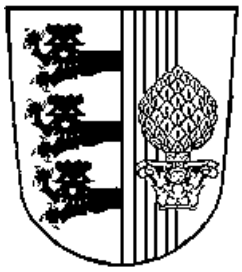
A situation-aware mobile traffic information system prototype

Wolf-Tilo Balke, Werner Kießling, Christoph Unbehend

Angaben zur Veröffentlichung / Publication details:

Balke, Wolf-Tilo, Werner Kießling, and Christoph Unbehend. 2002. "A situation-aware mobile traffic information system prototype." Augsburg: Universität Augsburg.

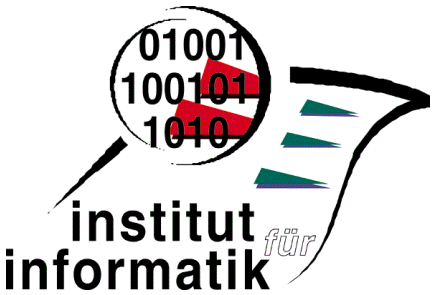
Universität Augsburg



A situation-aware mobile traffic information system prototype

Wolf-Tilo Balke, Werner Kießling, Christoph Unbehend

Report 2002-14 August 2002



Institut für Informatik

D – 86135 Augsburg

A situation-aware mobile traffic information system prototype

W.-T. Balke **W. Kießling** **C. Unbehend**

Institut für Informatik,
University of Augsburg
Augsburg, Germany
{balke, kiessling, unbehend}@informatik.uni-augsburg.de

Abstract

With the current evolution of mobile communication and Internet computing mobile services will play an increasing role in future work and private life. Especially enabling mobility in urban and populous areas needs tools for individual traffic planning. In this paper we present a prototype of a traffic information system enabling not only plain route planning, but also advanced services like traffic jam alerting, etc. Using a centralized database architecture our service integrates a location-based service with a situation-aware traffic information system. The fast changing nature of traffic data requires that traffic information is gathered from a variety of on-line Internet sources featuring traffic jam, weather, or road work information. Our asynchronous update strategy of the central service database allows to meet real-time requirements though providing up to date information. In the course of this paper we will investigate all necessary capabilities, present the development of our prototypical system, and give a case study on applying our prototype for real world use. Modern XML technology together with appropriate XSLT stylesheets allows the automatic conversion of generic traffic information for the delivery to a variety of mobile devices. In summary we believe that intelligently joining the latest powerful technologies for preference modeling, database-driven evaluation and standard XML technologies today promises a breakthrough for mobile information systems.

1. Introduction

The recent developments in mobile communication and Internet computing have paved the way for a wide variety of applications. Granting pervasive access to all kinds of data mobile services will essentially help to support the unrestricted mobility that is inherent in today's way of life. In urban and populous areas traffic information services especially for car dependent people, e.g. commuters, are crucial. Due to the development of GPS and navigation computers within cars those services can improve route planning and accompany the whole journey with up-to-date information. However, the traffic situation may undergo drastic changes during longer journeys, traffic jams may evolve or dissolve, the weather may change drastically, etc. Thus a wireless service is essential to provide the necessary information. Our intended situation-aware mobile service comprises:

- route planning
- alerting for traffic jams
- alternative routes

In the course of this paper we will discuss at sufficient detail all technological issues required for the successful design and development of such a complex prototype system. Reviewing most recent advances of preference modeling techniques and personalization of multi-attribute search engines, we give a specification of mobile user preferences and situation awareness, given the available on-line information sources. For efficient implementation of the resulting complex preference queries we show how to employ the novel SR-Combine algorithm. SR-Combine is a specifically tuned algorithm for efficiently evaluating ranked preference queries under the top-k query model.
We present an extensive case study of our prototype for real world use. As sample route databases we have chosen a map of the complex autobahn system from the German Ruhrgebiet area containing more than 100,000 different routes. The Internet sources selected are traffic information from a local radio station[1], road works from the German ministry[2], and local weather data[3]. Data from all these distributed sources is integrated and used to recommend routes for each specific user according to his or her personal preferences.
Once the routes are selected standard XML technology – including XSLT – achieves the automatic conversion of traffic information in generic XML formats for the delivery to a variety of mobile devices. As mobile device in our case study we selected i-mode cellular phones, but of course any other mobile client device can be used (WAP-phones, PDA, handhelds,…).
Performance benchmarks of our service yielded very encouraging results. We show that personalized mobile and situation-aware traffic information services can already be provided in real-time, being 3 seconds or less [Pop97].
Our paper is organized as follows: we focus on the modeling and evaluation of user preferences in the area of traffic information systems in section 2. Architectural issues will be discussed in depth within section 3. Section 4 presents an extensive case study with performance benchmarks. We conclude this work with a short summary and outlook.

2. Personalization Issues

Personalization in tasks like route planning is a difficult matter. Depending on the intentions of the traveler specific information has to be used within the retrieval process. Some of the static web sites already use simple information for a more personalized route planning. [Rou02] for instance offers optimizations such as shortest routes or fastest routes depending on the speed of your car (slow, medium, fast), but of course this does not lead to a satisfying personalization.
When considering personalization for complex web services the user's expectation has to be broken down to the single information sources. The length of alternative routes is such an expectation, but would a user really prefer a route with many road works to a slightly longer route with no road works ahead? In the following we will closely examine the nature of preferences for our kind of service. We will present an adequate preference model, identify the preferences needed for our service and last, but not least focus on efficient preference evaluation techniques.

2.1. The Preference Model

To get to a intuitive preference model the nature of human preferences has to be taken into account. Intuitively people express their wishes in the form "I like A better than B". Since this can be understood by everybody, it is a natural way of modeling user preferences. For in-

[1] www.wdr.de/epg/verkehr/vkdocs/vt_main.htm
[2] www.baunetz.de/bmvbw/verkehr/bab/karte01.htm
[3] www.wetter.de

stance anybody would prefer a dry road to a wet and slippery one. Mathematically such a preference semantics can be characterized by strict partial orders [Kie02, Cho02].
Preferences can be engineered inductively as follows. Depending on the application domain, a set of pre-defined base preferences are assumed to be defined. These base preferences are either domain dependent like our dry or wet road, or depend on the user. For instance a user might care less about the weather conditions than about traffic jams. Given these single preferences, more complex ones can be gained by means of three fundamental operators.Complex preference constructors are:

- Pareto preference: $P := P_1 \otimes P_2 \otimes ... \otimes P_n$
  P is a combination of equally important preferences $P_1,\ldots, P_n$, following the well-known Pareto principle.
- Prioritized preference: $P := P_1 \,\&\, P_2 \,\&\, ... \,\&\, P_n$
  P evaluates more important preferences earlier, similar to a lexicographical ordering. P_1 is most important, P_2 next, etc.
- Ranked preference: $P := rank_F(P_1, P_2, ..., P_n)$
  P combines individual numerical scores $f(P_i)$, $1 \leq i \leq n$, by means of some monotonic ranking function F. Input for ranked preferences are always score preferences f(x) that assign score values to each database object. For instance given score preferences $P1 = f_1(x)$ and $P2 = f_2(x)$ then:

$$a <_{rankF(P1, P2)} b \quad \text{iff} \quad F(f_1(a), f_2(a)) < F(f_1(b), f_2(b))$$

These combined preferences again are strict partial orders, i.e. preferences. Some fundamental algebraic properties of such strict partial order preferences are given in [Kie02]. In the following section we will have a closer look on the nature of the traffic data to choose the right preference constructors.

2.2. Preference Engineering for Route Planning

Building a mobile real-time traffic information system the integration of data from various sources plays an important role. Due to the nature of traffic data the current situation may drastically change within a few moments. Consider for instance a typical interaction like route planning: a user wants to go from city A to city B and has a number of routes to choose from. To give but a few characteristics of such routes they may differ in

- length
- traffic jams
- road works
- weather conditions

Though for efficiency reasons it is definitely preferable to travel short routes, for safety reasons it is advisable to avoid traffic jams or bad weather conditions like e.g. fog on some routes. Thus we have a typical *multi-attribute problem*. And what is more, once the user has decided which route to take, the situation can drastically change by e.g. an accident or some other unforeseen event. In these cases a notification that the situation has changed together with a recommendation for an adequate detour would be desirable.
Considering traffic data we find that almost all characteristics are of a typical quantitative nature. For instance a user would prefer a shorter route to a longer route and the shorter the route the more adverse factors one would accept. This also applies to the length of traffic jams or road works. Thus we will use ranked preferences with an adequate scoring functions f and a ranking function F that can be changed according to the requirements.
But first we have to look at the route classification and how to get a scoring for each route. The scoring of route lengths is straightforward. Alternative routes are ranked according to their length and normalized score values are assigned to each route, i.e. the shortest route has a score of 1.0 and routes are given in descending order up to the longest route with a score of

0 (we always assume that routes do not contain cycles and the longest acceptable route may be up to 50% longer than the shortest one). Using the difference between shortest and longest route the score values are normalized; thus routes of similar length get similar scores:
$f_{length}(x)$= (length(longest_route) – length(x)) / (length(longest_route) - length(shortest_route))
Traffic jams and road works, however, are a little harder to manage. First of all we can use the information about the number of jams/road works and their totalized lengths. Secondly we distinguish between two types of jams and road works:

- jams with standstill or stop and go traffic
- road works with closed lanes and those without closed lanes

Standstill jams and road works with closed lanes are severe obstructions and will thus be assigned double lengths. Again we can normalize these values to scores from 1.0 down to 0. In case of a complete road blocking the total score is automatically set to 0 (f_{road_works} analogous):

$f_{traffic\_jams}(x)$ = q * (1- (Σ (p * length_of_jam(x)) - length_of_jam(route_with_shortest_jams)) / (length_of_jam(route_with_maximum jams) – length_of_jam(route_with_shortest_jams)))

with q = 0, if a total blocking appears, else q = 1, and p = 2 for standstill traffic, else p = 1.
For the weather information we used a compound score consisting of average temperatures, rainfall, wind and humidity. Considering the temperature a too low or too high temperature is less desirable, thus we are assigning a score of 1.0 to all temperatures between 10-20°C and interpolate linearly down to –5°C and up to 40°C with a score of 0. For the rainfall we interpolate linearly from dry weather 1.0 to rainfall up to 1 l/m² with a score of 0. The same applies to wind we interpolate from a velocity of 0 km/h as 1.0 down to 75 km/h having a score of 0 (gale). For the humidity we again interpolate from 1.0 at 0 % humidity down to a score of 0 for humidity over 100 %.
All the weather data than is numerically integrated in a manner that respects the requirements of traffic especially the dangers of black ice, i.e. temperature and rainfall are more important than wind and humidity. As ranking function to aggregate a total weather preference $F_{weather}(x)$ for each route x we use:

$$F_{weather}(x) = 1/6\ (2\ f_{avg\_temperature}(x) + 2\ f_{avg\_rainfall}(x) + f_{avg\_wind}(x) + f_{avg\_humidity}(x))$$

The function may differ for other parts of the world, but can of course be exchanged. We also implemented some specific rules for the area setting the score to 0 for black ice, whole gales and fog with visibilities below 15 meters.
To integrate all the factors we used *user defined weights* that can be assigned to each condition (cf. fig. 3, right-hand side). The user may distinguish three degrees of importance (important, medium, weak) that are mapped onto adequate weights for the combining function. Since we distinguish between four factors for each route, we get weights $x_1, \ldots, x_4$ that can be assigned values between 1 (weak) and 3 (important). Considering our example in fig. 3 the weights would be $(x_1, x_2, x_3, x_4) = (3, 3, 2, 1)$. Since a route's length is of big importance we always increase its factor by 1. The normalized ranking function for each route x thus consists of:

$$F(x) = ((x_1+1)\ f_{length}(x) + x_2\ 2\ f_{traffic\_jams}(x) + x_3\ f_{road\_works}(x) + x_4\ F_{weather}(x)) / (\Sigma\ x_i + 1)$$

As final preference we get $rank_F(f_{length}, f_{traffic\_jams}, f_{road\_works}, rank_{Fweather}(f_{avg\_temperature}, f_{avg\_rainfall}, f_{avg\_wind}, f_{avg\_humidity}))$, i.e. a route *a* is preferred to a route *b*, if $F(a) > F(b)$.

2.3. Preference Query Evaluation

As stated in the previous section our web service collects information in the form of ranked lists from different sources and integrates them efficiently using an adequate user profile that influences the combining function F. The problem of efficiently integrating several ranked lists to get some overall best objects has been addressed in the BMO query model and as "top

k retrieval" [Kie02, KK02]. Here we face a special problem of efficiently evaluating a preference query given a numerical preference $rank_F(P_1, P_2, P_3, P_4)$.
[Fag96] first posed this problem and stated a simple algorithm. In the subsequent years algorithms to solve the problem became more complex, but also more efficient leading to algorithms for applications in fuzzy logic [NR99], multimedia databases [OR+98, GBK00] and web services with different retrieval cost profiles [BGM02, BGK02]. For our implementation we used the *SR-Combine* approach [BGK02] that promises applicability and efficiency in top k retrieval task for central server architectures, but that also allows to use distributed architectures, if high bandwidths and high performance networks can be guaranteed [KB02].
The *SR-Combine* algorithm overcomes the disadvantages of previous algorithms by suitably self-adapting to a variety of environments and controlling run-time costs. For each source a so-called cost-ratio is assigned determining the relative costs for different kinds of accesses to that source. Thus by carefully balancing the access costs and the expected use of the access we can optimize the total run-time of our algorithm. This flexibility is useful especially in the case of distributed sources and thus paves the way to migration from central server architectures towards distributed services.
The algorithm *SR-Combine* has another advantage that is helpful for mobile services. It consists of three phases that successively retrieve k overall best objects. That means that first results can already be delivered or prepared for delivery while the algorithm is still running. Especially for mobile environments this is a major advantage due to the efficient use of the available bandwidths. Benchmarks in [BGK02] show that *SR-Combine* scales well and in most cases of practical applications meets the acceptable 3 second run-time barrier.

3. Architectural Issues and Prototype Implementation

Due to the real-time requirements previous approaches were mainly static web services for route planning that did not use any current information at all, e.g. [Rou02] or the OnStar system by General Motors [OnS97] that featured a satellite-based traffic navigation services with human operators that help on request. Current information was first integrated by agent-based systems gathering the information from distributed sources within complex CORBA architectures [YTT00] or even intelligently collecting floating car data like [MCM98]. A solution using database technologies offering all the advantages like scalability or transaction concepts up to now failed due to the real-time requirements. First approaches to utilize database systems, e.g. [BGM02] building a simple location-based restaurant service in a database environment, reported run-times in the range of hours for even simple tasks to complete. We will thus first have a look at mobile service architectures and then focus on our prototypical implementation and give an outlook on the migration to distributed information sources.

3.1. Mobile Services

With the current developments of flexible devices like cell phones or PDAs, mobile services will play an important role in future information technology. Pervasive access on information becomes more and more attractive not only for work, but also for private uses. We will illustrate this by three examples of different services and discuss their specific characteristics:

- Stock trading with mobile access to global stock exchanges, especially intra-day trading offered today by most on-line banks
- Location-based city guide or restaurant service to find nearest restaurants with respect to certain user-profiles [BGM02]
- Mobile on-line auctions involving multi-media data and providing typical usage patterns as mobile services, e.g. checking auction bids [WBK02]

service	complexity	updates	sources
stock trading	low	often	external
location-based	medium	seldom	mixed
mobile auctions	high	seldom	central

Table 1: Characteristics for mobile services

As shown in the table above the three scenarios differ in characteristics like update behavior and the nature of data sources. Whereas stock market information provides simple data types, but has update ranges of few seconds, the information for more complex services involving e.g. multimedia data, is somewhat more durable like for instance city maps for location-based services. Typical update intervals in these cases range between days and weeks. This means that e.g. stock market information has to be imported directly from content providers. But durable information like multimedia data can be transferred on central servers enabling more efficient storage or indexing schemes. The research area of data integration over the web [GB97, TSH01, GW00] has lead to twofold architectures for different applications:

- **Central Server Architecture (CSA):** If the service provider is also the content provider or handles mainly durable information, services are provided using a high performance server with central data repositories.
- **Distributed Sources Architecture (DSA):** If the service and content provider differ or short update ranges are necessary, services may be provided using a middleware gathering information on demand from distributed sources accessed via the Internet.

In [BGM02] ways to build mobile services with a DSA architecture providing direct access to various Internet sources are presented. Though these services (e.g. location based city maps or restaurant guides, etc.) would be desirable, tests show that the processing even for simple tasks often need hours. Thus when trying to meet real-time requirements in services today, we have to build our traffic information system on a CSA architecture. A solution of combining Internet sources with local database servers is given by the WSQ/DSQ approach [GW00] that handles direct accesses to Internet sources in an asynchronous manner and caches the results for later use in virtual tables of a central database server. Since the service provider in general will know what type of queries to expect, what data is commonly accessed and how often updates are needed to meet the service design, a caching strategy with asynchronous updates is most suitable for mobile services.

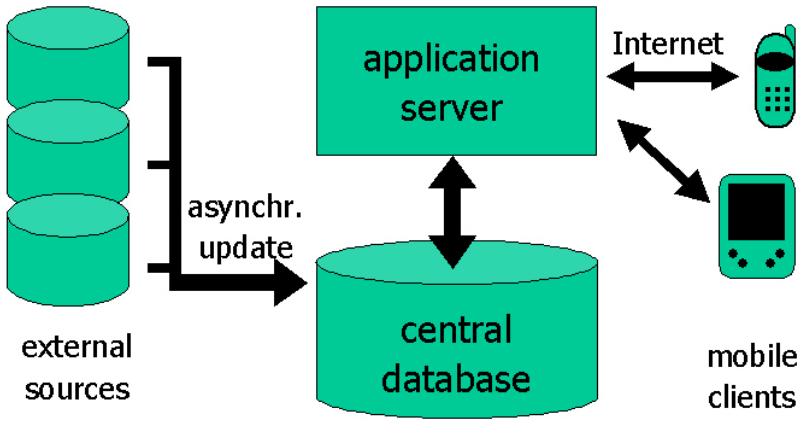


Fig. 1. Intended CSA architecture

The mobile service in Fig. 1 consists of a application server containing a combining engine for data integration according to the user-profile specified within the query. All the content is retrieved from a central database server (updated in an asynchronous manner). As shown in

[WBK01] for the example of mobile online auctions the delivery engine can automatically transform synthesized information in generic XML formats using XSLT to support mobile devices e.g. via WAP or i-mode gateways. For our prototype we will in the following consider the delivery to i-mode clients using compact HTML (cHTML [W3C02]), but of course we can also use any other format. Another advantage of this architecture is that data on the central server can be indexed to suit the service design. Through statistical analysis also costs for certain usage patterns can be estimated for different kinds of access personalized for each user.

3.2. Prototypical Implementation

To get an adequate service architecture for real-time information we will rely on a central server approach. Fig. 2 shows our service architecture. Please note that due to the flexibility of our SR-Combine approach we can straightforwardly adjust our service to distributed service architectures whenever the bandwidths and network performance allow real-time querying.

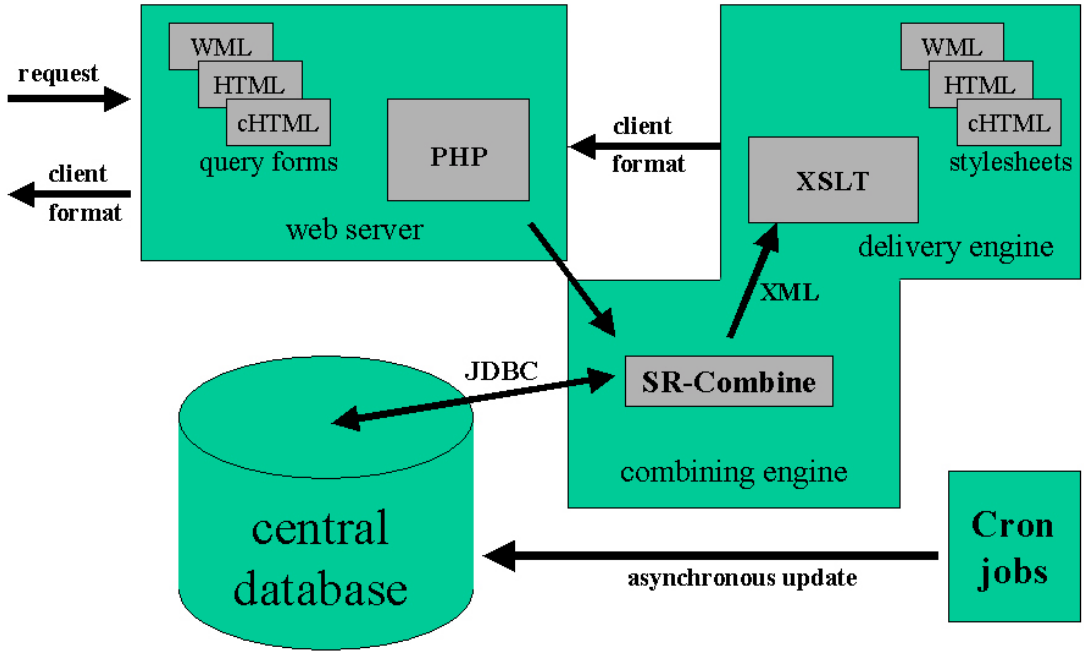


Fig. 2: Mobile service architecture

The service architecture is build around an IBM DB2 V7.2 Universal Database System that stores all different routes together with geographic, traffic and weather information. For the computation of routes within the database only the parts between adjacent cities are stored in tables. For queries on longer routes the transitive closure is build on demand from these smaller parts. This leads to an efficient storage of the routes, because we do not need to store over 100000 routes, but all sensible routes for each query are deduced with the restriction that no route is longer than 1.5 times the shortest route and no cycles appear. As mentioned before the database server is updated regularly in an asynchronous manner. In our UNIX system environments we used a set of adequate cron-jobs that supply the database with updates according to the update profile of the data source. For instance traffic jam information is far more frequently updated than road work information which tends to be more durable. In our current implementation we use update intervals of 1 minute updates for traffic information, 30 minute

updates for weather information and 6 hour updates for road works. Thus the update performance of the central database can be optimized.
On top of the database we use an application server that communicates via the Apache 1.3 web server whose capabilities are extended using the PHP 4 scripting language. At each service request we identify the browser type of the mobile client device and hand on the form data to our application server running the combining engine with the SR-Combine algorithm. The combining engine holds a connection to the central database and integrates the query results according to our user's preferences. When the final result is assembled the combining engine uses a generic XML format that can be transformed by the delivery engine to suit any registered client device.
For each registered device an appropriate stylesheet is kept by the delivery engine. Using the high-speed Xalan XSLT stylesheet processor the generic pages are automatically transformed and delivered according to the type of the user's browser. For example if a WAP cell phone is used the output will automatically be transformed into WML and for i-mode clients adequate cHTML pages are created. All pages contain the information needed to choose a route appropriate for the user. Having decided upon a rout the user can register with the service and subscribe to an alerting service for current changes of the traffic situation. This service is carried out via an SMS gateway and will be discussed in depth within section 4.

3.3. Seamless Migration to Distributed Architectures

In the previous sections we pointed out that today most architectures feasible of meeting real-time requirements when integrating data from various on-line sources are of the central server type. This means that data has to be collected and updated periodically. Obviously such a replication of information is not really desirable. For a start there are the expensive hardware costs when storing masses of data necessary for adequate qualities of service, the maintenance costs; network traffic by (often unnecessary) periodical updates, and last, but not least there may even be legal issues involved.
A better way would be to collect all necessary information (and only the necessary information) on demand and leave the keeping and updating of data to the content provider thus reducing network traffic and service costs. Unfortunately tests with distributed services have shown to be not yet feasible due to low bandwidths on insufficient networks. However, considering the architecture of our service and its main component the combining engine running *SR-Combine*, a distributed service is feasible with *almost no* changes. A seamless migration is enabled by the following features of *SR-Combine*:

- address different information sources in parallel
- optimize the accesses according to sources' cost-ratios
- deliver result objects successively as soon as they are found

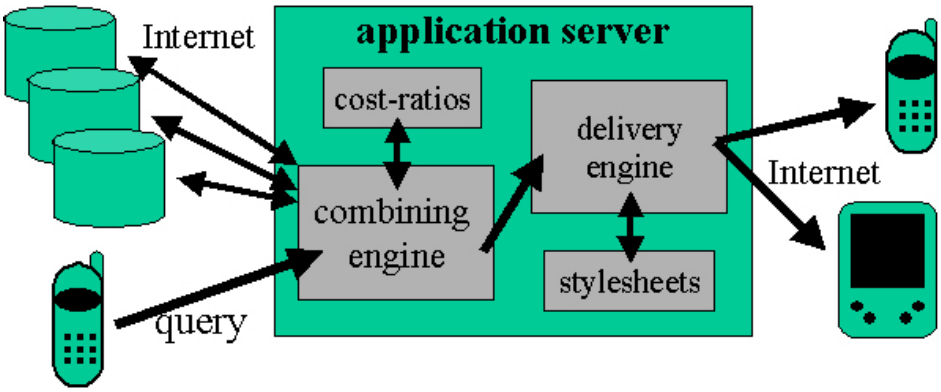


Fig. 3: Distributed service architecture

This leads to a simple architecture that can be used as soon as the current data transfer rates are competitive enough to meet real time environments. Figure 3 shows our intended distributed architecture. The service basically consists of an application server in which our two main components combining and delivery engine communicate with external sources/devices via the Internet. Once a query is posed, the combining engine decides with respect to the cost-ratios which sources to access. Expensive sources will only be accessed if necessary; thus run-times are controlled during the entire retrieval process. Every time a result object can be identified it is directly passed on to the delivery engine that can start to generate the adequate formats for the subsequent delivery. Thus also parts of the retrieval and XML generation/transformation can be performed in parallel.

4. Case Study and Performance Evaluation

4.1. Sample Application

In the following we will take a closer look on the typical interaction with our prototypical systems. Let's assume the motorway network in the Ruhrgebiet which is a particularly densly populated area within Germany and a sample user in Krefeld who has to go to Leverkusen. Fig. 3 shows a map of the area and we can quickly see different routes that however may entirely differ in their traffic conditions. Looking at our user's preferences we might find that it is important to get to Leverkusen quickly, i.e. on a short route preferably without traffic jams, but the weather doesn't really matter, because our user is dependent on using the car for transportation reasons. In this case a typical query would be: "Find the best route from Krefeld to Leverkusen and recommend short routes without jams regardless of the weather conditions". The posing of the query via an i-mode cell phone is carried out as shown in fig. 4 using a query form in cHTML.

Fig. 4: Mobile querying of routes with preferences

Having entered the current place and destination, the number of routes to return and the preferences as stated above the query is processed by our service. The evaluation of preferences assigns weights for our combining function F. If a "+" is checked we assign a weight of 3, the middle radio button assigns a weigh of 2 (default) and the "-" button results in a weight of 1.

Fig. 5: Query result with route suggestion, alternatives and SMS service

Thus in our above example we would get weights (3, 3, 2, 1) for the scoring on length, traffic jams, road works and weather conditions. The best routes are returned automatically formatted with cHTML sytylesheets for our i-mode client device (cf. fig. 5). By checking a box for

the route of choice and submitting the information the user can register for another service that alerts on changes in the current traffic situation for the route chosen.

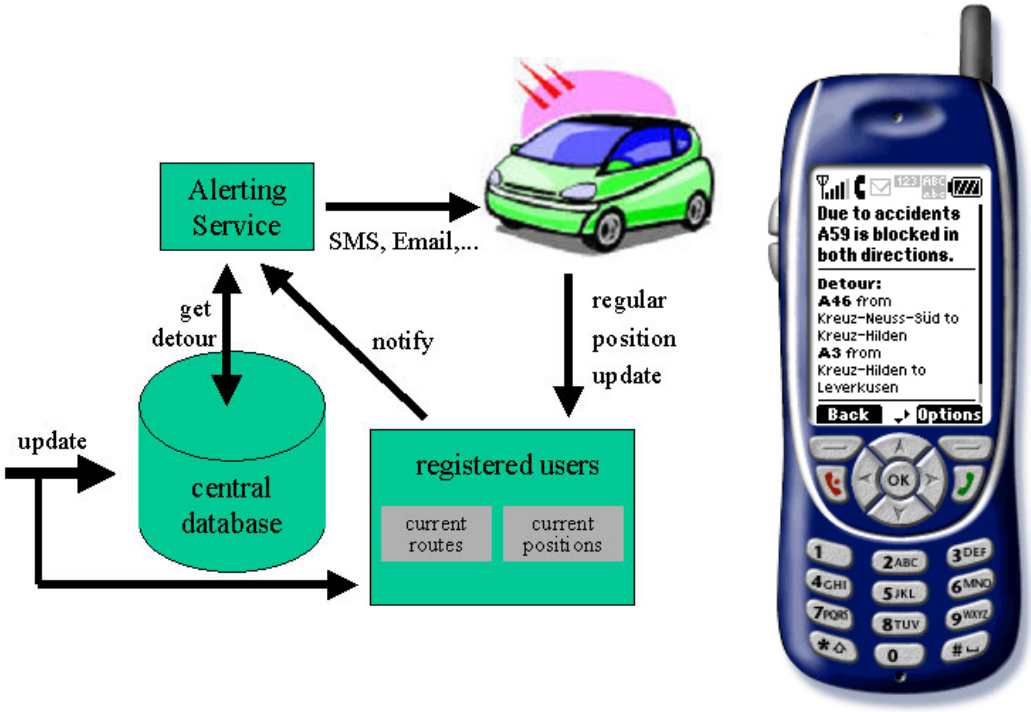


Fig. 6: SMS alerting Service for traffic information

The SMS service architecture adds an alerting service to our application server (cf. fig. 6). On each update of the central database its affection on the routes of all registered users is determined. If a route is affected by a current change the current position of the car is taken to repose the query with the car's current location and its destination. If a better route is detected an SMS or E-mail containing the affection and a most suitable rerouting is sent to the user (see fig. 6). Since SMS reading in cars may distract the driver's attention and is therefore not recommended, this service is right now not a fixed part of our traffic information system. Using voice output (e.g. VoiceXML, VoxML [XML02]) or getting the rerouting directly to car-bound navigation systems will in future solve this problem.

4.2. Quality of Personalization

To analyze the performance of our system we assume some sample values for our traffic condition. Our first aim will be to show, our preference settings improve the quality of service, secondly we are interested in run-times to investigate the service's real-world applicability meeting our psychologically founded 3 second response time restriction [Pop97]. Thus we will now take a look on our systems behavior given our sample application of the previous section. Table 2 shows characteristics for five routes from Krefeld to Leverkusen:

id	description	length	traffic jams	road works	weather
1	A57 Krefeld Köln A1 Köln Leverkusen	61 km	0 km	25 km	foggy with visibility 20 m
2	A57 Krefeld Neuss A46 Neuss Düsseldorf A59 Düsseldorf Leverkusen	64 km	4 km	10 km	cloudy, but dry

3	A57 Krefeld Neuss A46 Neuss Hilden A3 Hilden Leverkusen	66 km	6 km	15 km	sunny and dry
4	A57 Krefeld Neuss A46 Neuss Hilden A3 Hilden Langenfeld A542 Langenfeld Monheim A59 Monheim Leverkusen	71 km	0 km	5 km	weak rain, but warm
5	A57 Krefeld Neuss A46 Neuss Düsseldorf A59 Düsseldorf Monheim A542 Monheim Langenfeld A3 Langenfeld Leverkusen	75 km	10 km	20 km	weak rain, but warm

Table 2: Five routes and their characteristics

Given these values we can derive the following score values defining our ranked preference:

id	$f_{length}(id)$	$f_{traffic\ jams}(id)$	$f_{road\ works}(id)$	$F_{weather}(id)$
1	1.0	1.0	0.0	0.0
2	0.79	0.6	0.75	0.9
3	0.64	0.4	0.5	1.0
4	0.29	1.0	1.0	0.8
5	0.0	0.0	0.25	0.8

Table 3: Normalized scores for route characteristics

Considering our sample user from above who is interested in short routes avoiding traffic jams, we get a ranking as shown in the left hand side of table 4. The best route thus would be route 2 followed by route 1, though there are bad weather conditions. Consider on the other hand a user riding a motor bike. The preferences for this case would put a strong emphasis on the weather, since it's even more dangerous to drive on slippery roads when riding a motor bike. Using this profile we get the recommendation shown in table 4 on the right hand side. Thus the best two route would be number 2 and 3 both having dry weather conditions.

short without traffic jams	
id	**F(id)**
2	0.736
1	0.7
4	0.696
3	0.576
5	0.13

motor bike	
id	**F(id)**
2	0.8
3	0.74
4	0.71
1	0.43
5	0.38

Table 4:Recommendation for different user profiles

4.3. Performance Evaluation

Having shown our personalization to be sensible we will now focus on run-times. As bench-mark environment we used a 1.4 GHz AMD Athlon with 768 MB RAM as application and database server running IBM's DB2 V7.2 database and posed queries via 100MBit LAN. On our database of about 100000 routes we have taken the average over queries using different

locations and analyzed the runtimes for different numbers of routes to return (3, 5, 10). To verify that our preference settings have virtually no effect on the response times we compared 3 different settings. As can bee seen in figure 7 (left hand side) our response times meet the psychologically founded 3 sec response time barrier and only slightly rise with more objects to return. The right hand side shows how the average response time is composed. Please note that more than half of the time is consumed by the translation of the generic XML files for delivery and only about one tenth is the actual retrieval time.

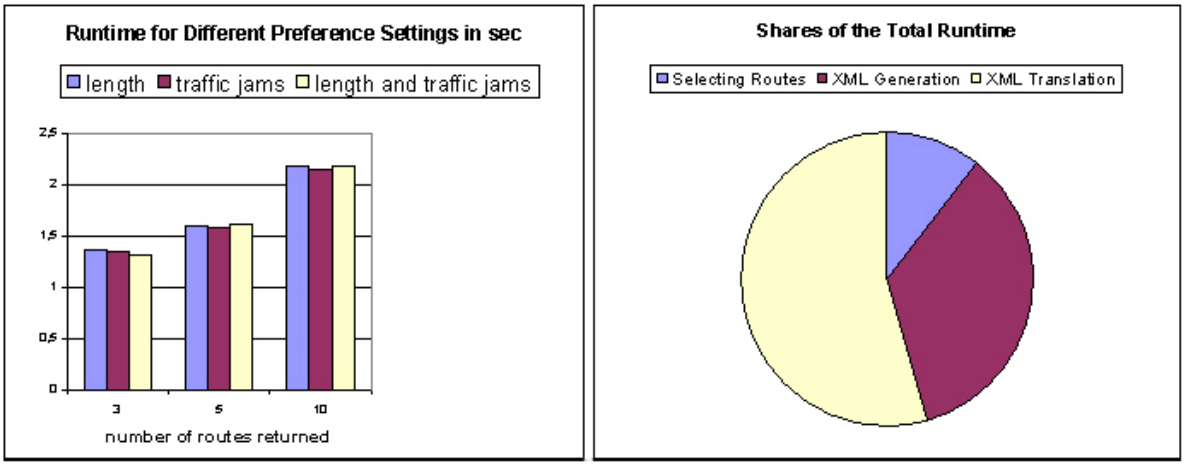


Fig. 7: Performance evaluation

5. Summary and Outlook

In this paper we presented the prototype of a mobile traffic information system. Using mobile client devices, like cell phones or PDAs, the user is informed about optimal routes for traveling. As a sample application we used the autobahn network of the German Ruhrgebiet area, but a generalization to other areas is of course straightforward. Due to low bandwidths and real-time requirements the routes are recommended by evaluating information within a central server architecture that uses asynchronous updates from various on-line-sources including traffic information from radio stations, road work information and local weather information. The user can express ranked preferences intuitively by emphasizing certain features of the route leading to an advanced personalization.

Integrating all the information with the user‘s preferences the efficient *SR-Combine* algorithm chooses adequate routes meeting the psychologically founded barrier of 3 seconds response time. Subsequently the results are delivered to the user by generating the appropriate delivery formats from generic XML data thus serving a variety of mobile client devices. A device dependent stylesheet library provides an adequate user interface design even on small screen estates. Moreover, changes in the current traffic situation are monitored and users can be alerted using an SMS service, if serious events on their routes suggest an alternative route for improved traveling. Though the current implementation is featuring a central server architecture to meet real-time requirements, a migration to distributed system architectures is easy to achieve due to *SR-Combine*’s flexibility. We presented an extensive case study together with encouraging performance results that promise the applicability of our situation-aware traffic information service in real-world applications.

Our future work will focus on improving the system’s personalization skills. Throughout this paper we always used ranked preferences, that are well suited for characteristics like the length of routes or number and length of traffic jams, but are somewhat clumsy in the case of qualitative characteristics like e.g. weather data. For these type of data we will improve our preference modeling focusing on qualitative aspects crucial for traveling, like e.g. wet, slip-

pery road conditions or black ice warnings. A second area of research deals with the integration of multi-modal transport and other characteristics that influence the traffic system, like e.g. public transport, parking facilities or floating car data. For these issues especially the GPS data processing has to be improved in order to get a reliable time schedule for the user's entire journey.

6. Literature

[BGK02] W.-T. Balke, U. Güntzer, and W. Kießling: On Real-time Top k Querying for Mobile Services, *Intern. Conf. on Cooperative Information Systems (CoopIS'02)*, Irvine, Ca, USA, 2002.

[BGM02] N. Bruno, L. Gravano, A. Marian: Evaluating Top-k Queries over Web-Accessible Databases. *Intern. Conf. on Data Engineering (ICDE'02)*, San Jose, CA, USA, 2002.

[Cho02] J. Chomicki: Querying with intrinsic preferences. *Intern. Conf. on Advances in Database Technology (EDBT)*, Prague, Czech Republic, 2002

[Fag96] R. Fagin. Combining fuzzy information from multiple systems. *Symposium on Principles of Database Systems (PODS'96)*, Montreal, Canada, 1996

[GB97] S. Gribble and E. Brewer. System design issues for internet middleware services: deductions from a large client trace. *USENIX Symposium on Internet Technologies and Systems*, Monterey, CA, USA, 1997

[GBK00] U. Güntzer, W.-T. Balke, W.Kießling: Optimizing Multi-Feature Queries for Image Databases. *Intern. Conf. on Very Large Databases (VLDB'00),* pp. 419-428, Cairo, Egypt, 2000

[GW00] R. Goldman and J. Widom: WSQ/DSQ: A practical approach for combined querying of databases and the web. *Intern. Conf. on Management of Data (SIGMOD'00)*, Dallas, TX, USA, 2000

[KB02] W. Kießling and W.-T. Balke. Mobile Search in a Preference World. *WWW 2002 Workshop on Mobile Search* , Honolulu, Hawaii, USA, 2002

[Kie02] W. Kießling: Foundations of Preferences in Database Systems. *Intern. Conf. on Very Large Databases (VLDB'02)*, Hong Kong, China, August 2002

[KK02] W. Kießling, G. Köstler: Preference SQL – Design, Implementation, Experiences. *Intern. Conf. on Very Large Databases (VLDB'02)*, Hong Kong, China, August 2002

[MCM98] A. Moukas, K. Chandrinos, and P. Maes: Trafficopter: A Distributed Collection System for Traffic Information, *Cooperative Information Agents (CIA'98)*, LNAI 1435, Paris, France, 1998

[NR99] S. Nepal and M. Ramakrishna: Query processing issues in image (multimedia) databases. *Intern. Conf. on Data Engineering (ICDE'99)*, Sydney, Australia, 1999

[OnS97] Onstar, General Motors Smart Car, http://www.onstar.com, 1997

[OR+98] M. Ortega, Y. Rui, K. Chakrabarti, K. Porkaew, S. Mehrotra, and T. S. Huang: Supporting ranked boolean similarity queries in MARS. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, Vol. 10 (6), 1998

[Pop97] E. Pöppel, A Hierachical Model of Temporal Perception. *Journal of Trends in Cognitive Science*, Vol. 1, Elsevier, 1997

[Rou02] http://www.routenplaner-online.com/

[TSH01] M. Tork Roth, P. Schwarz, L. Haas: An Architecture for Transparent Access to Diverse Data Sources. *Component Database Systems*, Morgan Kaufmann, 2001.

[W3C02] WWW Consortium: Compact HTML standard, *W3C Note - standard under discussion,* http://www.w3.org/TR/1998/NOTE-compactHTML-19980209

[WBK02] M. Wagner, W-T. Balke, and W. Kießling: An XML-based Multimedia Middleware for Mobile Online Auctions. *Enterprise Information Systems III*, Kluwer Academic Publishers, 2002.

[XML02] XML.org , http://www.xml.org/xml/resources_focus_voicexml.shtml

[YTT00] M. Yearworth, N. Taylor, J. Tidmus, I. Fraser, and P. Still: A CORBA Service for Road Traffic Information on the Internet. *Intern. Symposium on Distributed Objects and Applications (DOA'00),* Antwerp, Belgium, 2000.