spatial data. We also built a model for dynamic adaptation based on the Comma proxy platform. This is the first attempt to explore the possibility of using such techniques with the aim to maintain QoS for mobile mapping clients using the Comma proxy platform.

This thesis is a solid step forwards towards achieving the objective of providing better service for map users in wireless environments. It includes contributions in the following areas.

- Developing a stream filtering proxy (SFP) that can activate and deactivate filters.
- Developing filters that take advantage of map generalization techniques.
- A series of experiments and scenarios that not only verified the effectiveness and feasibility of the filtering concepts and implementation, but also achieved a better understanding of issues we should take into account when dealing with vector spatial data.
- Analyzing the experimental results to reach some conclusions and heuristics that may aid in optimizing the adaptation decisions taken by the client, including the following:
    - The user plays a vital role in the adaptation process by supplying his/her preferences regarding the importance or each layer, his definition of “too small objects,” allowing a certain filter to be used or not, etc. The user preferences could be used to decide on the filters to use, their combinations, and the parameters used with each.

- While maintaining QoS is a very important aspect for mapping applications, it is equally important to maintain usability of the maps. This means that the system has to strike a balance between lowering the amount of detail in the map to speed up the response to users' queries, and keeping enough detail in the map to be still useful for the user. In this thesis, we devised some heuristics about the kinds of filters to use and the amount of reduction to be achieved by each one to maintain that balance.
- When using FF alone, good reduction could be achieved by dropping about 30% of the layers without reducing the overall quality of the map by too much.
- There are direct correlations between the average number of segments per object in a map and the amount of reduction achieved using RF. This correlation could be used to approximate the amount of expected reduction in the overall map size given the percentage of reduction used by the elimination algorithm.
- When using EF, 16 pixels seems to be the most effective value for the attribute used with the filter in terms of the amount of reduction achieved compared to the degradation of quality resulting.
- When using combinations of filters, a reduction rate around 90% is usually optimum in terms of the amount of reduction achieved compared to the effect the reduction has on the quality of the map.

The thesis began with a background overview of wireless network characteristics and GIS. The Comma proxy system was chosen as the platform for this research, and was described in Chapter 2. The proposed adaptation model and detailed information about the design and implementation were described in Chapter 3. In Chapter 4, a number of experiments and real-life scenarios were designed to determine the effectiveness and feasibility of the proposed adaptation model and to evaluate the performance of the proxy. The experiment results were analyzed in detail.

5.2 Future Work

In the course of conducting this research effort we came across some issues that we believe could make for good future research opportunities.

5.2.1 Dynamic Adaptation Model Experimentation Tools

We originally planned to perform a set of adaptation experiments using Handheld PCs. However, due to time limitations we could not achieve that. Nevertheless, we designed and developed the experimentation environment described below to be used by future researchers.

5.2.1.1 Experimentation setup

In order to perform such adaptation tests, we need to have a way of emulating live network conditions. Network emulators are useful for lab testing of network-adaptive applications and protocols. The idea is to use the emulator to reproduce typical bad network conditions (packet loss, delay, etc.) and see how the application or protocol responds to it.

For the purposes of this experiment we can use NIST Net as a network emulation tool. NIST Net is a network emulation package that runs on Linux. NIST Net allows a single Linux PC to be set up as a router to emulate a wide variety of network conditions. Figure 35 shows the experiment setup we are proposing.

- NIST Net runs on a Unix machine acting as a router between the map client and server. The client and the server run on different machines and subnets. The client machine will be the only machine configured on its subnet to ensure all the network traffic going through the NIST Net machine is intended for the client.

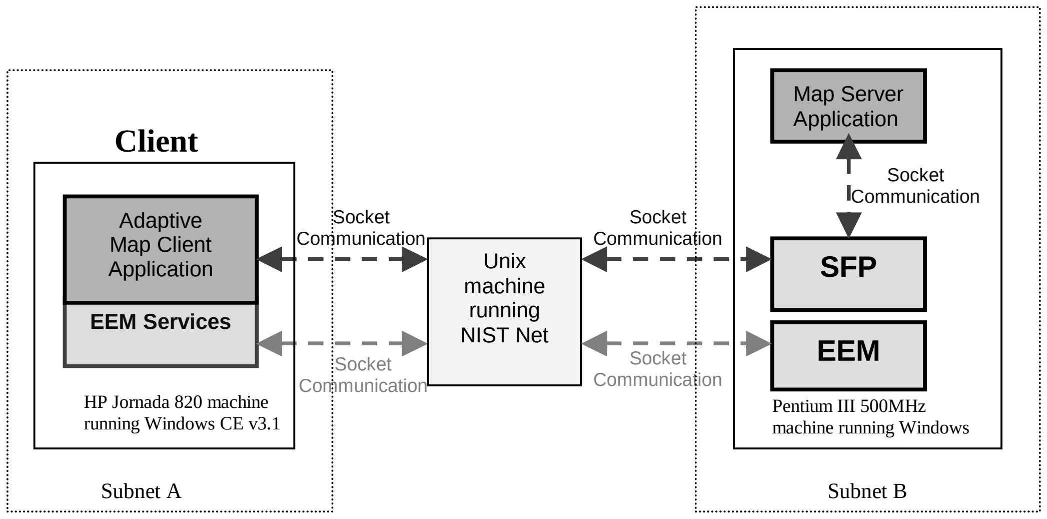

Figure 35. Experiment setup

5.2.1.2 Monitoring resources

Adaptation decisions need to take the environment changes into consideration. As mentioned before, to adapt we need a mechanism that provides the mobile application with this information. One central piece in our adaptive model is the EEM, which monitors and notifies the client of any changes in the environment.

To connect to the EEM, the map client instantiates an EEM services API and registers call-back functions for the variables it is interested in and how often to get updated. The EEM obtains information from instrumentations objects plugged into its instrumentation database. The instrumentation does the actual measurement of the variable, monitoring relevant environment parameters and notifying the application through registered call-back functions. To avoid interruptions of the client logic, these call-back functions execute in a separate thread.

We developed two Comma instrumentations to gather information about the mobile device and the wireless link. Therefore, the map client could register call-back functions with two instrumentations, one to monitor the network status called Network Instrumentation, and the other to monitor the Handheld environment called HPC Instrumentation.

The Network Instrumentation consists of several gathering units, one to gather information about the network using SNMP [41] based on the Management Information Base (MIB) RFC1213, another to gather information about the network throughput using TTCP [9] and a third one to measure the network latency using ping round-trip times to the default router.

The HPC Instrumentation uses several Windows CE structures to gather information about the handheld devices. For example, the MEMORYSTATUS structure contains information on the physical and virtual memory of the system, and the STORE INFORMATION structure contains information on the size of the object store and the amount of free space currently in the object store.

5.2.2 Allowing the User to Specify Preferences

As we saw earlier, a lot of the adaptation decisions taken by the client could be greatly influenced by a wide array of user preferences concerning the different aspects of adaptation. This includes the filters to use, their attributes, and the relative importance of the layers. In our implementation, we hard-coded many of these values to conduct our experiments. Adding the capability for the user to specify these preferences, and how the client could use this information in adaptation look like very promising research issues.

5.2.3 Using Automatic Algorithms to Decide on Filter Attributes

As we concluded, the reduction achieved by a certain filter could in some cases be approximated given some criteria. Devising automatic algorithms to perform these approximations could be a good area for future research.

5.2.4 Further Experimentation

The results we reached in this effort were based on the amount of experimentation we could perform within the time frame we had. We believe that further experiments performed on a wider array of maps could provide more results that could reinforce the results reached, fine tune them, or prove them to be true in certain situations only.

5.2.5 Improvement on the Filters

More measurements need to be taken into account in designing the filters. For our implementation of the Elimination filter, we based our algorithm on the resolution of the screen without taking into account the minimum size of a feature that can be displayed at a given scale. For example, "no lakes of less than 5 hectares surface should be displayed." Having a set of rules to specify the minimum feature size along with the map generalization techniques when activating the filters will be a good idea. Such as "areas (polygons) cannot be shown if they are smaller than the lines that draw them." For example, a polygon less than 250 meters wide cannot be shown when the map scale displayed is 1:250,000.

Another measurement we need to take into account is the data density. Data density is a measure of how many features per area can be viewed. Greater density implies more features in a given area, and therefore the screen can be cluttered. Taking this factor into account could cause panning operations to trigger adaptive actions to activate filters to keep the screen from being cluttered if the user pans to a high-density area of the map. Similarly, it could trigger adaptive action to deactivate filters to reveal more details if the user pans to a low-density area of the map.

When activating the Elimination filter, it could be a good idea to take the importance of the features into account when deciding what to eliminate and not to take the decision based solely on size. The importance of the feature actually stems from its relation to other features. For example, a bridge could be of high importance in spite of its small size. Eliminating this feature could cause a great deal of disturbance for drivers.

5.2.6 Other Filters

From the geometric generalization techniques, more filters can be designed to lower the data fidelity for vector maps such as collapsing and amalgamation (see Figure 6).

Collapsing consists of replacing objects represented by lines with points or area objects with lines; this is done as the user zooms out. For example, a single house becomes a point, a road becomes a line.

Amalgamation consists of joining of distinct objects into single representative shapes. For example, a group of small lakes can be joined with a larger lake.

Although these filters will reduce the data and achieve clarity by enhancing the visualization, especially on small screens, they have some issues. For example, collapsing to replace a polygon with a point is a relatively simple algorithm compared to generating a correct outline or a skeleton of a road, which could be a fairly complex algorithm.

For the amalgamation algorithm to give efficient results, objects should be of the same semantic class. In other words, the amalgamation should make sense with regards to the classification and the implied structure of the resulting amalgamated object. For example, it might be appropriate to amalgamate buildings that are part of a single block bounded by streets. It might make less sense to amalgamate buildings on opposite sides of an important street.

Bibliography

[1] T. H. Appleby, "Database design for geographical information systems," GIS For the 1990's Proceedings, National Conference, Ottawa, Canada, 1990, pp. 622-646.

[2] M. Arikawal, H. Kawakita, and Y. Kambayashi, "Dynamic maps as composite views of varied geographic database servers," Proceedings of the First International Conference of Applications of Databases, Vadstena, Sweden, 1994, pp. 124-157.

[3] Autodesk, "Autodesk MapGuide SDF Loader User's Guide," Autodesk, Inc., December 17, 1998, 1998. http://www.autodesk.com

[4] R. Bagrodia, W. W. Chu, L. Kleinrock, and G. Popek, "Vision, issues, and architecture for nomadic computing," *IEEE Personal Communications*, vol. 2, no. 6, pp. 14-27, 1995.

[5] B. P. Buttenfield, "Scale-dependence and self-similarity in cartographic lines," *Cartographica*, vol. 26, no. 1, pp. 79-100, 1989.

[6] B. P. Buttenfield, "Sharing vector geospatial data on the Internet," Proceedings, 18th Conference of the International Cartographic Association, Ottawa, Canada, 1999, pp. 39 - 45.

[7] CEM, *Downtown Cairo map*, Cairo Engineering & Manufacturing Co., 1999.

[8] E. P. F. Chan and J. Wong, "Querying and visualization of geometric data," Proceedings of 4th ACM International Workshop on Advances in GIS, Rockville, Maryland., 1996, pp. 131-140.

[9] Chesapeake, "Network Performance Testing with TTCP," ver. 1, 1999. http://www.ccci.com/product/network_mon/tnm31/ttcp.htm

[10] City of South Tyrol, *City of South Tyrol map*, City of South Tyrol, 1999. http://www.comune.bolzano.it/carta/cart5000v/

[11] K. C. Clarke, *Analytical and Computer Cartography*, 2nd Edition. Englewood Cliffs, New Jersey: Prentice Hall, 1995.

[12] M. Creech and C. Fedora, "A Beginner's Guide to Geographic Information Systems," School of Information, University of Michigan, Michigan, 1996. http://www-personal.si.umich.edu/~karleric/GIS/gis.html

[13] J. Dangermond, "What is a Geographic Information System (GIS)?," in *Geographic Information Systems (GIS) and Mapping - Practices and Standards*, E. A. I. Johnson, C. B. Petterson, and J. L. Fulton, Eds. Philadelphia: ASTM, 1992, pp. 11-16.

[14] E. de Lara, D. S. Wallach, and W. Zwaenepoel, "Puppeteer: Component-based adaptation for mobile computing," Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems, San Francisco, California, 2001. http://www.cs.rice.edu/~delara/papers

[15] D. H. Douglas and T. P. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Canadian Cartographer*, vol. 10, no. 2, pp. 112-122, 1973.

[16] D. Duchamp, "Issues in wireless mobile computing," Proceedings of Third Workshop on Workstation Operating Systems, Key Biscayne, Florida, 1992, pp. 1-7.

[17] Engineering Bureau, *City of Clearwater map*, City of Clearwater,Florida, 2000. http://www.clearwater-fl.com/engineer/atlas/

[18] ESRI, "About GIS: How GIS works," ESRI website, 2000. http://www.esri.com/library/gis/abtgis/gis_wrk.html

[19] Faculty of Environmental Studies, *University of Waterloo map*, Faculty of Environmental Studies, University of Waterloo, 1999. http://library.uwaterloo.ca/locations/umd/cart/vectordata.html

[20] G. H. Forman and J. Zahorjan, "The challenges of mobile computing," *IEEE Computer*, vol. 27, no. 4, pp. 38-47, 1994.

[21] A. Fox, S. D. Gribble, E. A. Brewer, and E. Amir, "Adapting to network and client variability via on-demand dynamic distillation," *SIGPLAN Notices*, vol. 31, no. 9, pp. 160-170, 1996.

[22] A. Fox, S. D. Gribble, Y. Chawathe, and E. A. Brewer, "Adapting to network and client variation using infrastructural proxies: Lessons and perspectives," *IEEE Personal Communications*, vol. 5, no. 4, pp. 10-19, 1997.

[23] A. Guttman, "R-Trees: A dynamic index structure for spatial searching," Proceedings of ACM SIGMOD International Conference on Managment of Data, 1984, pp. 45-57.

[24] C. Jones, *Geographical Information Systems and Computer Cartography*, 1st Edition. London: Longman (Pearson Education), 1997.

[25] A. D. Joseph, J. A. Tauber, and M. F. Kaashoek, "Building reliable mobile-aware applications using the Rover toolkit," Proceedings of the 2nd ACM International Conference on Mobile Computing and Net-working (MobiCom ’96), Rye, New York, 1996, pp. 117-129.

[26] I. Kamel and C. Faloutsos, "Hilbert R-tree: An improved R-tree using fractals," Proceedings of 20th VLDB, 1994, pp. 500-509.

[27] R. H. Katz, "Adaptation and mobility in wireless information systems," *IEEE Personal Communications*, vol. 1, no. 1, pp. 6–17, 1994.

[28] D. A. Kidston, *Transparent Communication Management in Wireless Networks*, Master thesis, Computer Science Department, University of Waterloo, 1998. http://www.shoshin.uwaterloo.ca/publications/index.html

[29] J. J. Kistler and M. Satyanarayanan, "Disconnected operation in the Coda file system," *ACM Transactions on Computer Systems*, vol. 10, no. 1, pp. 3-25, 1992.

[30] M. Kofler, *R-trees for Visualizing and Organizing Large 3D GIS Databases*, P.h.D. Thesis, Computer Science Department, Technical University Graz, 1998.

[31] T. Kunz and J. P. Black, "An architecture for adaptive mobile applications," Proceedings of Wireless 99, the 11th International Conference on Wireless Communications, Calgary, Canada, 1999, pp. 27-38. http://www.shoshin.uwaterloo.ca/publications/index.html

[32] J. I. Mac Cuaig, "Geographic information systems: New technology for the transmission of high volume database information by satellite," GIS For the 1990's Proceedings-National Conference, Ottawa, Canada, 1990, pp. 401-405.

[33] M. McIlhagga, A. Light, and I. Wakeman, "Design choices for adaptive applications," The Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking, 1998. http://www.cogs.susx.ac.uk/projects/lowband/papers/DesignChoices.rtf

[34] G. Mocquard, *A new Environment Monitoring Architecture for Adaptive Mobile Applications*, Shoshin publication, Computer Science Department, University of Waterloo, 2000. http://www.shoshin.uwaterloo.ca/publications/index.html

[35] J. Moline, "Inline support for vector files: Engineering drawings on the WWW," Proceedings of the Second World Wide Web Conference '94: Mosaic and the Web, 1994. http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/DDay/moline/moline.html

[36] L. B. Mummert, M. R. Ebling, and M. Satyanarayanan, "Exploiting weak connectivity for mobile file access," Proceedings of the 15th ACM Symposium on Operating Systems Principles, Copper Mountain Resort, Colorado, 1995, pp. 143-155.

[37] B. D. Noble, *Mobile Data Access*, P.h.D. thesis, School of Computer Science, Carnegie Mellon University, 1998.

[38] B. D. Noble, M. Price, and M. Satyanarayanan, "A programming interface for application-aware adaptation in mobile computing," Second USENIX Symposium on Mobile and Location Independent Computing Proceedings, Ann Arbor, Michigan, 1995. http://www.usenix.org/publications/library/proceedings/mob95/noble.html

[39] B. D. Noble and M. Satyanarayanan, "Experience with adaptive mobile applications in Odyssey," *Mobile Networks and Applications Journal*, vol. 4, no. 4, pp. 245-254, 1999. http://www.baltzer.nl/monet/contents/1999/4-4.html

[40] B. D. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. R. Walker, "Agile application-aware adaptation for mobility," *Operating Systems Review (ACM)*, vol. 51, no. 5, pp. 276-287, 1997.

[41] M. T. Rose, *The Simple Book: An introduction to Management of TCP/IP-based internets*. New Jersey: Prentice Hall, 1991.

[42] H. Samet, *The Design and Analysis of Spatial Data Structures*. Reading, MA: Addison-Wesley, 1990.

[43] M. Satyanarayanan, Fundamental challenges in mobile computing," Fifteenth ACM Symposium on Principles of Distributed Computing, Philadelphia, Pennsylvania, 1996, pp. 1-7.

[44] M. Satyanarayanan, "Hot topics: Mobile computing," *IEEE Computer*, vol. 26, no. 9, pp. 81-82, 1993.

[45] R. F. Tomlinson, "Geographic information systems: a new frontier," in *Introductory Readings in Geographic Information Systems*, D. J. Peuquet and D. F. Marble, Eds. NewYork: Taylor and Francis, 1990, pp. 67-85. http://www-personal.si.umich.edu/~karleric/GIS/concepts.html

[46] T. J. Whalen, *Design Issues for an Adaptive Mobile Group Editor*, Master Thesis, Computer Science Department, University of Waterloo, 1997. http://www.shoshin.uwaterloo.ca/publications/index.html

[47] J. D. Wilson, "Taking IT to the streets," in *Business Geographic*, 1999, pp. 4-5. http://www.geoplace.com/bg/1999/0299/299mob.asp