retino – real time notifications

Seminarbericht Context-Aware Computing
Ubiquitous / Pervasive Computing

Daniel Buchmüller, Fabian Hensel
db@icontent.ch, fabian@iswitch.ch

Einleitung

In der heutigen Welt sehen sich immer mehr Menschen mit einem Überfluss an Informationen eingedeckt. Trotzdem ist es wichtig über relevante Geschehnisse und Abläufe stets informiert zu sein. Information bedeutet Macht. Durch immer intelligentere mobile Geräte und gut ausgebaute Telekommunikationsinfrastrukturen, ist der Zugang zu elektronischen Medien auch unterwegs zunehmend gesichert. Nicht alle Inhalte können einem mobilen Benutzer jedoch in der gleichen Form präsentiert werden, wie dies an einem voll ausgebauten Arbeitsplatz möglich wäre. Mobile Geräte sind etwa durch kleinere Bildschirme in der Fähigkeit komplexe Inhalte darzustellen stark eingeschränkt. Um jemanden unterwegs also mit Informationen versorgen zu können, bedürfen diese einer speziellen Aufbereitung und

Darstellung. Ausserdem ist es wichtig, dass Benutzer die für sie wirklich relevanten Informationen angezeigt bekommen. Die Relevanz kann jedoch anhand von verschiedenen Parametern, wie Ort und Zeit, mehr oder weniger stark variieren. Es gilt deshalb diese Parameter zu erfassen und eine entsprechende Filtrierung der Informationen vorzunehmen, bevor diese dem Benutzer präsentiert werden.

retino Projekt

retino steht für „real time notifications“, liefert einen Lösungsansatz für das einleitend beschriebene Problem Benutzer von retino werden durch ihr mobiles Gerät automatisch mit den für sie relevanten Informationen versorgt. Dabei greift retino primär auf Daten zurück, die von den Benutzern selbst in das System eingespiesen werden. Jeder Nutzer von retino ist also Teil einer Community und trägt als solcher zum Funktionieren des Systems bei. Die Context-Awareness von retino wird hauptsächlich durch die drei Parameter: Zeit, Ort und Interessen bestimmt. Der Ort des Benutzers wird kontinuierlich erfasst, weshalb man retino zur Klasse der Location-based-services zählen kann. Die Interessen des Benutzers werden vor der Verwendung des Systems erfasst und können zu jedem Zeitpunkt geändert werden. Ziel von retino ist es, aus den gesammelten Daten aller Benutzer einen Informationsmehrwert für die Community zu schaffen.

Entstehung

Das retino Projekt entstand in Zusammenhang mit der Teilnahme des österreichisch, schweizerischen Teams (Alpenregion) am internationalen Studentenwettbewerb „Imagine Cup“ im Jahre 2004. Das Motto des Wettbewerbs war dabei „Imagine a world where smart technology makes everyday life easier“. Dieser von Microsoft lancierte Event teilt sich in mehrere Bereiche moderner Technologien. retino wurde dabei für das weltweite Finale, welches in Brasilien stattfand, in der Kategorie „Software Design“ entwickelt.

Ort: Geographische Positionsbestimmung

Inzwischen gibt es verschiedene Möglichkeiten, die Position eines Gerätes und damit meist auch eines Menschen zu bestimmen. Grundsätzlich lassen sich die Systeme dabei in zwei Kategorien einteilen: Kontinuierliche Systeme und diskrete Systeme. Während kontinuierliche Systeme die Position fortlaufend bestimmen können und diese Daten in periodischen, meist kurzen Zeitabständen zur Verfügung stellen, liefern die diskreten Systeme jeweils nur eine Positionsangabe.

Sicherlich das bekannteste Positionsbestimmungssystem ist GPS. Das vom amerikanischen Militär entwickelte System erlaubt eine in der Regel auf mehrere Meter genaue Bestimmung. Derzeit ist ein alternatives System der europäischen Raumfahrtbehörde ESA namens „Galileo“ in Entwicklung[1] [1], welches in Zukunft noch genauere, satellitengestützte Positionsbestimmung möglich machen wird. Problematisch und fehlerbehaftet ist die satellitengestützte Positionsbestimmung in Gebäuden und bei fehlendem Sichtkontakt zu den Satelliten. Die Empfangsgeräte sind heute sehr günstig zu beschaffen und liefern als kontinuierliche Datenquellen Positionsdaten in regelmässigen, kurzen Abständen. Ausserdem können sie als Quelle von genauen Zeitdaten verwendet werden, da diese für den Betrieb satellitengestützter Systeme benötigt werden.

Eine alternative Möglichkeit der kontinuierlichen Bestimmung bieten die Parameter von Mobilkommunikationsnetzen wie GSM oder UMTS. Da ein Endgerät für den funktionsfähigen Telefonbetrieb die jeweils aktuellen Netzparameter speichern muss, ist der Zugriff und die Nutzung dieser Daten sinnvoll. Im Allgemeinen gehen die Netzbetreiber jedoch sehr restriktiv mit der Herausgabe ihrer Netzdaten um, so dass eine Zuordnung von gemessenen Parametern zu einer tatsächlichen Position nicht ohne weiteres möglich ist. Des

Weiteren spielt die Zellgrösse eine Rolle. Da die Funkzellen in ländlichen Regionen in der Regel um ein Vielfaches grösser (bis zu 30km) als in städtischen Gebieten (wenige Meter) sind. Der grosse Vorteil gegenüber satellitengestützten Systemen ist deren Verfügbarkeit auch im Innern von Gebäuden, überall dort wo das Netz empfangen werden kann.

Zunehmend attraktiv vor allem für die Indoor-Positionsbestimmung und in Ergänzung zu GPS [2] ist die Verwendung der Netzparameter von drahtlosen Netzwerken, WLANs. Zugangsknoten von WLANs senden in kurzen Abständen sogenannte Beacons. Diese werden von potentiellen Clients empfangen und weisen den Benutzer auf das Vorhandensein eines bestimmten Netzes hin. Die in jedem Beacon enthaltene ESSID ermöglicht die Zuordnung zu einem spezifischen Zugangsknoten. Nach entsprechender Aufzeichnung der Empfangsbereichs und der Verknüpfung mit der tatsächlichen geographischer Position lässt sich damit ein System mit einer Genauigkeit von einigen zehn Metern realisieren. Die Methode lässt sich auch mit fremden und verschlüsselten Netzen realisieren, da keine Notwendigkeit zur Verbindungsaufnahme besteht. Da nicht überall Zugangsknoten vorhanden sind, ist diese eine diskrete Form der Positionsbestimmung.

Selbige Methode lässt sich auch für andere Kleinfunknetzwerke, wie etwa Bluetooth realisieren. In der Vergangenheit wurden ähnliche Systeme mit Infrarot-Transpondern eingesetzt, die jedoch den Nachteil der notwendigen Sichtverbindung hatten und deshalb nur in sehr eingeschränkten Räumen empfangen werden konnten. In Zukunft könnten RFIDs eine wichtige Rolle bei der diskreten Positionsbestimmung spielen, da sie einfach und kostengünstig installiert werden können.

Letztlich existiert auch die Möglichkeit die Position manuell in ein System einzugeben. Dies könnte beispielsweise die Eingabe von Strassennamen an einer Kreuzung sein. Diese diskrete

Methode erlaubt es einem System jedoch nicht, automatisch auf Positionsänderungen des Benutzers zu reagieren. Die Context-Awareness ist in diesem Fall nicht gegeben.

Zeit

Die gegenwärtige Zeit ist sowohl auf dem mobilen Gerät wie auch auf dem Serversystem durch die in der Hardware implementierte Echtzeituhr gegeben. Allenfalls muss diese bei der ersten Inbetriebnahme manuell eingestellt werden.

Bei der Verwendung von GPS, können die Zeitsignale der Satelliten für eine äusserst genaue Zeitbestimmung verwendet werden. Um die Funktionalität von retino zu gewähren, ist die Genauigkeit der üblich verbauten Quarzoszillatoren genügend.

Interessen

Die Interessen eines Benutzers stehen nicht zum Vornherein fest. Deshalb müssen die Interessen in Form von Abonnements zu Services vor der ersten Verwendung auf der Weboberfläche angegeben werden. Mit zunehmender Vielfalt von Services, wird die Auswahl der passenden Angebote für den Benutzer komplizierter. Oft wollen Anwender jedoch möglichst wenig Zeit mit dem persönlichen Einrichten von Diensten verbringen, so dass hier hohe Anforderungen an die Benutzeroberfläche gestellt werden (z.B. Vorhandensein von Suchfunktionen).

Clienttypen [3]

Aufgrund der beschränkten Rechenleistung auf mobilen Geräten, ist es nicht sinnvoll Applikationen mit grossem Rechenaufwand auf den Geräten auszuführen. Stattdessen wird das mobile Gerät bloss zur Darstellung von Informationen und als Eingabegerät verwendet (client tier). Die Daten werden mittels einer Netzwerkverbindung zu einer zentralen Stelle

befördert, wo sie weiterverarbeitet werden (application-server tier). Die Speicherung der Daten werden meist von einer Datenbank übernommen (data-server tier).

Bei einer Thin Client Implementierung würde gar keine oder eine nur sehr kleine Grundapplikation auf den client tier geladen werden, die zur Verbindungsherstellung benötigt wird. Oft wird heute auf Web-Interfaces zurückgegriffen, die keine Installation von Software erfordern, da ein Web-Browser zumeist schon vorinstalliert ist.

Im Falle eines Fat Clients wird hingegen ein umfangreiches Softwarepaket auf das Gerät geladen. In der Regel geschieht sodann auch die Berechnung auf dem Gerät selbst. Die Applikation kann folglich auch autonom, also ohne Netzwerkverbindung funktionieren. Ein Beispiel hierfür ist Beeloop.com, welche interaktive Stadtkarten zum Herunterladen auf mobile Geräte anbieten.

retino, in welchem das mobile Gerät den eigentlichen Awareness Sensor darstellt, wurde als Mischform realisiert. So muss zwar eine Clientsoftware auf dem Gerät installiert werden (fat), welche jedoch keinerlei Logik enthält (dumb), sondern nur die Benutzeroberfläche zur Verfügung stellt und somit für die Ein- und Ausgabe von Benutzerdaten zuständig ist. Eine reine Web-Anwendung konnte nicht realisiert werden, da die Anbindung von externen Geräten, wie beispielsweise einem GPS-Empfänger auf diese Weise nicht möglich ist.

	Smart	Dumb
Thin		Keine lokale Applikation Web-Application
Fat	Lokale Applikation Lokale Intelligenz → beeloop.com (Stadtkarten)	Lokale Applikation Entfernte Intelligenz → retino

Tabelle 1: Clienttypen

Fachbegriffe

Um die folgenden Erklärungen besser verstehen zu können, sollten die folgenden Begriffsdefinitionen beachtet werden:

- Ein „Location Report“ ist jene Nachricht, die ein Benutzer von retino auf seinem mobilen Gerät absendet, wenn dieser ein bestimmtes Ereignis sichtet und dieses der Community mitteilen will.

- Ein „Event“ wird auf dem Server aus einem oder mehreren „Location Reports“ generiert. Dieser steht dann zur Weiterverteilung an andere Benutzer bereit.

- Ein „Service“ ist ein Interessenskanal für „Events“. Benutzer können sich zu „Services“ abonnieren und selber neue „Services“ erstellen.

- Eine „Notification“ ist jene Nachricht, die ein Benutzer auf seinem mobilen Gerät dargestellt bekommt, welche ihn über den Ort eines „Events“ informiert zu dessen „Service“ er sich abonniert hat.

Architektur

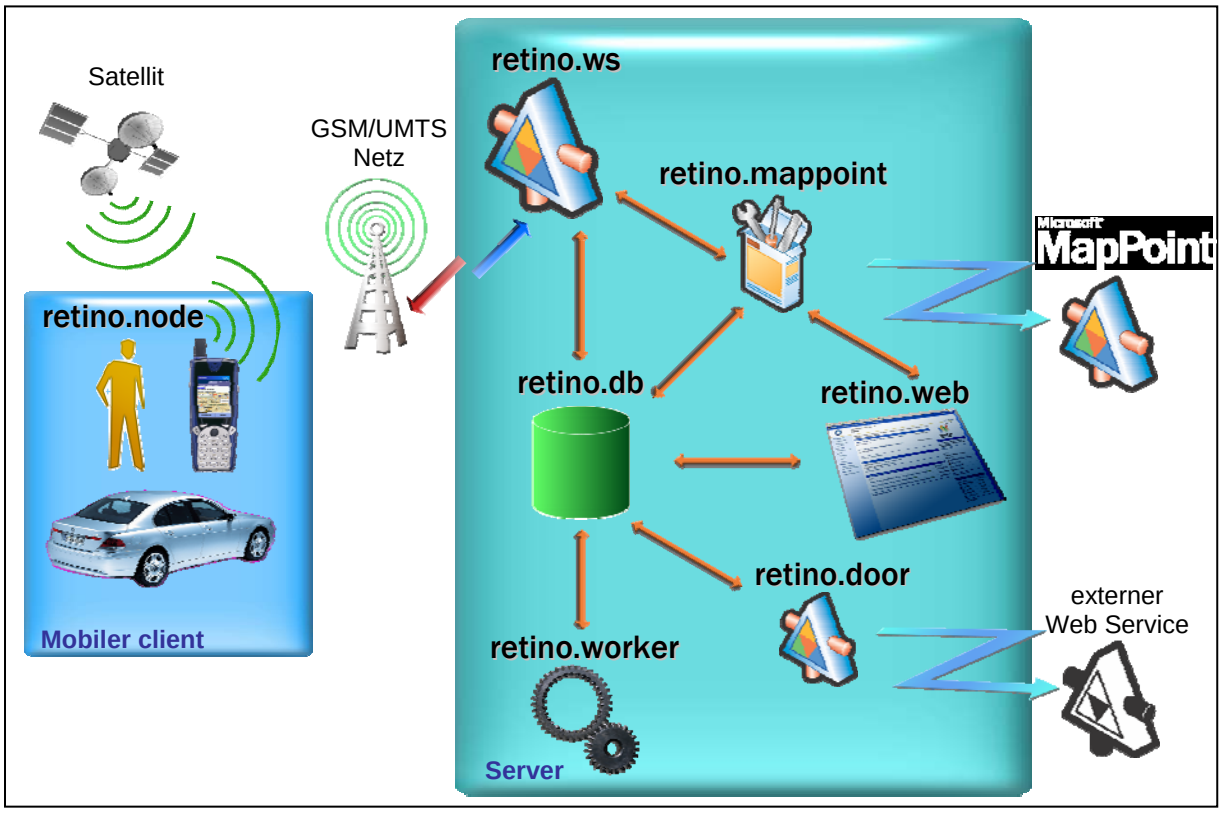

Abbildung 1: Architektur von retino

- retino.node

  Der Sensor und zugleich Melder von retino ist retino.node. Dies ist eine auf Microsoft .NET Compact Framework aufgebaute Applikation, welche auf Microsoft Pocket PCs installiert wird. Nach dem Start von retino.node müssen die auf retino.web eingerichteten Benutzerdaten nur einmal eingegeben werden. Anschliessend meldet sich retino.node am System an (über retino.ws). Sobald auch das GPS-Signal verfügbar und verlässlich ist (vier und mehr Satelliten empfangen werden), kann retino.node zum Senden von Location Reports und Empfangen von relevanten Notifications verwendet werden. Diese werden ebenfalls über retino.ws ins System eingespiesen bzw. empfangen.

- retino.db

  Die Daten von retino werden zentral in retino.db gehalten. Technologisch ist das ganze mittels dem Microsoft SQL Server 2005 realisiert.

  retino.db ist neben über der Datenspeicherung auch die Kommunikationsschnittstelle zwischen retino.web, retino.ws, retino.mappoint, retino.door und am wichtigsten retino.worker. Die Kommunikation ist nach einem Request/Response-Modell aufgebaut, so dass Request-Records von Anfrager geschrieben werden und von der anbietenden Komponente aufgelesen und gekennzeichnet (d.h. es wird vermerkt, dass der Request bearbeitet wurde) werden. Nach dem Verarbeiten schreiben die Komponenten die Antwort in der Form eines Response-Records in retino.db, welcher dann nach dem gleichen Prinzip wieder von der ursprünglich anfragenden Komponente verarbeitet wird.

  Beim Microsoft SQL Server 2005 können für die Benachrichtigung über neue Request/Response-Records Trigger eingesetzt werden. So kann ein performance-belastendes Polling auf retino.db vermieden werden. Einige Komponenten fragen nur auf Requests von Aussen retino.db an.

- retino.ws

  Die Kommunikation zum retino.node stellt retino.ws zur Verfügung. Es ist ein auf Microsoft ASP.NET 2.0 basierender Web Service. Der retino.ws ist nur ein Mediumsbroker und hat selber keine Logik, sondern reicht die Requests nur an retino.db weiter und empfängt von dieser die Responses welche dann in Form von Notifications an retino.node gesendet werden.

- retino.web

  Die gesamte Administration, sowie das Einrichten von Benutzerkonten und Services erfolgt über retino.web. Dies ist eine auf ASP.NET 2.0 aufgebaute hochperformante Webapplikation. Auch diese enthaltet selber keine Funktionslogik auf hoher Schicht, sonder reicht nur die Eingaben an retino.db weiter.

- retino.mappoint

  Kartendaten werden über die retino.mappoint Web Service-Komponente von Microsoft bezogen und über das Dateisystem retino.web und retino.node angeboten.

- retino.worker

  Das zentrale “Hirn” von retino ist retino.worker. Dies ist ein Konsolenprogramm, welches auf dem Server in einer Endlosschleife läuft, auf Requests/Responses von retino.db wartet und diese verarbeitet. retino.worker beinhaltet die ganze Funktionslogik und -regeln. So können zum Beispiel benutzerdefinierte Klassifizierungsregeln für die Generierung von Events aus den einzelnen Location Reports darin implementiert werden. Die einzige Kommunikationsschnittstelle von retino.worker ist retino.db.

- retino.door

  Um das Prinzip des offenen Frameworks zu demonstrieren, wurde retino.door implementiert. retino.door kann externe Datenquellen (z.B. von einem Web Service) einbinden und in retino.db ablegen. Diese werden dann über die erwähnten Pfade, retino.web und retino.node verteilt. Ein Beispiel hierfür ist der AtomicLava Moblog Web Service: ein Web Service auf dem Server von AtomicLava UK, welcher uns die Firma eigens für retino zur Verfügung gestellt hat.

Node-Server Verbindung (retino.ws)

Das mobile Gerät, der retino.node, kommuniziert über Webservices (retino.ws) mit dem Server. Webservices stellen eine mit .NET einfach zu implementierende Art zum Gebrauch von Objekten an unterschiedlichen Orten dar. Als grösster Nachteil beim Einsatz in mobilen Systemen ist bestimmt der relativ grosse Overhead zu betrachten, der beim Datenaustausch mittels der bei Webservices verwendeten XML entsteht. Der Internetzugang des mobilen Gerätes, welcher meist als GPRS-Verbindung realisiert ist (in gewissen Gebieten wäre auch WLAN denkbar), ist vergleichsweise langsam, weist hohe Verzögerungszeiten auf und sehr teuer. Ein in nächster Zukunft zu erwartender Preisabfall wird viele Dienste dieser Art attraktiver machen.

Webservices funktionieren über fast jede Art von vorhandener Verbindung, da sie zur Übertragung das HTTP Protokoll verwenden. Weil jedoch viele Mobilfunkanbieter, welche die GPRS-Verbindung zur Verfügung stellen, NAT verwenden, ist es meist nicht möglich an das mobile Gerät Daten zu senden, ohne dass zuvor von diesem eine Datenverbindung aufgebaut wurde. (Ausnahme: WAP/SMS-Push) Deshalb wird im retino.node der Webservice auf dem Server in periodischen Zeitabständen aufgerufen, um nach neuen Benachrichtigungen zu suchen. Der grosse Nachteil dieser Lösung sind folglich hohe Datenvolumen und geringe Akkulaufzeit der Geräte.

Datenverarbeitung

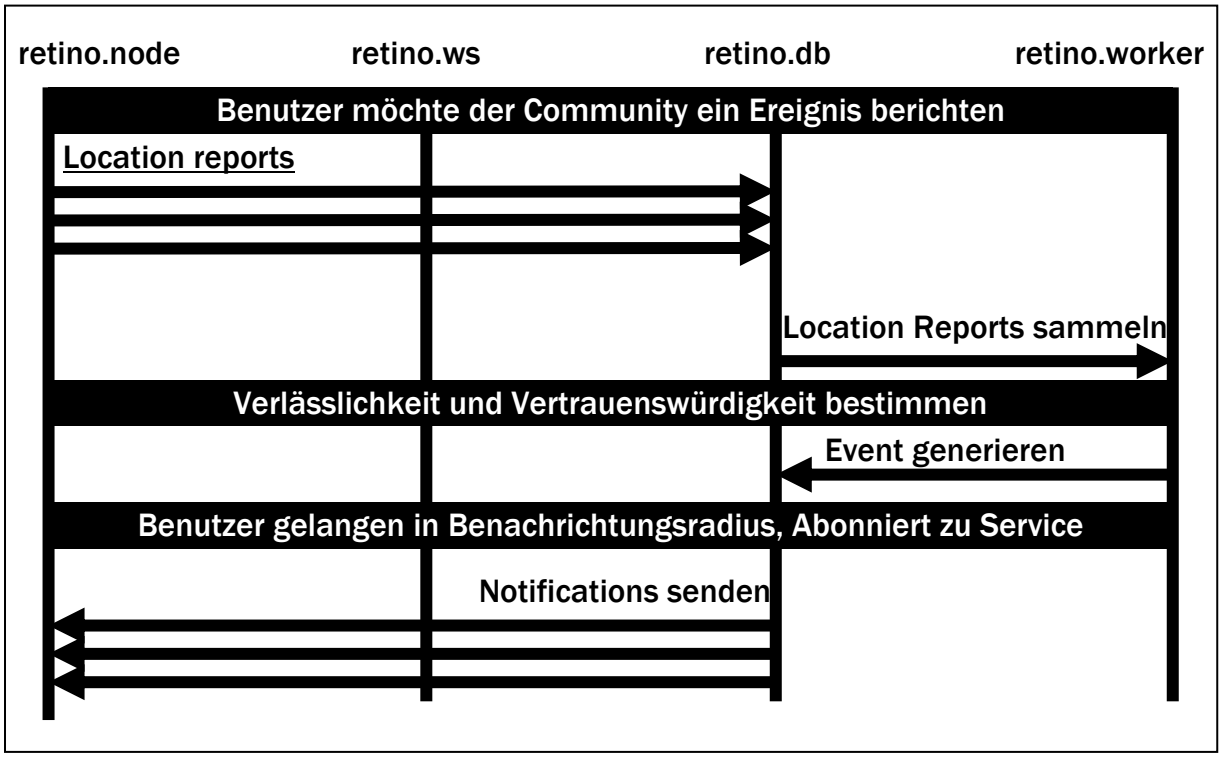

Abbildung 2: Datenverarbeitungsprozess

Möchte ein Benutzer von retino ein von ihm beobachtetes Ereignis der Community mitteilen, so tut er dies durch das Anwählen des entsprechenden Service auf dem Report-Tab. In der Folge wird dieser Report erst lokal auf dem Gerät gespeichert und mit Positionsdaten angereichert. Bei der nächsten Verbindungsaufnahme zum Webservice wird der Report sodann zum Server übertragen. Dieser speichert vorerst alle empfangenen Location Reports in einer Datenbank (retino.db).

Da ein einzelner Location Report von einem einzigen Nutzer im Normalfall keine genügende Verlässlichkeit bietet, müssen zur anschliessenden Generierung eines Events erst noch einige Location Reports von anderen Benutzern empfangen werden, um das Vorhandensein dieses Events mit einer vorgegebenen statistischen Verlässlichkeit garantieren zu können. Die Logik, welche hierbei verwendet wird, ist vollständig im retino.worker modularisiert. Die Parameter,

die zur Entscheidung über eine Eventgenerierung herangezogen werden, differieren je nach Service und können bei dessen Erstellung durch den jeweiligen Benutzer festgelegt werden.

In der derzeitigen Implementierung von retino sind die folgenden Parameter realisiert worden:

- Das Zeitfenster gibt an, über welche Zeit hinweg ein Event von Bedeutung ist.

  Während einige Ereignisse nur während einer kurzen und bestimmten Zeit relevant sind, können die Events anderer Services über eine längere Zeit von Bedeutung sein. Um eine hohe Informationsrelevanz erreichen zu können, ist zum Beispiel eine kürzere Zeitdauer des letzten Location Reports bei einer mobilen Radarkontrolle wichtiger als bei einem Restaurant.

  Ein Event wird verworfen, sobald die Zeit seit dem letzten Location Report die angegebene Zeit überschritten ist (timeout).

- Die Positionstoleranz gibt an, in welchem Radius die gesammelten Location Reports als ein einziger Event eines Service klassifiziert werden.

  Da nicht alle Benutzer für den Event einen Location Report am genau demselben Ort absetzen werden, ist es nötig eine gewisse Toleranz einzuberechnen. Ausserdem ist zu berücksichtigen, dass auch die Positionsdaten des GPS-Signals variieren können.

Zur Bestimmung, ob ein Event als Notification einem bestimmten Benutzer zugestellt wird, ist schliesslich die Benachrichtigungstoleranz von Bedeutung, welche angibt, in welchem Radius der Benutzer benachrichtigt werden, wenn sie sich dem Ort eines Event nähern.

Für den Testbetrieb wurde das System kurzgeschlossen, da nur ein einziges Clientgerät zur Verfügung stand. Folglich wurden aus allen eingetroffenen Location Reports direkt Events generiert. Im produktiven Betrieb wäre hier beispielsweise der Einsatz von neuronalen Netzen oder Bayesian Belief Networks denkbar.

Nähert sich ein Benutzer, welcher zu einem bestimmten Service abonniert ist, einem Event, so wird dies durch die kontinuierlich zugesandten Positionsdaten serverseitig erkannt und dem Client bei der nächsten Kontaktaufnahme eine Notification zugestellt. retino.node macht den Benutzer durch ein akustisches Signal auf das Vorhandensein einer neuen Notification aufmerksam, wechselt automatisch auf den Notification-Tab und zeigt die neueste Notification an. Dies erlaubt es dem Benutzer ohne weitere Eingaben stets über neue relevante Ereignisse informiert zu sein, was beispielsweise beim Gebrauch des Systems während der Fahrt von Vorteil ist.

Community

retino verwendet als Informationsquelle primär seine Benutzer selbst. Externe Quellen können jedoch dank der modularen Struktur und der Verwendung der Service Oriented Architecture sehr einfach angebunden werden. Communities bieten viele Vorteile, doch bergen sie auch Risiken. Es ist naheliegend, dass das System nur ab einer kritischen Grösse von Teilnehmern überhaupt funktionieren kann, da ansonsten nie genügend Informationen im System überhaupt zur Verteilung vorhanden sind. Das Interesse und die Motivation einen Beitrag zur Community zu leisten, wächst mit dem Vorhandensein bereits existierender nützlicher Informationen. Deshalb ist bei einem produktiven Einsatz diesem Aspekt sicherlich grosse Beachtung zu schenken.

Ein immer grösser werdendes Problem im Umfeld elektronischer Systeme ist auch der Missbrauch. Bei der Verwendung von retino hat jeder Benutzer ein eigenes Benutzerkonto, welches zu dessen Identifikation dient und sowohl auf dem mobilen Gerät, wie auch bei der Administration auf der Weboberfläche eingegeben werden muss. Für zusätzlichen Komfort lassen sich die Benutzerdaten auf dem mobilen Gerät abspeichern, so dass diese nicht bei jeder Verwendung neu eingegeben werden müssen. Um das, gegebenenfalls auch unabsichtliche, Problem des Absetzens von falschen Location Reports zu entschärfen, sind mehrere Location Reports notwendig, um das Generieren eines Events auszulösen. In einer erweiterten Anwendung liesse sich auch ein Bewertungssystem verwenden, welches bewährten Nutzern mehr Vertrauen schenkt und deren Location Reports stärker gewichtet, so dass im Extremfall ein einziger Report eines vertrauenswürdigen Benutzers ausreicht, um einen Event zu generieren.

Analog liesse sich auch ein „Web of Trust“ [4] realisieren, bei welchem die Vertrauenswürdigkeit der Benutzer durch die bereits bestehende persönliche Bekanntschaft unter den Benutzern zur Einstufung verwendet werden kann.

Zusätzlich zum manuellen Absetzen von Location Reports, lassen sich befreundete Communitymitglieder mit dem „Friends-Kennzeichnung“ versehen. retino erkennt in diesem Fall, wenn sich zwei befreundete Benutzer in einer bestimmten Entfernung befinden und sendet beiden eine Notification, die sie über die Position des anderen Benutzers informiert, die sich dann beispielsweise zu einem Treffen verabreden können.

Anwendungsbeispiele

Die folgenden Beispiele sollen das Verständnis für die Einsatzvielfalt von retino verdeutlichen. [5]

- retino ist dank der einfachen Benutzeroberfläche auch in Fahrzeugen verwendbar. Als Ergänzung zu bereits vorhandenen Navigationssystemen. Ein beliebter Dienst ist das Darstellung von Geschwindigkeitsmessanlagen, bei dessen Vorbeifahrt der Fahrer einen Hinweis erhält, seine Fahrtgeschwindigkeit entsprechend anzupassen. Mit retino wird der Fahrer nicht nur über fest installierte Anlagen informiert, sondern auch über mobile Anlagen, die von der Polizei nur vorübergehend aufgestellt werden. Entsprechend kann der Fahrer die Community über noch nicht erfasste Anlagen orientieren.

- Am Wochenende stösst ein retino Benutzer zu einer spontan organisierten Party. Da er die Veranstaltung gut findet, sendet er einen Report über sein mobiles Gerät an die Community. Andere Mitglieder, die sich in der Nähe befinden und zum „Party-Service“ abonniert sind, bekommen eine Notification zugestellt und begeben sich ebenfalls dazu.

Benutzeroberfläche

Der Zugang zu retino wird für den Benutzer über zwei verschiedene Zugangspunkte realisiert: zum einen das Webinterface, zum anderen das mobile Gerät. Diese Aufteilung wurde absichtlich so gewählt. Der Hauptgrund liegt in den sehr eingeschränkten Anzeige- und Bedienmöglichkeiten von mobilen Geräten. Das kleine Display verhindert die effiziente Darstellung vieler Informationen gleichzeitig. Das mobile Gerät kann deshalb als Schnittstelle zwischen dem Menschen und retino gesehen werden, als Sensor und Benachrichtigungsgerät. Weitergehende Tätigkeiten, wie das Verwalten von Service-Abonnements, geschehen deshalb auf über das Webinterface von einem beliebigen Computer mit Webbrowser aus.

Abbildung 3: Report- und Notification-Tab von retino.node

Abbildung 4: Settings- und GPS-Tab von retino.node

Der mobile Client, retino.node, wird erst vom Webinterface heruntergeladen und mittels der Microsoft Synchronisierungssoftware ActiveSync auf den Pocket PC geladen. Die Software steht anschliessend zur Ausführung aus dem Menü bereit. Nach dem Programmstart wird dem Benutzer eine Statusübersicht präsentiert, welche über die Verbindung zum Server und das Vorhandensein eines brauchbaren Signals des GPS-Geräts orientiert. Um den begrenzten

Platz des Displays besser ausnutzen zu können, wurden Tabs eingesetzt, die die mehrfache Verwendung derselben Displayfläche erlaubt. Neben dem Status-Tab existieren die folgenden weiteren Tabs:

- Der Tab „Notifications“ zeigt dem Benutzer die eingetroffenen Notifications an. Die Navigationspfeile erlauben das Durchblättern der Notifications, wobei neuere immer zuerst angezeigt werden. Der Nutzer wird kurz auf den Service hingewiesen, um welchen es sich bei der Notification handelt. Im Zentrum steht die Karte, welche dem Nutzer seine gegenwärtige Position mit einer Richtungsangabe und den Ort des Events anzeigt. Schliesslich wird dem Benutzer die Position in Form der Adresse in Textform präsentiert. Beim Empfang neuer Notifications, wechselt retino.node automatisch auf den Notifications-Tab und gibt eine akustischen Hinweis aus, um dem Benutzer möglichst schnell über den Event zu orientieren, auch wenn dieser gerade anderwärtig beschäftigt ist.

- Der Tab „Report“ zeigt eine Liste mit den vom Benutzer abonnierten Services an. Diese Liste wird periodisch aktualisiert. Die grosse Schrift ermöglicht das Absetzen von Location Reports ohne den Stift des mobilen Gerätes verwenden zu müssen, was ebenfalls die problemlose Verwendung während der Ausführung anderwertiger Tätigkeiten erlaubt.

- Der Tab „GPS“ zeigt die vom GPS-Empfänger empfangen Daten an. Die Statuszeile informiert analog zum Status-Tab, ob das gegenwärtige Signal zur genauen Positionsbestimmung genügt. Die Position wird in CH1903-Koordinaten [6] angegeben, weitere Daten sind Höhe, Geschwindigkeit und Richtung.

- Der Tab „Settings“ dient dem Einrichten von retino.node. Die Eingabe des Login und Passwort dient der Authentifizierung gegenüber dem Server. Des Weiteren kann die Schnittstelle für die Verbindung zum GPS-Gerät ausgewählt werden. Der „File Tracker“ erlaubt das Speichern der Positionsdaten während der Benutzung des Programms und kann der anschliessenden Auswertung mit anderen Programmen dienen. Die „Auto-connect“ Funktion erlaubt das Speichern der eingegebenen Benutzerdaten auf dem Gerät, so dass bei der nächsten Benutzung die Software automatisch zum Server verbindet.

Das Webinterface [7] von retino wird für alle Belange der Benutzeradministration verwendet. Neue Benutzer können sich auf dem System registrieren. Nach einem Login stehen dem Benutzer verschiedene Funktionen zur Verfügung: Serviceabonnierung, Serviceerstellung, Abrufen von Location Reports, Events und Notifications, Übersicht über andere Benutzer von retino, sowie die Festlegung des Friends-Status. Des Weiteren können erweiterte Funktionen der Plugins von retino.door abgerufen werden.

Verwandte Arbeiten

Erste Ansätze zur Bewältigung des Informationsflutproblems wurden bereits 1996 mit dem „Remembrance Agent“ begangen. [8] Grundsätzlich stellt dieser eine Erweiterung zum Texteditor Emacs dar, der den Benutzer auf andere Dokumente aufmerksam macht, die für ihn von Interesse sein könnten. Dazu werden vor allem Wortvergleiche und –zählungen eingesetzt. Ähnliche Systeme werden heute beim AdSense Programm von Google [9] eingesetzt, sowie beim Microsoft Agent [10], der den Benutzer beim Verwenden von Office-Programmen entsprechende, kontextbasierte Hilfestellungen bietet.

Benachrichtigungsdienste im Zusammenhang mit Location-based Services wurden von IBM analysiert [11]. Hierbei wurde das System stark in die Infrastruktur eines GSM Netzwerks eingebunden. Anwendungsgebiete reichen hierbei von Staumeldungen bis hin zu Werbebotschaften, die beispielsweise ein Supermarkt ihren Kunden auf das Mobiltelefon zustellen kann, wenn diese den Parkplatz erreichen. Ein ähnliches System, welches Bluetooth verwendet, ist in Allianz Arena in München in Betrieb [12]. Mit comMotion [13] wurde ein System entwickelt, welches dem Benutzer ebenfalls ortsabhängige Informationen zur Verfügung stellt, jedoch nur über persönliche Daten verfügt und keine externen Quellen verwendet. Dafür bietet comMotion eine Sprachsteuerung an.

Die grösste Verbreitung finden ortsbasierte Dienste in der Tourismusbranche. Unzählige Arbeiten befassen sich mit der Anwendung von Ortinformationen, um Touristen in einer fremden Stadt an interessante Orte zu führen. Verschiedene Systeme, welche unterschiedliche Technologien zur Lokalisierung verwenden, wurden entwickelt. Cyberguide [14] ist eines der ersten kontextbasierten Touristenführer, welches sowohl innerhalb wie auch ausserhalb von Gebäuden eingesetzt werden kann. Basierend auf Cyberguide wurde später das Guide [15] Projekt realisiert, welches diskrete Positionsbestimmung anhand von Wireless-LAN Zugangsknoten nutzt und den Benutzern auch einen einfach Kommunikationsdienst zur Verfügung stellt. Die Awarenessfaktoren, welche retino verwendet: Zeit, Ort und Interessen, wurden ebenfalls bereits analysiert. [16] Allerdings verwenden diese Lösungen statische Daten und erlauben keine interaktive Beeinflussung dieser für andere Benutzer.

Eine innovative Lösung, welche dasselbe Ziel verfolgt, ist mSpace Mobile. [17] Im Unterschied zu retino, verwendet mSpace jedoch primär Datenquellen des Semantic Web. Es existiert jedoch ein Rückkanal in Form einer einfachen Kommentierungs- und Bewertungsfunktion, welche eine Einflussnahme auf die zur Verfügung stehenden

Informationen ermöglicht. mSpace Mobile zeichnet sich durch eine neuartige Benutzeroberfläche aus, die es dem Benutzer wahlweise ermöglicht die Karte oder ein anderes Informationselement vergrössert darzustellen, während die anderen in kleinerer Form angezeigt werden.

Des Weiteren wurde im Bereich Communities und mobile Geräte geforscht. Sumi und Mase [18] haben ein System entwickelt, welches die Entwicklung von ad-hoc Communites während Anlässen, wie Tagungen und Ausstellungen, erlauben sollen. Diese sollen, basierend auf ihren gemeinsamen Interessen, Meetings organisieren können. Die „Semantic Map“ erlaubt die Darstellung von Informationen der Community in einem strukturierten Graphen. Während der „Agentsalon“ die persönliche Kontaktaufnahme erleichtern soll.

Ausbaufähigkeit

Da für die gesamte Kommunikation in retino Webservices eingesetzt werden, für die die Schnittstellen offengelegt sind, können sehr leicht Erweiterungen für das System entwickelt werden. Die starke Modularisierung der Einzelkomponenten vereinfacht deren Austausch.

Die Entwicklung eines Clients in Form eines J2ME-Midlets könnte den retino.ws Webservice konsumieren und die Verwendung der meisten aktuellen Mobiltelefone als retino.node erlauben. Um eine genügend grosse Verbreitung von retino zu erreichen, müsste dies mit Sicherheit umgesetzt werden. Während alle aktuell verfügbaren Mobiltelefone J2ME unterstützen, basieren nur wenige Geräte auf Windows Mobile, welches die Installation des .NET Compact Frameworks erlaubt.

Umgekehrt liessen sich aufbereitete Datenquellen des Semantic Web in retino einbringen. Die Informationsvielfalt liesse sich damit mit anderwertig erfassten Daten erweitern. Analog liessen sich Events in retino als Syndication Feed [19] anderen Systemen zur Verfügung stellen.

In Fahrzeugen mit bereits eingebautem Navigationssystem, könnte retino als zusätzlicher Lieferant für Points of Interest dienen. Im Unterschied zu den bereits bestehenden, könnte der Fahrzeuglenker mit retino das System mit aktuellen, dynamischen Informationen versorgt werden.

Fazit

retino zeichnet sich durch die Kombination von verschiedenen Awarenessfaktoren aus, die mittels mobilen Geräten erfasst werden und einer Community zur Verfügung gestellt werden. Die offene Gestaltung der Lösung als modularisiertes Framework basierend auf der Service Oriented Architecture, erlaubt es ohne grossen Aufwand das System zu erweitern. Für einen produktiven Einsatz ist das System aufgrund der ungenügenden Entwicklung von mobilen Geräten noch nicht geeignet. Erst die feste Integration von GPS-Empfänger in Mobiltelefone und längere Akkulaufzeiten werden die effiziente Nutzung des Systems erlauben. Die Anbindung von externen Datenquellen würde für den Erfolg des Systems vor allem in der Startphase eine wichtige Komponente darstellen.

Referenzen

[1] http://ec.europa.eu/dgs/energy_transport/galileo/index_en.htm, 15.05.2006

[2] Martinez Salvador (2004), Context-Awareness in Mobile Systems, Seminar on User Interfaces and Usability, Helsinki Institute for Information Technology, http://www.hiit.fi/uerg/seminaari/T-121900-2004-essay-martinez.pdf

[3] http://de.wikipedia.org/wiki/3-Tier, 16.05.2006

[4] http://de.wikipedia.org/wiki/Web_of_trust, 18.05.2006

[5] Buchmüller Daniel, Ball Rudolf, Hensel Fabian (2004), Projektdokumentation retino, http://retino.icontent.ch/resources/ProjectSpecification.pdf

[6] http://www.swisstopo.ch/de/basics/geo/system/projectionCH, 16.05.2006

[7] http://retino.icontent.ch, 16.05.2006

[8] Rhodes, B. J. & Starner, T. (1996), Remembrance Agent, in The Proceedings of The First International Conference on The Practical Application Of Intelligent Agents and Multi Agent Technology (PAAM '96), pp.487-495.

[9] https://www.google.com/adsense/, 16.05.2006

[10] http://www.microsoft.com/msagent/default.asp, 16.05.2006

[11] Munson, J. P. and Gupta, V. K. 2002. Location-based notification as a general-purpose service. In *Proceedings of the 2nd international Workshop on Mobile Commerce* (Atlanta, Georgia, USA, September 28 - 28, 2002). WMC '02. ACM Press, New York, NY, 40-44.

[12] http://www.golem.de/0507/39567.html, 16.05.2006

[13] N. Marmasse and C. Schmandt, “Locationaware Information Delivery with comMotion,”Proc. 2nd Int’l Symp. Handheld and Ubiquitous Computing (HUC 2K), Lecture Notes in Computer Science, P. Thomas and H.-W. Gellersen, eds., vol. 1927, no. 1, Springer-Verlag, Berlin, 2000, pp. 157–171.

[14] Abowd, G. D., Atkeson, C. G., Hong, J., Long, S., Kooper, R., and Pinkerton, M. 1997. Cyberguide: a mobile context-aware tour guide. *Wirel. Netw.* 3, 5 (Oct. 1997), 421-433.

[15] Cheverst, K., Davies, N., Mitchell, K., and Friday, A. 2000. Experiences of developing and deploying a context-aware tourist guide: the GUIDE project. In *Proceedings of the 6th Annual international Conference on Mobile Computing and Networking* (Boston, Massachusetts, United States, August 06 - 11, 2000). MobiCom '00. ACM Press, New York, NY, 20-31.

[16] A. Hinze and A. Voisard. Location- and time-based information delivery in tourism. In Advances in Spatial and Temporal Databases (SSTD 2003), July 2003.

[17] Wilson, M. L., Russell, A., Smith, D. A., Owens, A. and schraefel, m. c. (2005), mSpace Mobile: A Mobile Application for the Semantic Web. Submitted to End User Semantic Web Workshop, ISWC2005, Galway, Ireland.

[18] Sumi, Y. and Mase, K. 2002. Supporting the awareness of shared interests and experiences in communities. *Int. J. Hum.-Comput. Stud.* 56, 1 (Jan. 2002), 127-146.

[19] http://de.wikipedia.org/wiki/RSS, 16.05.2006