Presented at WWW10, Hong Kong, May 2001
This version based on web proceedings, see: www.www10.org
Conference proceedings also published by ACM

Map Adaptation for Users of Mobile Systems

Dan Chalmers, Morris Sloman, Naranker Dulay
Department of Computing
Imperial College of Science, Technology and Medicine
180 Queen's Gate, London SW7 2BZ, U.K.
{dc,mss,nd}@doc.ic.ac.uk

ABSTRACT

Display characteristics, network Quality of Service, and the user's current task all exhibit a wide range of variation when users interact with mobile and ubiquitous devices. It is desirable to enable applications to adapt to these variations. The user's experience in interacting with the application can be significantly enhanced by adapting the data presented. However, we find that naive degradation of data can quickly result in an unacceptable presentation.

In this paper we present a means of describing compound documents and preferences according to the semantics of the data, and an algorithm for performing a selection amongst the offered data based on these descriptions. Our model of the user's preferences is more descriptive than most common specifications used for Web-based applications. These have been implemented for a map viewing application, along with application level network resource management. We present test results for adapting to network bandwidth and a download deadline. We show the technique is applicable over two orders of magnitude in bandwidth variation, and successful in meeting deadlines. By degrading the data presented in sympathy with the user's needs, the degradation has less impact on the user for any given benefit in download time than arbitrary selection.

Keywords

Adaptive application, quality of service.

1 INTRODUCTION

Many applications are designed with the assumption that they are to be run on a desktop system of some common specification. The use of applications on mobile or embedded devices exposes the user to a range of variation in behaviour which many of today's applications are not able to accommodate gracefully. This variation is a result of various factors: a wide range of device sizes and user interface devices; a variety of network connections and the environmental effects many of these experience; and also a range of new or modified working practices and tasks which may evolve. Future networks may include active nodes to add dynamic hyperlinks to allow the user to access relevant information services, e.g., a map display can be enhanced, by one or more monitoring services, to indicate congestion or fog by modifying the feature data such as colour of a road. These additional data services may stimulate new applications for mobile computing. The wide range of data may easily lead to information overload: for the display, network bandwidth, or for the user's ability to digest it. Too much dynamic information can prove very distracting. A selection mechanism must therefore be able to differentiate between necessary, interesting and unnecessary data (and many degrees in between). This must be balanced with timely delivery, screen space availability, cost, etc. Different users, in different situations, will have widely varying preferences. Successful management of these issues, in a predictable and unobtrusive manner, will be key to the success of advanced applications.

Current user activity can influence preferences for adapting to system variation, and offered data services. A motorist will have different preferences for information compared to a pedestrian. Likewise a tourist may wish to take the scenic route through a town and be presented with some information about the districts they are travelling through, and nearby attractions. In contrast, emergency services simply need the fastest route, given current traffic conditions. A wheelchair user might be interested in indications of ramps and other features for easy access. Blind users may prefer data in certain formats, which can be output in audio. A maintenance engineer, using paper based manuals, would have to return to base to collect relevant documentation if he discovered an unforeseen problem or the equipment, at a remote site, was different from that expected. If relevant plans and manuals can be downloaded, then the engineer can more easily deal with unexpected problems or be diverted to a new call without returning to base i.e. the information available to the engineer has to be adapted to the current task

There is a need to provide for Quality of Service (QoS) management and content awareness in mobile and ubiquitous computing to manage the incompatibility and frustration which can arise from these variations in device capabilities and user preferences [1,2,3,4]. The wide range of variation renders many of the low-level network approaches to QoS management inappropriate or infeasible. The range and evolution of devices and user needs makes tailored design of applications and data a moving target, which is hard to meet. Application aware adaptation [5] seems, then, to be appropriate: to describe data, user preferences and application interactions enables application level decisions about the most appropriate way to adapt to a situation. We believe that it is need for a more flexible solution than just the provision of standard translations or restrictions according to device parameters, as in [6,7,8,9]. These may be part of the solution, particularly for very small devices. However, when using devices of the next level of capability (PDA to laptop, kiosks, in-car systems, etc.), the device itself ceases to be an absolute limitation. The user interface and network connection are often both sufficient to support more than the most basic data presentation, although still being limited in comparison to desktop PCs. The choices to be made in adjusting to the limitations are then directed by both the device and the user's preferences, which may vary widely. Maintaining a user-driven specification, while offering any available dynamically generated assistance can better support device and user centric adaptation of the data presentation, as more detailed and dynamic information can be available to the adaptation process.

Providing map data to users engaged in a range of tasks, using a range of devices is an example of the needs outlined above. There is a large body of location correlated data available, which may be needed for a specific task, and can potentially be displayed on many different devices. The standard vector map formats provide data with a rich structure typical of many emerging media standards. We have focused on map-based applications, but our techniques are intended, and implemented, to be generally applicable to many other application such as remote learning or web browsing from mobile devices.

The rest of this paper is structured as follows: in section 2 we expand on the problem being tackled. Section 3 describes the techniques we have applied in our solution. Section 4 describes an example interaction. Test results are presented in section 5. Section 6 compares our work with other approaches, and we conclude in section 7.

2 DELIVERING DYNAMIC MAPS TO MOBILE DEVICES

Some data, such as maps, has content which is of general interest, and content which is more specialised. When navigating in a car, details of roads, traffic, etc., are of key importance, while hikers may consider footpaths, hills and rivers to affect their journey more significantly. Delivery drivers and emergency services will probably know the general geography of an area, but may want to identify detail such as house numbers and temporary diversions. Much other information is available on maps: administrative boundaries, spot heights, etc., which might clutter the map on a small screen and be undesirable. While most map information is static in nature, feature updates can be propagated by on-demand delivery in a more transparent manner than for statically stored maps such as on local CD. The potential volume of map data is also high, making static storage on limited devices undesirable. As mentioned above, highly dynamic information, such as weather forecasts, traffic conditions or local entertainment information could be provided over the geographical features on a map through overlays or hyperlinking. Just-in-time delivery of this information is preferable for a range of tasks.

There are various terms we use in our work to describe the structured multimedia data we are working with. Some of these suffer from many uses in computer science, so we shall briefly outline the key terms below:

- A **document** is a unit of presentation for structured multimedia information. Examples include a map or web page. A document may be considered to be a collection of one or more *elements* which form the unit of representation.
- An element is a generalised part of the data, which fulfils a specific role in a document. In a map example this may correspond to a **feature**, e.g., the M1 motorway, A427 road, River Thames. Elements are identified by a **type** which defines its semantic content e.g., road, river or building. Motorways, major-roads or A-roads, minor-roads or B-roads are all sub-types of road. The Dublin Core meta data [10] also uses the term "type" to describe a similar concept.
- An element may be represented by multiple **variants**. For instance, a map may contain representations of the M1 motorway surveyed at 1:10000, 1:50000 and 1:100000 relating to lesser inclusion of fine detail on small bends, etc. Similarly, a picture can have variants relating to different resolution. A variant has an **encoding format**, describing its syntactic encoding. In our work we use MIME-type descriptions. A variant is described by **parameters**, which may be general, such as size (in bytes), or encoding format specific, such as survey scale, and survey date.
- We permit a document to be an element in a compound document, so a map **tile** which is a unit of presentation, can be considered as a document. One tile's relationship to another is defined by the area it describes. A web page is a compound document with links to other web pages.
- An element may contain other elements, e.g., to add labels to the representation of a road, or a picture in a web page contained by the HTML which refers to it. By being contained, these elements rely on the containing element to be present in order that they may be displayed. Note that the graphical representation of a road and the textual label of the road are two different but semantically related elements. Labels identifying roads are meaningless without the representation of the road they identify, although some roads may be unlabelled. An element such as a road label may occur at multiple places for a long road in a map.

We illustrate the class structure we use for maps in figure 1, using UML. An *Atlas* is described as being a collection (aggregation) of *Map Tile*, both of which are sub-types of *Document*. The *Map Tile* is an aggregation of *Feature*, of various types. The types describes the semantic type of the element. We show types which have been specialized in the case of *Road*. Clearly other feature types would also sub-type *Feature*. This diagram is abstract, and applies over the whole of the atlas. We illustrate an instantiation for a specific map tile in figure 3. Each element may be represented by one or more variants. We illustrate variants which offer alternative representations of the same semantic information. The *Variant* type is shown as having a specialisation, *Map Variant*, which is capable of describing information specific to maps.

2.1 Selection of Information

As mentioned in section 1, there is a need for users to select specific information to be displayed. Current solutions for web based applications fall into three broad categories:

- Specific types of data may be restricted. This restriction is generally implemented as a binary selection on the client application, e.g., the "no images" selection commonly found on web browsers.
- A weighting is defined for a specific encoding format (e.g. GIF = 0.9, JPEG = 0.8) or text language (e.g., French = 0.8,

English = 0.5) to indicate the user's preference or capability of the user's device. These weights are generally used in a selection mechanism provided with the data request [7,8,9].

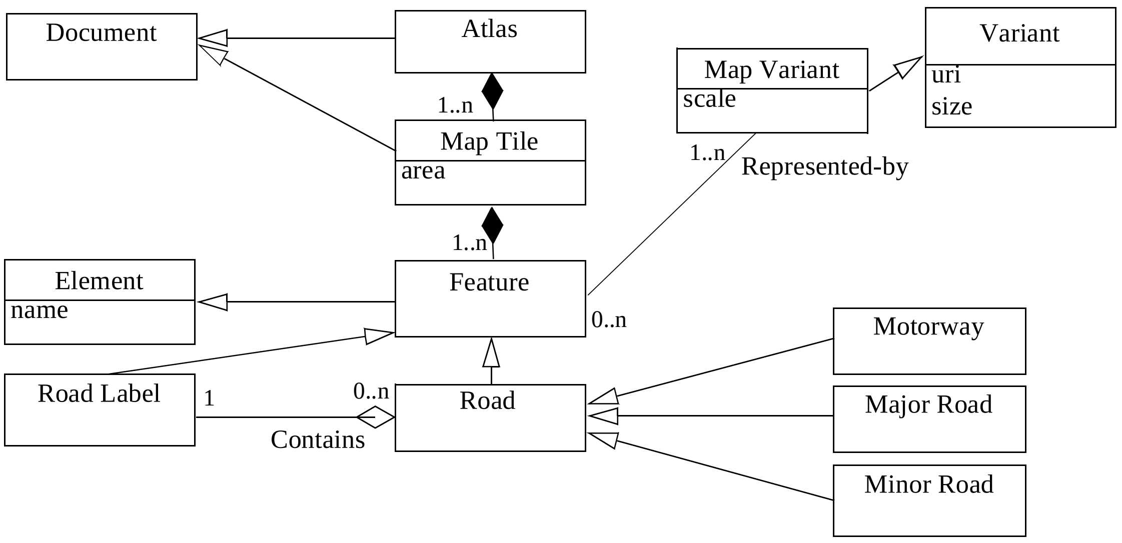


Figure 1: Extract From Class Model For Representation Of Maps

- A standard transcoding may be applied to data in order to provide it in a particular format, or to meet device restrictions. Commonly image size and colour depth are adjusted to meet the needs of restricted client devices. These transcoders are generally implemented as in-network proxies offering transcoding for a small range of target devices [6].

These techniques suffer from significant limitations, particularly for the class of applications we are studying. Where structured data are being used, selection according to the data's encoding format is unlikely to fully capture the user's needs. In web browsing it may be sufficient to say that one prefers particular languages, or that one dislikes images. However, for a vector map, format selection based on encoding format does not perform a meaningful selection. Where a large amount of information is contained within one encoding format it is desirable to be able to express preferences related to the *semantics* of the information, i.e., display roads but not contour lines where both may have the same encoding format.

As the range of device capability and tasks expands, it is undesirable for content providers to be obliged to provide many different versions of their content, each adjusted to a small range of devices or users. However, the user is best served by data which is finely adjusted to their needs. A mechanism for content negotiation which allows for general specification of requirements is needed. Many other proposals for media selection in response to context or resource constraints are applied to each element of the data separately [7]. This per-element selection limits the range of possible trade-offs, and does not allow for consideration of the interaction of the various data elements within the whole document presentation.

The network link capacity may also restrict the amount of data which can be transferred in a reasonable time, leading to long and variable delays between a request for a map tile or web page and its display. For many tasks timely data provision is key. A data selection mechanism should therefore enable deadlines to be met by restricting the data selected. Deadlines may be due to a number of factors, such as the user's patience, and the need for timely display due to movement. The first case is a simple one experienced for many applications, and often cited as a source of irritation in web browsing. The second is most relevant during mobile use, where a map segment must be displayed in advance of needing it, e.g., in a car, a navigator prefers to understand the layout of a junction in advance of reaching it, in order that the driver can be given clear directions. Conversely, old data may give out-of-date information, as congestion can change quite rapidly on roads. Deadline-based selection may be achieved in a similar manner to the selection according to display capability and the user's requirements and must be performed in conjunction with user preferences. For more capable devices the network connection may be the most significant limiting factor in selecting data.

We are addressing the use of map data, both general and specialised, in specific tasks. The restriction of the domain gives the user a clear means of identifying what information is relevant. The restriction of the data allows standardised descriptions of the semantics of the content to be developed. Similar techniques may be applied to less restricted domains, once suitable and commonly understood task and data descriptions are developed. Where the user base and information are well controlled, such as licensed databases or corporate networks, the definition of the tasks and descriptions is also clearer, these techniques can be extended to other information retrieval problems.

3 OUR APPROACH TO MEDIA SELECTION

We have used a simple separation of concerns to define what information comes from which party. We address the description of the data, and of the user's requirements, which together lead to a selection process. Our approach to these issues, and a brief description of our approach to resource management and context awareness are described below.

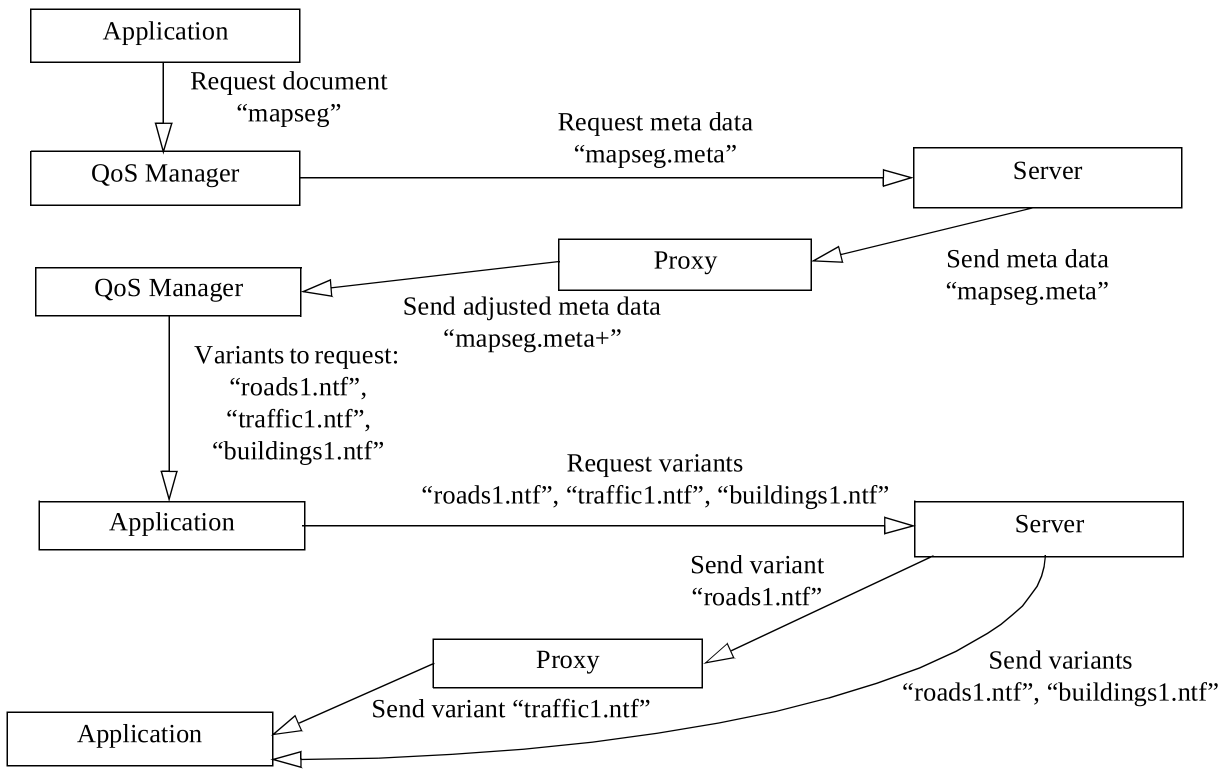

Application
Request document "mapseg"
QoS Manager
Request meta data "mapseg.meta"
Server
Proxy
Send meta data "mapseg.meta"
QoS Manager
Send adjusted meta data "mapseg.meta+"
Variants to request: "roads1.ntf", "traffic1.ntf", "buildings1.ntf"
Request variants "roads1.ntf", "traffic1.ntf", "buildings1.ntf"
Application
Server
Send variant "roads1.ntf"
Proxy
Send variants "roads1.ntf", "buildings1.ntf"
Application
Send variant "traffic1.ntf"

Figure 2: Illustration of Interaction with Meta Data

3.1 Interaction Process

In summary, the process of retrieving data, illustrated in figure 2, is:

a. The application makes a request to the QoS manager. The QoS manager requests the meta data.

b. The server's response describes the various elements of the data, and the variants of those elements it has to offer. The server, when responding with meta data, may make a provisional reservation of resources, based on the variants offered. In cases of high load it may adjust its offer.

c. Any intermediate nodes (proxies) append to the meta data and pass it on. The addition advertises their capability to modify the data (e.g., add traffic information, compress data, or text language translation).

d. The client receives the meta data, and makes a selection amongst the variants. It reserves sufficient resources locally to support the request.

e. The request is then passed back, the proxies and any low-level resource management capable network elements also making reservations. The server then adjusts its reservation to reflect the chosen variants.

f. The data requested is returned to the client, passing through any proxies whose services were selected.

3.2 Meta Data

Document authors (or their agents) describe the structure of the document's elements and their variants, and the objectively measurable attributes of the variants. Each element of the document can be represented by one of a set of variants, and/or refer to a more specialised element. Variants represent data with a similar purpose, but differing characteristics. For instance, features may be presented with different minimum feature sizes, corresponding to the level of detail presented at different map scales in the paper world. So, a change in scale can affect both feature inclusion, and detail in the description of features. For instance, roads may be presented in variants which show progressively more detail such as small bends. Variants may also be used to encode data in different formats, e.g. maps may be presented using vectors or rasters. In addition meta data will describe system-level properties, such as size (bytes).

An element can be described as containing other elements, to enable the description of the various components of a structured document. For instance road information on a map may be represented by the basic road definition, and an optional contained representation for the labels identifying the road. More detailed basic representations might contain further contained elements, e.g. boundaries of the road (e.g. wall, hedge, open).

In general new elements should be introduced to contain semantically different information, and variants should be used to offer different quality versions of the data. Variants are units of data which may be individually requested. There need not be a 1:1 relationship between element instances and variants. Meta data describes the instances and relationships such as those shown in figure 3, of the class model, as described in figure 1.

Figure 3 illustrates four variants (roads1v, mways1v, mways2v, m1v), each of which represents the "M1" motorway, and in all but one case other feature instances as well. If one were only interested in the "M1" motorway, one of these variants would be selected for retrieval from the server, and any possible elimination of unwanted features made locally. Instances of elements may be anonymous, such as for “Minor Road” and “Road Label”, where the name is “*”. This notation describes a container element for all instances of the type, without the need to explicitly identify each one. This is useful where there is no clear identity for many small features, or in order to trade smaller meta data for expressive power.

The variants may be pre-computed files, e.g., all motorways, or dynamically selected data, such as a particular road from a database. The trade-offs involved in these retrieval mechanisms are not described here. Our techniques can accommodate either, although the overhead in describing very fine grained features may become excessive. Subjective meta-data requiring significant human effort to author is not always popular with content creators, and is technically hard to produce where content is dynamically generated. By concentrating on automatically derivable meta data its use becomes less of a burden to content providers. If a proxy is offering translations on data then the meta data must be modified to reflect this. For instance, a proxy which performs some standard translation on images for PDAs would describe the results of the application of its function over each image. The modified meta data would describe the expected parameters of the generated image as the proxy does not yet have the image to act on. Caching transcoders might offer greater accuracy, and faster delivery.

Web pages may be represented in a similar way. The structure of frames and included images, etc., is supported by the containment of elements in variants. A “no-frames” version of the page may be described by the page element being represented either by a frameset, or by a single-frame HTML file. Different language versions may be described as different variants of each element.

3.3 User Specifications

The user (or their agents) define their preferences for element types (what semantic part it plays in the document, e.g., road, vegetation, administrative boundaries); various media encoding formats (such as text, image, vector-map, etc.); how they perceive the quality of the presented data to vary according to its attributes, including limits of tolerance for low quality and of perceived improvement for high quality; and goals in the presentation, e.g., download time.

Users specify their preference for elements of a particular type by associating a **weight** with the type, for instance, to describe a preference for displaying information about roads rather than rivers. The following type, value pairs define the default weight for any type to be 0.1, for roads to be 0.4 and for major-roads to be 0.7. Higher numbers indicate stronger preference.

```
type=*: weight=0.1
type=road: weight=0.4
type=major-road: weight=0.7
```

The types are structured through specialisation, and one weight definition may supersede another where one defines a weight for a general type, and another for a more specific type. Based on the above preferences, the following 3 element types would be allocated weights as shown below. A minor-road takes the default weight for roads, and vegetation takes the default for all other types. A variant representing several elements takes the highest weight of those elements.

```
type=major-road => weight=0.7
type=minor-road => weight=0.4
type=vegetation => weight=0.1
```

The encoding format weight defines a preference for different syntactic representations of data in a similar manner to the type weight. We have limited the data encoding format to NTF (a vector format for maps) in our current work, and so we shall not discuss the use of this weighting further here.

Preferences for different variant representations of an element are defined using a **utility function** applied to the parameters of the variants, similar to that in [11]. For example, a low resolution picture will be very grainy, a medium resolution picture will display well, but a high resolution picture will be too large for the screen size, requiring scrolling to view it. To capture the effect of these limitations the perceived utility function would be defined in terms of the resolution parameter of the picture variants. We describe our use of utility functions in more detail in section 3.4.

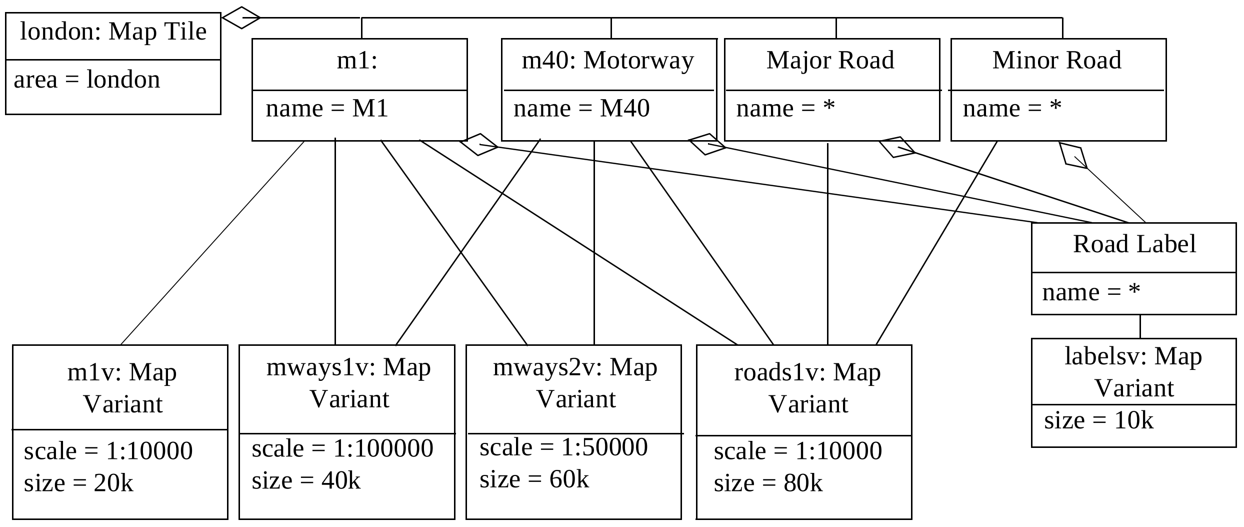


Figure 3: Instantiation Of Document, Elements, Element Groups And Variants For A Map Tile

The utility function is associated with parameters, which are generally specific to an encoding format. In the case of maps, we have an encoding format specific handler that understands meta data relating to survey scale. Differing scales imply differing levels of detail in the representation. For maps, the utility function may relate to both encoding format and type, e.g.:

```
(encoding format=map/*, type=*,
 parameter=scale): uf=uf_gen_scale
(encoding format=map/*, type=road,
 parameter=scale): uf=uf_important_scale
```

In this example uf_gen_scale may describe the scale as acceptable over a wide range, but for uf_important_scale the utility returned may fall off much faster with increasing scale. An element of type road, or any sub-type would take the more specific utility function, "uf_important_scale".

For parameter based utility, the encoding format must be at least partly known, as parameters are specific to an encoding format, e.g., survey scale is only a meaningful parameter for map data. When defining utility functions applicable to a general encoding format, e.g., "parameter = "scale" for encoding format = "map/*", it is not necessary to re-specify that definition for more specialized formats, e.g., "map/ntf", unless different behaviour is required.

The number of utility functions can be minimised by operating at the most general encoding format level available. A default utility for an encoding format and/or type is given, to support the case where no measured parameters are available. For default utility the greater of the utilities due to encoding format and type is taken. It should be clear that these utility functions do not say anything about the perceived importance of roads within a map. If one were navigating it is likely that the "road" data would have a high type weighting. However, a tourist might be less interested in the detail of the road than in historical monuments and restaurants, so loading unnecessary detail would delay the loading of more important information over a wireless network, and clutter the display. Utility functions over parameters can aid in describing these preferences. When defining user preferences it is important to be clear whether one is describing the importance of a feature (type weighting) or the effect of the representation quality on the perception of the data (utility function over parameters of a variant).

The final part of the specification is rather simpler. This part describes goals, or constraints, over the document retrieval. To date the only goal we have worked with is a download deadline. Where bandwidth is limited with respect to the download deadline this has the effect of constraining the variants selected. We describe our process for selecting variants in section 3.5.

In figure 4 we illustrate a user specification, giving type and encoding format weights; a deadline; and utility functions to be applied over the scale parameter for different types, and a single function to be applied over the survey date. The type weight specifications describe a preference for road features over water features, etc. Here the most specific weight for a motorway is for road, which is a super-type of motorway. The utility functions over scale may describe a perception that the loss of utility with increasing scale is greater for topographical details than for roads. For instance, a hiker will want considerable detail to navigate safely through an area with cliffs, whereas a loss of precision in the description of a road is less likely to be dangerous. Similarly the utility function over the survey date would indicate that up-to-date data are to be preferred.

Type Weights

```
type=road: weight=0.75
type=road-name: weight=0.73
type=water: weight=0.66
type=fence: weight=0.53
type=building: weight=0.5
type=topology: weight=0.41
type=vegetation: weight=0.23
type=building-name: weight=0.09
type=boundary: weight=0.04
type=*: weight=0.03
```

Encoding Format Weight

```
encoding format=map/*, weight=1
```

Utility Functions

```
(type=*, encoding format=map/*,
  parameter=scale): uf=gen_map_detail
(type=water,fence,vegetation, encoding
  format=map/*, parameter=scale):
  uf=med_map_detail
(type=topology, encoding format=map/*,
  parameter=scale): uf=important_map_detail
(type=*, encoding format=map/*,
  parameter=survey date): uf=recent_is_better
```

Goals

```
download deadline=15seconds
```

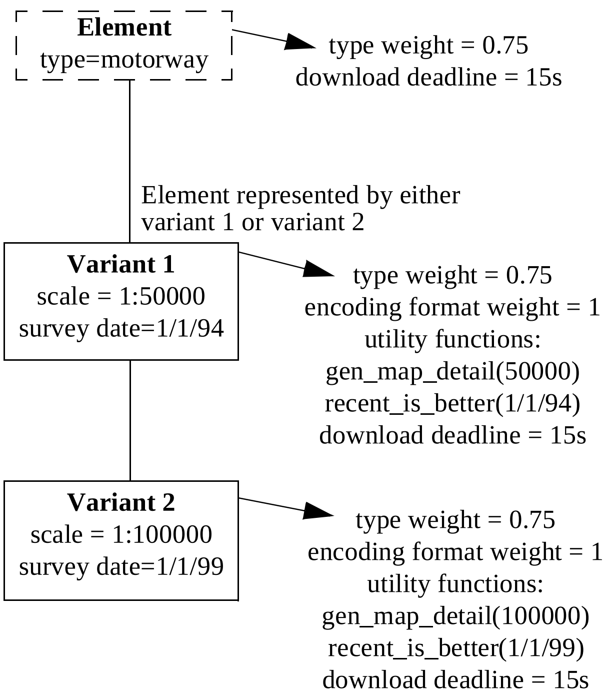


Figure 4: An Example of Weights and Utility Functions in a Specification and their Application

Our use of weights and utility functions may lead to a somewhat complex description of requirements, which are inappropriate for "average" users to create or edit. However, we believe that having specialist rule creators provide rules for users who possibly only select between a few sets of rule classes would be an appropriate solution. A selection of utility function combinations due to device and task might be abstracted by a higher level user interface. In the context of the map viewer application we have been developing, the restricted problem domain assists in the specification of effective preferences. The selection of utility functions, weightings and goals should be context dependant. Thus the need for context aware resource management, and the possibility of context-dependant applications raises the need for some automated system-wide context awareness. Task, device and location awareness are our immediate focus.

Further examples of how we envisage context to affect utility functions, weightings and goals are given below:

- When the cost of the link is great, such as for mobile phones, goals can be defined to reduce cost. Possibly correlating cost to overall utility. The assumption being that one is prepared to pay for more interesting data.
- Location and proximity information may cause changes in behaviour. For instance, "in the car" may be identified as placing some urgency about fetching map data. Utility functions, encoding format and type weightings may also be varied to achieve more suitable data presentation.
- When using a screen of restricted colour capability, the utility functions over colour depth may reflect this. Rather than taking the user's limit for perception of colour depth, the screen's ability to render colour defines the limit.
- When driving a car, audio data (or text which can be rendered as speech) may be preferred over visual data. Where data has to be graphical, key information will be rendered as highly important, but detail will quickly be described as unimportant, to avoid distraction.

3.4 Utility Functions

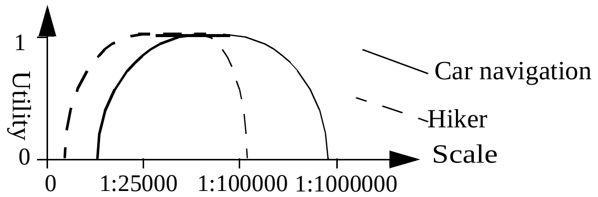


Figure 5: Example Utility Functions for Scale

A key part of the selection process is the calculation of utility scores for variants. We shall now illustrate a simple slope utility function. Other functions describing "s" curves, or which indicate an optimum value range may also be used. To illustrate the use of utility functions based on types of data, we shall illustrate possible variations from our example map. For in-car navigation various features are relevant. For roads, one might define utility functions relating to scale as depicted by the solid line in figure 5. Here a lot of detail makes the map harder to read, while a large scale is often still useful. For hikers, there might be less of a tolerance for the loss of detail of bigger scales, but more time to read and use the greater detail of the smaller scales. The dashed line in figure 5 illustrates this different perceived utility. Our illustrative graphs are not based on user studies, but show intuitive trends we might expect.

Our use of utility functions is similar to their use in rating utility of more common multimedia encoding formats. As the size of an image increases its utility may rise, in general. One might qualify this specification to have utility fall off once the image started to become larger than the display could show at one time. This general utility function might be overridden for types of image such as icons and logos, in order to load images whose size is appropriate to the display (unless it is actually the logo one in interested in). Other formats may require discrete functions, for instance, over the language of text, e.g. a native English speaker may give a high rating to "english", and a rating of 1 to their local english dialect, e.g. "english/uk". Second languages would return a rather lower value, and languages of which one has no understanding would return zero. This use of weights has a similar effect to the explicit weightings found in some other approaches, although the mechanism is more general.

A discussion of the use of utility functions across multiple media dimensions to control QoS adaptation is described in [11]. The model they describe concentrates on providing QoS for a video stream. Our approach contrasts with this in concentrating on multi-element media. In our case we have two dimensions for trade-off: we may select sub-sets of information to be represented, based on the semantic content of the information; and also between different representations of that information, based on the perceived utility of the representations.

The variant may take a utility function on each parameter understood. Specifications may not always match all parameters. Where variant handling of parameters, or utility function specification for a parameter, or meta data for a parameter are not available, that parameter is ignored. The following information is used to calculate the utility of a variant:

- Parameter utility limits: at what point do changes make no perceived difference (as good as it can be, or too bad to be worth considering). From these values, the correlation between value and utility may be inferred.
- Utility / parameter function rate: how does utility vary with respect to the parameter's value.

The utility ratings for the various parameters are combined to derive a final overall **intrinsic utility**, u_i, score for the variant. As each parameter utility value, $u(p)$, has been normalised to between zero and one, we simply multiply all utility values together for each variant. The use of a product function has the benefit that any variant which is unacceptable in some aspect (zero utility) registers as a wholly unacceptable variant. The comparison may need adjustment when comparing between encoding formats which have different numbers of parameters. The effect of successive multiplications will be to degrade the utility reported in comparison to another encoding format for which fewer parameters have utility functions defined. We suggest that some value, say the mean of the various $u(p)$ values, is taken and used to provide a comparable number of parameter utilities for the product.

The utility rating is then scaled according to the media encoding format, w_m, and type of element, w_k. We call this the **adjusted utility** for the parameter (see below).The adjustment allows for preferences to be described between forms of representation and parts of a structured document. Note that any change in the utility function due to encoding format and type still results in a utility function returning between zero and one.

$$u_i = w_m w_k \prod_{\forall \text{parameters, p}} u(p)$$

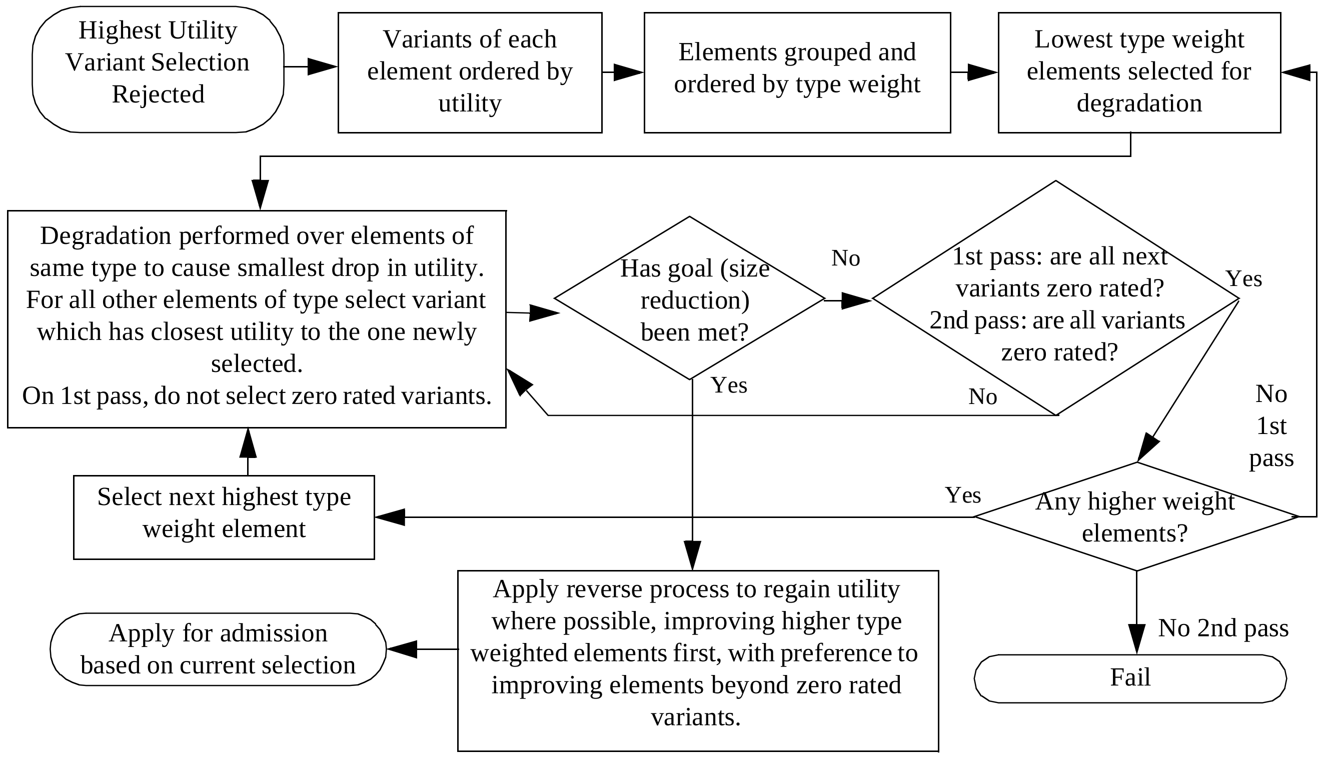
3.5 Selection Process

We require that our system choose between an arbitrary number of alternative representations of a compound document, each element of which may vary, e.g. a map page may consist of roads, contours, water features, etc. Each of these may be built up from basic information and additional detail. It is likely that more than one of the offered variants for any element would be *acceptable* in principle.

It is, however, desirable to provide the best possible presentation, which would be context dependant. The selected variants should maximise overall utility, and present data in a consistent and predictable manner. In addition, other goals may have to be met, such as a deadline for the map download. We may derive the deadline from the estimated utility of the data: how “interesting” it is may determine how long we are prepared to wait for it to download.

Initially the highest utility variant representing each element is selected. Where multiple variants have the same utility the one which makes the smallest resource load is chosen. The selection function makes a note of the selections on the meta data graph, and provides a summary of the selected variants. From this selection the total resource requirements are calculated, and requested. The resource manager’s admission function will take the resource requirements and goals (e.g., deadline) and test its resource model to see if the request can be admitted. If the admission request is accepted then the reservation model is updated, and the data requested. If rejected the resource shortfall is returned. The shortfall is incorporated into the goals, e.g., reduce data volume by 10kB.

If admission is denied, a search is then performed to satisfy the additional goal, while maintaining quality. For each element, its variants are ordered by descending utility. Where utility is zero, the variant is considered to have no value, and so is omitted from the selection entirely. Where an element has no variant any elements it includes are also omitted. Where contained elements which would be lost have a higher type weighting, the containing element is promoted, e.g., where road labels have a higher weighting than roads, the roads will be promoted to the weighting of the labels rather than be lost (degraded to zero utility). The roads and labels may be removed together if the degradation algorithm progresses to degrading the labels. It is likely that multiple solutions to the goal will exist, consisting of differing variant selections over the various elements. Our selection algorithm is designed to embody the following aims: To degrade the least interesting elements rather than more interesting ones; to maintain consistency between the presentation of elements of the same type; to incorporate utility due to format and parameters in the selection; to avoid loss of elements where possible; to cease degradation once the goals are met and then perform a test for potential improvements to regain lost utility, improving higher weighted elements first, and reversing element loss by preference. The algorithm is illustrated in figure 6.



Highest Utility Variant Selection Rejected
Variants of each element ordered by utility
Elements grouped and ordered by type weight
Lowest type weight elements selected for degradation
Degradation performed over elements of same type to cause smallest drop in utility. For all other elements of type select variant which has closest utility to the one newly selected. On 1st pass, do not select zero rated variants.
Has goal (size reduction) been met?
No
1st pass: are all next variants zero rated? 2nd pass: are all variants zero rated?
Yes
Yes
No
No 1st pass
Select next highest type weight element
Yes
Any higher weight elements?
Apply reverse process to regain utility where possible, improving higher type weighted elements first, with preference to improving elements beyond zero rated variants.
Apply for admission based on current selection
No 2nd pass
Fail

Figure 6: Selection Algorithm

While this approach will not (in general) result in the optimal use of available resources, it does seek to limit unnecessary adaptation. By grouping elements by type, degrading the least important elements by preference, and reinstating utility to the more important elements by preference once resource requirements are met, user satisfaction should be maintained, and consistent behaviour experienced. A more in-depth analysis of selection algorithms for resource management is presented in [12]. An illustration of the use of our selection technique is presented in section 4.

3.6 Resource Management and Monitoring

For wireless networks, where the underlying QoS is highly variable, monitoring feeds into the network resource models. As a model of effective parameters is built up, the admission function can provide more accurate guarantees. These models of resources will never provide a guarantee of resource parameters. However, by improving the accuracy, application level QoS management can improve the predictability of its behaviour.

Our resource management is provided with a simple time-sliced model. We have only implemented bandwidth management to date. Each time slice (100ms was used in the tests described here) has a capacity and one or more reservations associated with it. Resources are applied for as a data size, start time and end time. If the data / time is greater than the model of available bandwidth, the size difference required to satisfy this basic constraint is calculated, and returned with the rejection. Otherwise reservations are applied to each time slot in turn. Any shortfall, for instance where some other request already has reservations in that slot, is divided over the remaining slots. If the request can be fitted into the model, admission is granted. Otherwise the shortfall is returned, and the prospective reservation removed. At this time we have not implemented any interfacing from this application level model into lower level resource models, or network QoS control mechanisms. The technique presented relies on sufficiently large data loads to make accurate measurements. Further accuracy could be gained by making use of the measurements commonly performed within the network protocol stack, however our results are satisfactory at this time.

Resource monitoring is provided by the application. Each request (element request or meta data request) is supported by a thread which handles the connection establishment, reading from the buffer and timing. Once a minimum size of buffered data (or end of file) has been reached, the buffer is passed to the application. In the case of map data partial data may be parsed during data loading. Any data which the parser cannot use is pushed back to the loader. The application measures round trip time and effective bandwidth for the meta data load and the full collection of elements to be loaded. The round trip time is measured as:

$$\text{rtt} = (\text{1st reply time} - \text{1st request time}) - \left(\frac{\text{1st reply size}}{\text{bandwidth}}\right)$$

Request and reply times are measured as the application calls the HTTP API’s “get” method, and as the data is read from the buffer. The reply size is considered in calculating the round trip time, as a significant volume of data may be delivered in the first reply to the application. As a multi-part document may be loaded, the interaction of the various requests is hidden by considering the whole of a group of requests. The use of first request, first reply and last reply achieves this combination. The bandwidth is measured as:

$$\text{bw} = \frac{\text{data size} - \text{1st reply size}}{\text{last reply time} - \text{1st reply time}}$$

These values are reported to the resource manager, and the models updated. Firstly, a decaying average model is updated at a given ratio of new to old value. Typically we use 10 to 20% new value. Secondly, a recent history is updated. We typically keep the values of the last four loads. A recent average is calculated. If this is found to be a long way from the decaying average value the decaying average is set to this average value. The difference might be less than one half or over double. This aims to provide a faster response to large changes in monitored values, or a large deviation from some initial assumption. We do not present tests of the responsiveness of this monitoring mechanism here.

4 EXAMPLE MAP SEGMENT LOAD

We shall now illustrate the application of these techniques in loading a map segment. The interaction is similar to that described in figure 2. The full range of data and structuring is necessarily reduced to aid comprehension. We shall consider four types of feature: roads, driving restrictions (one-way systems, turning prohibitions etc.), traffic conditions, and contours. Our user assigns “roads” a type weight of 1, driving restrictions and traffic conditions a type weight of 0.8, and contours a type weight of 0. Utility functions over detail indicate a preference for map information detail at between 1:50000 and 1:250000 scale. For driving restrictions this function is overridden to indicate a preference for detail between 1:2500 and 1:25000, as small restrictions still affect the route taken. In contrast to the function applied to traffic conditions, where a small queue is not so significant. Outside the stated preferences the functions’ value falls off gradually.

The location service, taking information from a GPS receiver, and the route plan, indicates to the application an area of map which is expected to contain the driver’s location in two minutes. The application confirms that it does not already hold a recent copy of the requested data, and passes the request to its QoS manager. The QoS manager takes the request, and requests the meta data for that map segment from the map server, a second request to the local traffic service, and a third to the local council for up-to-date traffic restrictions, including planned road works.

The three responses are processed as the data arrives, to build the element - variant tree. The selection algorithm then walks over the tree, calculating the utility of each variant. For each element the variant with the highest utility is marked as selected. Contour data are all marked with a utility of 0, and so is not selected. Road information is offered at 1:10000, 1:50000 and 1:100000 for the area. 1:50000 returns the highest utility, and so is chosen. Driving restrictions are offered at 1:10000 and 1:50000, so 1:10000 is chosen. Traffic information is only offered in 1:10000, so is selected.

The resource requirements for these three data elements is 100kB. If five seconds have passed since the notification came in, and the map needed a minute before arrival in the area, then the request should therefore made for network bandwidth to download in 55 seconds from the current time. If the network connection available is only 1kB/s, the resource manager then returns a shortfall of 45kB to the QoS manager. A revision of the selection is then initiated.

Two degradation paths are built: one for road information with a type weight of 1, and one for restrictions and traffic data, with a type weight of 0.8. The less important type is degraded first. The driving restriction information is degraded to 1:50000 detail. The traffic data remains at 1:10000 as no other alternative is available. The shortfall is now 30kB. The next stage of degradation removes both elements, and so is passed over at this stage. The road data are treated next, and degraded to 1:100000. The data size is now 20kB. under the available resources. The goal has then been reached.

The 0.8 type weight information is tested to see if improvement is possible without breaking the goal. If it is found that it is, then the driving restriction is returned to its original level. The resource requirement would then be 50kB. On reapplication to the resource manager the request is accepted. A reservation is put on the network by the resource manager, and acceptance returned to the QoS manager. The QoS manager then returns to the application a list of the data variants to request. The application requests these from the appropriate sources, and displays the data returned.

One can imagine a second user, who is a cyclist, defining different preferences: Traffic information is of less concern, as traffic jams are easier to pass. However, hills have a greater impact, and so contour information would have a non-zero type weighting. A similar process would be followed to select the required data, but the resulting display would provide information relevant to that user.

5 TESTS AND RESULTS

5.1 Test Environment

We chose to simulate a series of network environments, rather than take results from actual network tests. While removing some sense of realism, it provides clearer results without a large scale deployment. Finding a range of "typical", or even stable, real network connections with characteristics which are well enough understood to enable discussion of results is difficult.

Our simulation environment used a 400MHz Pentium based system with 128MB RAM, running Linux and X as our platform. Sun's 1.3 beta JDK was used for implementing both the viewer and the simulation. Both the network simulation and application were run on the same system. The application and server load were sufficiently light-weight that no significant interference occurred. This also shielded the results from any variation due to network connections. There was only minimal further background load. The network simulation was based on controlling the playout rate of data from an HTTP server. We used a Tomcat servlet environment for the controlled web server. A get request for an element (or meta data) was taken, the data identified and an initial delay applied. Once the initial delay was satisfied the request was added to a "live requests" list. A single thread in the servlet environment ran a cycle of "delay, send data". Data was sent in "packets" of 512 bytes, or whatever data remained at the end of a variant. The requests were selected for playout in a round-robin fashion from the list. The data sent was the next section of the variant selected, so that data remained in-order. The inter-packet delay was modified according to any delay in the sending process to maintain an overall constant inter-packet delay. As a network simulation this technique is clearly simplistic; however, for the purposes of testing our system's ability to select data in situations with reduced bandwidth or long round trip times, it is sufficient. The delays were defined to millisecond resolution using Java threads, and so are minimum values. The opening of the HTTP connection and initial servlet processing also made the initial delay slightly longer than the setting given. Data rates are given in bytes per second here, as there is no explicit serial stream of data, or model of the protocol overheads usually encountered. The settings used as test cases are shown in table 1.

Table 1: Bandwidth Models

Model Name and Playout Rate (B/s)	Inter "Packet" Delay (ms)	Explicit Initial Delay (ms)
200,000	3	5
100,000	5	10
50,000	10	20
20,000	26	40
10,000	51	80
5000	102	160
2000	256	320
1000	512	640

For test data we used a single Ordnance Survey map tile. The tile was of a suburban area, and so contained a moderate volume of features. For each test it was loaded 10 times. The requirements given were based on an intuitive level of importance to a hiker, rather than detailed user studies. It is sufficient, therefore, that the specification caused the data to be divided up sufficiently to allow meaningful selection. The type specifications are given in table 2. We also present here the number of elements in the test data described by each specification, and the total size of the data for these elements. Note that our specification does not contrive to break the data into even-sized groups. The single tile data of the unmanaged viewer contained 301182 bytes. There is an overhead of 13809 bytes, due to repeated header data, where all the elements are loaded as separate data. The meta data was 5282 bytes.

Table 2: Experimental Data Set and Specification

Type Weight	Type Specified	No. of Elements	Total Size of Elements (Bytes)
.75	road	2	55820
.73	road-name	1	2189
.66	water	4	4922
.53	fence	1	92765
.50	building	3	65234
.41	topology	4	29537
.23	vegetation	6	7736
.10	survey	1	1053
.09	building-name	1	7594
.04	boundary	1	1386
.03	*	4	46755

We used an unmanaged viewer, based on the managed viewer as a control system. For each simulated network we used the managed viewer with a range of deadlines, and the unmanaged viewer. The deadlines used were a sub-set of 5,10,15,20,30,45, and 60 seconds.

We stopped performing tests with increasing times when we reliably loaded all elements at a lower time. We did not apply shorter deadlines where these deadlines had resulted in very low-quality selections (approaching failure to select any data) at a higher bandwidth model. In the results presented only the lowest one or two bandwidths tested for each deadline show data which we felt to have little actual value to a user.

The 10 loads at each bandwidth / deadline model were performed sequentially. For each new model the servlet environment was restarted, and an unmeasured load performed to allow it to initialise. At the start of each managed load the resource manager's history was cleared, and the bandwidth and round trip times set to the nominal bandwidth, and explicit delay in the simulation settings. While there was some adjustment to experienced rates and delays in the test series, the inclusion of the earlier tests does not significantly alter the results. The resource management and monitoring's ability to measure and adjust to change is not described here.

5.2 Deadline Satisfaction

We present here the results of the timings of map downloads in various conditions. It should be remembered that the goal of our selection algorithm was not optimum resource usage. The overall results are presented in figure 7, and a detailed view of these results for longer and shorter deadlines are presented in figure 9.

There are three key measures of our ability to meet deadlines in these results:

- For a given deadline, its result line should be below that time on the y axis.
- Where the unmanaged viewer can beat the deadline, the managed load time should be as close as possible to the unmanaged time.
- For the majority of cases, at a given deadline, the managed time should be below the unmanaged.

These three tests embody simple principles: firstly, if a deadline is to be specified it needs to be met most of the time to be meaningful. Times close to the deadline imply that the resources available are being taken advantage of, but are not a test of the quality of the selection. Secondly, while we expect the management to have some overhead, it should be minimised as far as possible. Thirdly, if management does not offer some advantage in the majority of cases, then its benefit is questionable.

In figure 7 we also note that the straightness of the line for the unmanaged browser indicates that the server is not meeting any limit in the system's ability to provide accurate inter-packet delays. At the 200kB/s simulation we are approaching two limits: firstly, the millisecond timing resolution of inter-packet delays, and secondly deadlines of sub-5 seconds start to become unrealistic for the class of applications we are considering. As can be seen from the results, we do in general satisfy these requirements. The variations from smooth lines seen in the timings, particularly at 2kB/s, can be attributed to the grainy nature of the selection algorithm and data set.

In summary, we find that for the volume of data involved, across the range of bandwidths modelled, the management technique is useful. The shorter deadlines are only applicable at higher bandwidths, and at higher network speeds the benefit of management is marginal. However, the overheads experienced are generally quite small, and the comparison of the managed download times to the unmanaged times at 20kB/s and below shows clear benefits. The question of how useful the data chosen are then presents itself, and shall be addressed below.

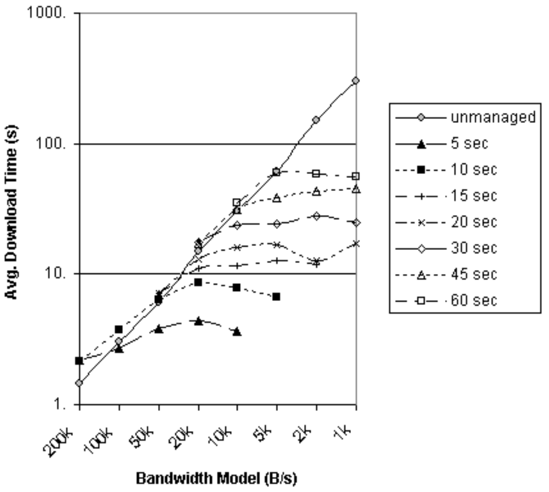


Figure 7: Download Time, Varying Bandwidth and Deadline

5.3 Data Utility

We now consider the quality of the results produced. Our application and specification tools lack the necessary user interface sophistication to make effective tools in a user study. However, by measuring factors in the selections made we believe we can draw some general conclusions about the quality of the data loaded. In the results presented some steps are seen, as for the timings. Again, these can be attributed in part to the grainy selection over the data set used. Repeating the tests over a number of map tiles, and with a more detailed specification would reduce some of these effects.

There are a variety of tests of data quality which might suggest themselves. Others such as [13,14] have used the premise that the volume of data corresponds to overall utility. It is true that there has to be some correlation, and for some data encoding formats they are more directly related than here. In general 10kB of data will provide a greater volume of information than 1kB, and where these represent to variants of some element it is likely that the 10kB version will have greater utility.

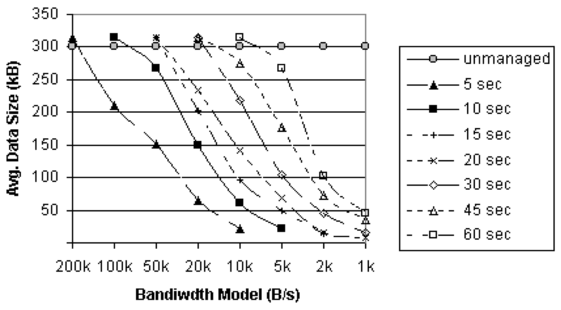
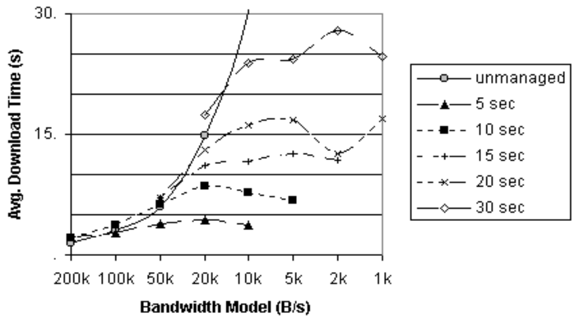


Figure 8: Total Data Size for Varying Bandwidth and Deadline



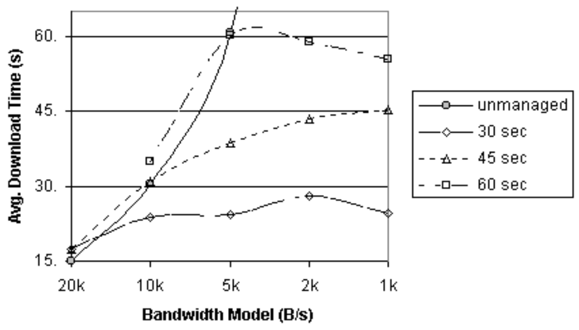


Figure 9: Average Download Time for Varying Bandwidth and Deadline, Detail

With reference to table 2 we can see that for any given rating of importance there are data elements small and large, at each end of the importance scale. However, the utility imparted to the user by 1kB of uninteresting data is unlikely to equal that imparted by 1kB of interesting data. The graph of data volume decaying with deadline and bandwidth model in figure 8 therefore illustrates successful selection to meet deadlines rather than giving any great insight into the quality of the data presented. We also see the overhead in terms of data volume, by comparing the total size of the unmanaged data with that of the complete loads under the managed presentation.

We then examined the selections in detail, to observe the effects of the different sized data and the specifications on the data presented. We present in figure 10 a binary selection indication for the different specification type weights. The results are for a typical selection for each deadline in the 10kB/s simulation. At this data rate we observed full selection only at 60s, and a progression through to a marginal selection at 5 and 10s. The results shown therefore cover the full range of overall presentation qualities. This representation is also rather easier to comprehend and make comparisons with than 7 small screenshots. From these results we see that the selection algorithm is behaving as expected: selecting higher type weight data in preference to lower weight data. These results are generally meaningful only in correlation with the specifications, and require a view of the data to fully understand (see tables 1 and 2). The variants for fence, building, and topology contain large volumes of data, which explains the jump observed around the selection of the corresponding elements.

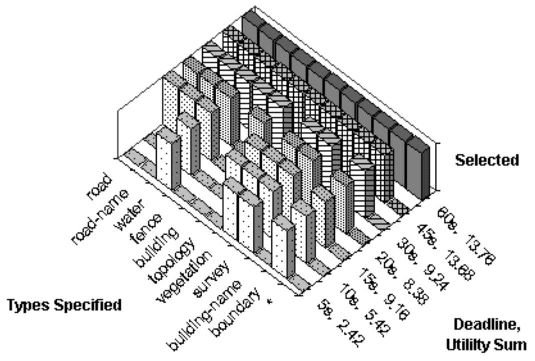


Figure 10: Example Selections for 10kB/s Model

We see in these results that the less important types are quickly dispensed with. However, the most important types – roads, road names and water – all remain present in all but the final two stages. This loss corresponds to our experimental procedure of abandoning a deadline once the results became too degraded to be useful. We see that the 3rd, 7th, 8th and 10th (by type weight) elements are always included, being relatively small. In most cases, treating at most one type-level omission within the first three type levels as acceptable, we find that any given deadline from our range gives a selection of all (or nearly all) features at some point in the bandwidth range, and at least two degraded but useful results. For longer deadlines the complete result extends to higher bandwidths, for a sub 5% overhead in download time. So, each deadline is valid over at least one order of magnitude bandwidth model. While the unmanaged viewer always gives a complete result, at any modem rate connection (sub 5kB/s) the time taken to download exceeds all our tested deadlines.

Our approach differs from mathematical or data encoding based compression to achieve QoS management. Data size may in some way correlate to presentation quality. However, the quality is still acceptable (by our definition above) after the size has fallen to 30 to 50% of that at the complete presentation. An algorithm which degraded without regard to the impact on the user would, in general, loose quality even faster with respect to data volume, by indiscriminate degradation. For instance, for JPEG of sample maps degraded from a quality factor of 80 to one of 50 (approaching the limit where detail remained acceptable), the data size was still 60 to 70% of the higher quality version.

6 RELATED WORK

There has been much work on low level QoS management and media presentation which space precludes us from discussing in detail. We discuss below some other approaches to the issues in providing adaptation for information retrieval applications.

6.1 Meta Data and Adaptation

Unlike many other approaches, e.g. [7,9,14,15], no quality rating is provided by the authors of the data or meta data in our approach. The meta data provide only an objective description of the data. The QoS management applies its own utility rating to the data. Our approach places a burden of complexity on the user to specify their QoS requirements in detail. However, we believe that this complexity may be hidden from the average user through the use of parameterised specifications authored by specialists for particular classes of users and tasks. The content provider's loss of control is balanced by their ability to specify the type of data represented (which is often the underlying theme in justifications for abstract quality ratings by the provider), and to limit the variants provided. The benefit is in enabling users of diverse devices with diverse interests to access data, for a small overhead in the data provision process. The effort required to produce meta-data is significantly lower than that required to hand-tailor data.

[15] includes the use of a "role" in the context of adapting web pages, along with an "importance value". They suggest that the client may use these in allocating resource limits for data volume and screen area. Their motivation is the ability to describe data for modification (by proxies) to provide for display on restricted devices, in particular by splitting web pages into sub-pages. Our approach is more general, in that the relationships between the elements of the maps are not generally a matter of sequence which need to be satisfied in their display. Also, the rating of the elements is not included in the meta data. The issues of sequence discussed in work such as [15,16,17] is necessary to consider when applying these techniques to other application domains and media encoding formats.

6.2 Negotiation and Selection Techniques

The approach described supports a more general mechanism for selecting "important" data than the layered approach often adopted in media scaling, e.g., MPEG, and [18]. By addressing structured data where different semantic data or levels of representation are explicitly separated out, the improvement / degradation path can offer rather more choice than a sequence of layers. In the case of maps, it is also possible that a layer definition would not suit all users of the map.

One popular approach to limited resources is connection to the wider networked world through some transcoding mechanism, e.g. [6]. These commonly provide a fixed adaptation to known device characteristics. This approach is included in our model, through the use of intermediate nodes which advertise their capability through meta data. The advertisement has the advantage that use is not tied to particular classes of devices, and does not require separate effort to produce documents tailored to specific devices. This approach facilitates the use of emergent devices, while allowing specialised data to be used where the effort has been taken to provide it.

There is work on content selection being undertaken in various standards [7,9,8]. These approaches generally describe the hardware and software capabilities of devices, and limited preferences of users. The user preferences catered for are primarily binary switches or exact matches, e.g. for selecting language. The suggested interactions for applications such as web browsing are that the client sends a minimal specification at the start of a session (with a given server). These techniques suffer from a potential scalability problem, in that the server must hold a description of all recent clients, and perform selection, in addition to delivering data. These protocols also seem limited with respect to allowing a rich description of user preference in addition to device limitations, or describing perceived quality over parameters which may take a wide range of values, although it provides a clear means to express limitations, particularly with respect to devices.

CMIF [16] and SMIL [17] describe temporal and spatial behaviour of a presentation, and have similar constructs for describing multiple variants of media elements in a structured presentation. The selection may be subject to a test on various parameters of the system in SMIL. CMIF focuses on providing alternative content for various contexts or user abilities, as defined by the author. In both cases, the result is that the selection of alternatives is limited by the authored selection support. Our approach takes similar approaches to whole document adaptation, and supporting a range of alternatives for elements. However, we believe that our method offers a more flexible approach for emerging system and user classes.

Our approach to selecting data according to its properties has similarities to the approach in [13], based on a notion of abstraction. They describe an interplay of willingness to degrade data against urgency. Their approach assumes that the quality of the data can be inferred from its size. While this relationship may hold in the case of images, it is not true for all data. We do not regard the selection process as being solely about degrading data (a common starting point, e.g. [5,6]). The total data about a geographic area is likely to be overwhelming and contain much irrelevant information, in which case it is not the case that omission constitutes a degradation. Similarly we measure utility across the user's perception, rather than in relation to some ideal or original version, such as in [14].

6.3 Resource Management and Response to Context

We are interested in soft QoS management, under conditions where hard guarantees may be unrealistic. We also make no claims that our model of resource management can offer a firm guarantee of resource characteristics. However, in the context of highly variable resource characteristics a weak model of resource management and negotiation offers a practical solution, as long as the user and application designer are aware of its weakness. In [13] an overview of approaches to adaptability is presented, and a suggestion for linking perceived quality to deadline specification for web browsing is described. Their approach to deciding how to adapt is based on allocating percentages of the data the user is willing to use, for a given element. We believe our approach of rating the utility of variants by measurable parameters provides a finer degree of control than their assumption that any given proportion of data volume corresponds to a known (and acceptable) degradation in quality.

7 CONCLUSIONS

In this paper we have presented techniques designed to enable application level QoS management for presentation of structured data. This management is achieved by selection (among multiple variants) of the most appropriate form of each element in a multimedia presentation. We support structured data, where elements may be both content and containers for subordinate data. The integration of context-dependant specifications and resource management are crucial to support mobile users and ubiquitous computing. The treatment of the entire data to be presented at the point of media selection and resource negotiation supports an application aware approach to QoS management. By treating whole documents, full consideration can be given to the impact of different elements on resource use, and selection made with awareness of all available trade-offs. This "whole document" approach enables selection in a consistent, user-friendly manner. We maintain a separation between measurable parameters and how they affect the user's perception of the utility of the presented data. This separation reduces the burden arising from authorship of meta data or behaviour specifications, in contrast to many others' approaches. At the same time it enables user controlled specification of preferences.

Our use of the description, at an abstract level, of the type of data, in addition to its encoding format and measurable properties, without the use of author rating of content, is unusual in the field of application QoS management and context awareness. In particular, we are using the type both to derive a weight *and* to select between utility functions, which gives a greater degree of control than a simple weighting or preference indication alone. Type preference (indicated by a normalised weight) is used to ensure consistency between similar data, and differentiation due to user's interest, as well as due to the data's parameters. The combination of these numerical values offers a greater expressive power than the more common boolean operators.

The type factor is particularly useful for applications such as maps, where the feature classifications provide a clear basis for type definition in common terms to the user. The user of a map can also identify distinct tasks and the relevance of the types of data to those tasks. As the data encoding formats become more general, or the user task less clearly defined, the ability to express clear type preferences becomes more complex, and is left for future investigations. We believe that as data formats become richer, the techniques described will become widely applicable in the field of multimedia applications. We include data retrieval applications in this definition, as well as streamed media applications, which are more often the subject of QoS research.

Our use of resource management treats the wide variation in network interfaces available on mobile systems. We acknowledge that strict deadline satisfaction with wireless networks is generally impossible, however a soft deadline is acceptable for the example application. The use of an abstraction for resource reservation supports the use of lower-level network QoS approaches such as RSVP.

Our initial test results indicate that the selection process and application level resource management enable deadline satisfaction across a wide range of bandwidths. The degradation of data takes into account user preference, and enables a useful presentation across a significant range of bandwidths and deadlines.

7.1 Future Work

As discussed, the work presented here is defined in terms of adapting vector maps. As the nature of data on the WWW becomes more highly structured, data production techniques more sophisticated, and meta data more readily available these techniques will become more applicable to the wider world of the Web. There are many issues to be resolved in this area. Our current interests include:

- Extending our map application to provide a fuller test-bed for our ideas. We have undertaken some work on using these techniques to adapt the presentation web pages. The integration of this, and other adaptive multimedia techniques to provide a multimedia map application is one goal of our work. In particular we are interested in integrating hyperlink data with the map data.
- The use of proxies and multiple data sources presents some interesting problems in selecting configurations of proxies, and in managing the view of the available options at the selection point. The use of intelligent caching, transcoding and dynamic data integration proxies also provides for a wider variety of selection options.
- The definition of easily extensible specifications, which may be varied according to context is necessary. This includes work on abstracting the specifications for the user, integrating partial specifications, and work on defining type vocabularies. Where media continue to contain much data within one file, such as images, the ability to describe an element as having a compound type is also necessary.
- Some user studies, and real-world testing will be undertaken, once the development of the application and user interface has progressed further.

8 ACKNOWLEDGEMENTS

We gratefully acknowledge financial support from the EPSRC (Grant GR/M 26428 "Link-me" project).

REFERENCES

1. M. Weiser, "Some Computer Science Issues in Ubiquitous Computing," *Communications of the ACM*, vol. 36, pp. 75-85, 1993.
2. B. N. Schilit, N. Adams, and R. Want, "Context-Aware Computing Applications," presented at Workshop on Mobile Computing Systems and Applications, 1994.
3. J. Gecsei, "Adaptation in Distributed Multimedia Systems," *IEEE Multimedia*, pp. 58-65, 1997.
4. D. Chalmers and M. S. Sloman, "A Survey of Quality of Service in Mobile Computing Environments," *IEEE Communications Surveys*, 1999.
5. B. D. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. R. Walker, "Agile Application-Aware Adaptation for Mobility," presented at 16th Symp. on Operating System Principles (SOSP-16), Saint-Malo, France, 1997.
6. 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*, 1998.
7. K. Holtman and A. Mutz, "Transparent Content Negotiation in HTTP," IETF RFC 2295, 1998.
8. W3C, "Composite Capability / Preference Profiles (CC/PP): A User Side Framework for Content Negotiation," W3C NOTE-CCPP-19990727, 27th July 1999.
9. G. Klyne, "A Syntax for Describing Media Feature Sets," IETF RFC 2533, 1999.
10. Dublin Core Metadata Initiative *http://purl.org/dc*
11. J. Walpole, C. Krasic, L. Liu, D. Maier, C. Pu, D. McNamee, and D. Steere, "Quality of Service for Multimedia Database Systems," presented at Data Semantics (DS-8), Rotorua, NZ, 1999.
12. C. Lee, J. Lehoczky, R. Rajkumar, and D. Siewiorek, "On Quality of Service Optimization with Discrete QoS Options," presented at IEEE Real-time Technology and Applications Symposium, 1999.
13. W.-Y. Ma, I. Bedner, G. Chang, A. Kuchinsky, and H. J. Zhang, "A Framework for Adaptive Content Delivery in Heterogeneous Network Environments," presented at Multimedia Computing and Networking (MMCN00), San Jose, California, USA, 2000.

14. R. Mohan, J. R. Smith, and C.-S. Li, “Adapting Multimedia Internet Content for Universal Access,” *IEEE Trans. Multimedia*, vol. 1, pp. 104-114, 1999.

15. Hori, M., Kondoh, G., Ono, K., Hirose, S., Singhal, S., “Annotation-Based Web Content Transcoding,” presented at WWW-9, Amsterdam, Holland, May 2000.

16. D. C. A. Bulterman, “User-Centered Abstractions for Adaptive Hypermedia Presentations,” presented at ACM Multimedia’98, Bristol, UK, 1998.

17. W3C, “Synchronized Multimedia Integration Language (SMIL) 1.0 Specification,” W3C REC-smil-19980615, 15th June 1998.

18. T. Ye, H. A. Jacobsen, and R. Katz, “Mobile awareness in a wide area wireless network of info-stations,” presented at MobiCom’98, Dallas, TX, USA, 1998.