TomTom Navigationssystem als kontextsensitiver Webserver

In dieser Arbeit wird ein TomTom GO Navigationssystem um einen Webserver mit dynamischer Webseiten-Generierung erweitert, so dass dieser mobile Webserver wie ein normaler Webserver über das Internet (und das Telekom Netz) ansprechbar wird. Durch den Zugriff aus dem Internet können zum Beispiel die Positionsinformationen des Navigationsgerätes genutzt werden, um kontextsensitive Anwendungen zu entwickeln. Es wird auch gezeigt, dass der mobile Webserver eine zusätzliche Infrastrukturkomponente benötigt, um die Interaktion in das Netz des Telekomanbieters zu ermöglichen.

Philip Handschin, Jürg Luthiger | juerg.luthiger@fhnw.ch

Navigationsgeräte wie das TomTom [TomTom] sind mobile Kleinrechner mit einer erstaunlichen Funktionalität. Ein leistungsstarker GPS Empfänger und ausgereiftes Kartenmaterial erlauben eine exakte Positionsbestimmung. Diese Kontextinformation wird für das Finden eines Zielortes, für die Planung einer Reiseroute, für die Anzeige interessanter Orte und vieles mehr genutzt. Dank Bluetooth kann über ein mobiles Telefon eine Kommunikationsverbindung in die GSM Infrastruktur und in das Internet aufgebaut werden, um über diese Verbindung Sprache und Daten auszutauschen.

Navigationsgeräte werden in einer verteilten Anwendung ausschliesslich als Clients eingesetzt, obwohl sie in ihrer Rechenleistung, ihrer Speicherkapazität und ihrer Funktionalität zu den Servern der 90er Jahre, dem Beginn des Web Zeitalters, überlegen sind.

Die Idee, ein mobiles Gerät als Webserver zu nutzen, ist nicht neu. In [Pra03] wird eine Architektur für Web Server auf Microsoft Pocket PC präsentiert und in [Wik06] beschreibt Nokia eine interessante Umgebung für mobile Telefone. Ihr Webserver ist eine Portierung des bekannten Apache HTTP Deamon mit Python und WebDAV Unterstützung. Mittels Webzugriff lassen sich zum Beispiel Nachrichten übers Internet direkt aufs Handy schicken, ein auf dem Handy «gehostetes» Blog führen, SMS bequem auf dem Computer tippen oder auch die Kontakte bearbeiten. Dafür muss nur die entsprechende Server-Software auf dem Handy installiert werden. Der Zugriff aus dem Internet auf das Handy erfolgt über den normalen Webbrowser. Das Unternehmen stellt zu Testzwecken und Proof-of-Concept eine gesamte Infrastruktur [Nokia] für die Idee der mobilen Websites zur Verfügung.

Navigationsgeräte bieten die attraktive Möglichkeit, die Positionsdaten des Gerätes in die Anwendung zu integrieren und über diese Kontextinformation neuartige Applikationen zu entwickeln. Diese Idee wurde in einer Informatik Semesterarbeit [Han07] aufgegriffen. Das Ziel der Arbeit war die Umsetzung der Idee aus [Wik06], deren Portierung und die Integration der Positionsinformationen in eine Anwendung für die TomTom GO Plattform [TomTom]. Als Prototyp wurde ein Applikation realisiert, in der die Position des entsprechenden TomTom Gerätes in Google Maps visualisiert und in einem normalen Webbrowser dargestellt wird. So kann zum Beispiel eine Speditionsfirma über die bestehende Internet Informatikinfrastruktur die Position ihrer Lieferwagen verfolgen und benutzergerecht darstellen.

Ein Webserver für die TomTom Plattform

TomTom GO basiert auf einem ARM Linux Betriebssystem. Um auf dieser Plattform eine attraktive Webanwendung realisieren zu können, muss der entsprechende Webserver dynamische Webseiten unterstützen. Denn nur mit dynamischen Webseiten ist möglich, die Position jederzeit dem aktuellen Standort nachführen zu können. Die Wahl fiel schliesslich auf den Klone [Klone] Webserver. Klone ist ein Multiplattform-Webserver speziell für Embedded-Geräte entwickelt, der dank C/C++-Scripting auch dynamische Inhalte erlaubt, ohne auf zusätzliche Komponenten wie PHP oder Perl angewiesen zu sein. Durch den Verzicht auf Erweiterungen des Websevers wie PHP soll vor allem die Leistung gesteigert und die CPU-Auslastung gesenkt werden.

Mittels des Software Development Kit (SDK) können Entwickler ihre dynamischen Inhalte in C oder C++ schreiben. Normale HTML Seiten werden durch sogenannte Scriptlets ergänzt. Analog wie bei den Java Server Pages umfassen diese Scriptles den dynamischen Teil, der in diesem Falle jedoch in C/C++ geschrieben wird.

Das Resultat der dynamische Webpage aus Listing 1 ist eine Auflistung des Root Ordners. Ein spezieller Compiler übersetzt anschliessend den Quelltext in nativen Code, der dann als Bina-

```
<%!
    #include <string.h>
    #include <sys/types.h>
    #include <dirent.h>
%>
<html>
<head>
    <title>My first page</title>
    <link rel="stylesheet" href="kl.css" type="text/css">
</head>
<body>
    <h3>Root dir content:</h3>
    <ul>
    <%
        DIR *dirp;
        struct dirent *dp;

        dirp = opendir("/");
        while ((dp = readdir(dirp)) != NULL)
            io_printf(out, "<li>%s</li>", dp->d_name);
        closedir(dirp);
    %>
    </ul>
</body>
</html>
```

Listing 1: Beispiel einer dynamischen Webpage mit dem Klone Webserver

ry gegen den Klone Server gelinkt und integriert werden kann.

Kommunikationsinfrastruktur

Ein Webserver auf der TomTom Plattform macht erst dann Sinn, wenn dieser Server auch aus dem Internet angesprochen werden kann. Ein explizites Ziel dieser Arbeit war es, dass ein beliebiger Webbrowser den TomTom Server jederzeit über das Internet adressieren und nutzen kann. In [Wik06] wurde gezeigt, dass diese Aufgabe auch über gängige Telekommunikationsnetze wie GSM oder GPRS erreicht werden kann. Der Zugang in solche Kommunikationsnetze ist aber nicht trivial, da normalerweise ein Verbindungsaufbau von aussen in das Netz des Telekomanbieters über Firewalls und NAT Systeme (Network Address Translation) blockiert wird. Diese Restriktion kann jedoch umgangen werden, wenn die Verbindung von einem Gerät initiiert wird, das sich im Telekomnetz und hinter dem Firewall befindet. Um diesen Verbindungsaufbau entgegen nehmen zu können, muss aber ein zusätzlicher Gateway vor dem Firewall, also im Internet, installiert werden. Dieser Gateway ist die Schnittstelle zwischen dem Telekom-Netz und dem Internet. Er muss, wie in Abbildung 1 dargestellt:

- den Verbindungsaufbau vom mobilen Geräten entgegennehmen;
- die Verbindung zum mobilen Gerät verwalten;
- die Anfrage des Browser entgegennehmen;
- die Kommunikation zwischen Webbrowser und mobilem Endgerät herstellen.

Der HTTP Request vom Browser zum mobilen Webserver erfährt mit diesem Lösungsansatz eine Umleitung über den Gateway. Da dieser Gateway ausserhalb des Telekom-Netzes steht, ist er jederzeit über das Internet erreichbar. Der Gateway kann deshalb die Anfrage entgegennehmen und diese über eine proprietäre Socket Verbindung zum entsprechenden mobilen Webserver weiterleiten. Die Socket Verbindung muss vorher jedoch vom mobilen Webserver aufgebaut werden, was zum Beispiel bei einer Anmeldung beim Gateway erfolgen kann. Sobald die Socket Verbindung steht, ist der mobile Webserver über das Internet ansprechbar. Um aus dem Internet adressierbar zu werden, muss der mobile Webserver eine eindeutige Identifikation erhalten.

TomTom Webserver Identifikation im Internet

Wie bereits beschrieben, wird sich der TomTom Benutzer am Gateway anmelden müssen, um das mobile Gerät im Internet sichtbar machen zu können. Nach der Anmeldung ist eine proprietäre, aber eindeutige Verbindung über TCP/IP Sockets zwischen dem mobilen Gerät und dem Gateway vorhanden. Der Gateway muss alle Verbindungen zu den entsprechenden mobilen Geräten verwalten können. Der TomTom Handler wird diese Aufgabe übernehmen. Er wurde in dieser Arbeit konzipiert und implementiert. Der Partner auf der mobilen Seite ist der Gateway Connector. Er stellt die Interaktion mit dem mobilen Webserver her.

Für die Verwaltung der Verbindung erstellt der TomTom Handler eine HandlerMap. In dieser werden alle aktiven TomTom Verbindungen eingetragen. Als Schlüssel kann zum Beispiel der übermittelte Name des TomToms verwendet werden. Der Vorteil dieses Ansatzes ist es, dass der Name nur einmal vorkommen darf und deshalb auch für

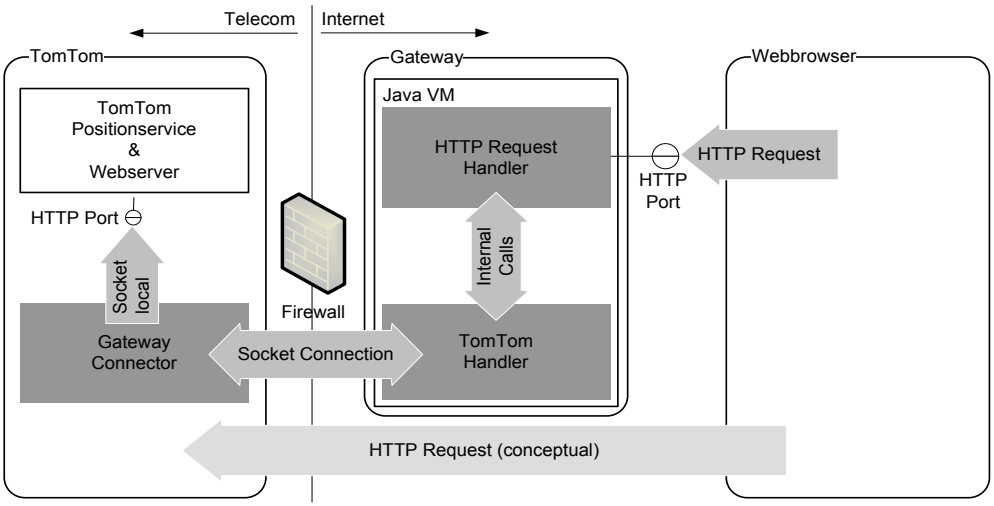

Abbildung 1: Das Gateway Konzept

die Generierung einer eindeutigen URL eingesetzt werden kann. Sobald der Gateway eine TomTom Anmeldung akzeptiert hat, wird die Verbindung über einen eigenen Thread kontrolliert.

Im Internet ist das TomTom Gerät anschliessend unter der HTTP Adresse `http://<gateway-host>:<gateway-port>/<tomtom-name>` sicht- und ansprechbar.

Kommunikation zwischen Browser Client und TomTom Webserver

In Abbildung 2 sind die wichtigen Kommunikationsverbindungen aufgezeichnet.

Verbindung 1:
Die Verbindung 1 ist eine TCP/IP Socket Verbindung zwischen dem mobilen Gerät und dem Gateway. Sie wird beim Starten des Gateway Connectors auf dem TomTom aufgebaut. Dieser Verbindungsaufbau entspricht der Anmeldung auf dem Gateway unter einem eindeutigen Namen.

Verbindung 2:
Die Verbindung 2 zeichnet den HTTP Request nach. Der HTTP Request kann nicht direkt an den mobilen Webserver erfolgen, da er hinter der Firewall des Telekom Anbieters versteckt ist. Der HTTP Request muss deshalb über das Internet an den HTTP Port des Gateways geleitet werden. Hier kann der HTTP Request Handler aus der URL das TomTom detektieren, das angefragt ist. Der TomTom Handler übernimmt den Request und leitet die Anfrage über die bestehende TCP/IP Socket Verbindung 1 an den Gateway Connector des TomTom's weiter. Vom Gateway Connector gelangt der HTTP Request schliesslich zum mobilen Webserver. Der mobile Webserver liest mit Hilfe des Positionsservices die aktuelle Position des mobilen Gerätes aus und generiert die HTML Antwort.

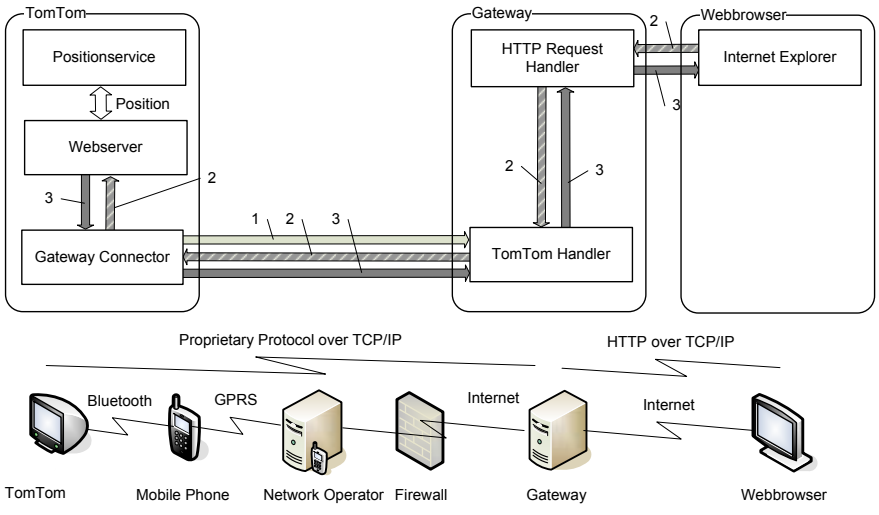

Abbildung 2: Kommunikation zwischen Browser und mobilem Web Server

Verbindung 3:
Die HTML Response verfolgt den gleichen Verbindungsstrang in umgekehrter Richtung wie der HTTP Request, um anschliessend im Webbrowser als Webseite dargestellt zu werden.

In der Abbildung 2 sind die notwendigen Hardware Komponenten aufgezeichnet, die es für diese Kommunikationsaufgabe braucht. Das TomTom baut mit Hilfe eines mobilen Telefons, das über Bluetooth gekoppelt ist, die GPRS Verbindung in das Internet, zum Gateway auf.

Die Positionsdaten des TomTom können nun auf der Client Seite ausgewertet werden, die z. B. mit entsprechendem Kartenmaterial visualisiert werden.

Abbildung 3: Tomsite mit dem BMW Zeichen als Positionsmarkierung

Visualisierung der TomTom Position im Browser Client

Google Maps bietet nicht nur schöne Karten und Satellitenbilder an, sondern auch eine einfach zu nutzende Programmierschnittstelle. Der Einsatz von Google Maps zur Visualisierung der TomTom Position bringt den grossen Vorteil, dass die aufwändigen Kartendaten unabhängig vom mobilen Navigationsgerät zur Verfügung stehen. Durch die externe Verknüpfung der Positionsdaten mit dem Kartenmaterial kann auf dem mobilen Gerät das kostenintensive Verwalten der Kartendaten weggelassen werden. Zwischen Webbrowser und mobilen Webserver werden lediglich die Positionsdaten ausgetauscht.

Über eine Programmierschnittstelle (API) können Webentwickler die Google Karten in ihre eigenen Websites einbauen. Dadurch werden die Funktionen der Google Maps auf der eigenen Seite nutzbar. Zusätzlich können in den Karten eigene Markers gesetzt werden, um auf die Weise eine Position explizit erkennbar zu machen (Abbildung 3). Listing 2 stellt die entsprechende Umsetzung für den Klone Webserver dar.

```
<%!
    // some includes, variable definitions, ...
%>
<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>TomTom Position Service</title>
    <script src="http://maps.google.com/maps?file=api&amp;v=2&amp;
            key=<my google key>" type="text/javascript"></script>

    <script type="text/javascript">

    //<![CDATA[
    function load() {
      if (GBrowserIsCompatible()) {
        var map = new GMap2(document.getElementById("map"));
        map.addControl(new GSmallMapControl());
        map.addControl(new GMapTypeControl());
        map.setCenter(new GLatLng(
            <% // call to the mobile webserver which returns the
               // position of the TomTom device %>
            ,true),13);

        var icon = new GIcon();
        icon.image = "bmwlogo.png";
        icon.iconSize = new GSize(50 ,47);
        icon.iconAnchor = new GPoint(31, 15);
        icon.infoWindowAnchor = new GPoint(5, 1);

        function createMarker(point, name) {
            var marker = new GMarker(point,icon);
            GEvent.addListener(marker, "click", function() {
                marker.openInfoWindowHtml("Position von <b>" + name +
                "</b>");
            });
            return marker;
        }
```

```
          marker = createMarker(map.getCenter(),"My Car");
          map.addOverlay(marker,icon);

          window.setTimeout(function() {
              location.reload()
          }, 30000);

        } // end if()
      } // end load()

      //]]>
      </script>
  </head>

  <body onload="load()" onunload="GUnload()">
      <p>TomTom Position Service</p>

      <div id="map" style="width: 500px; height: 500px"></div>

      <p>Diese Seite wird alle 30 Sekunden automatisch aktualisiert.</p>
      <p>F&uuml;r Manuelle Aktualisierung, bitte Knopf dr&uuml;cken:
          <input type="button" value="Aktualisieren"
          onClick="location.reload()"
          class="altButtonFormat"></p>

  </body>
  </html>
```

Listing 2: Integration von Google Maps in die dynamische Webpage

Herausforderungen

In diesem Abschnitt werden ein paar Aspekte aufgegriffen, die typisch für eine Tomsite sind und eine nähere Betrachtung bedürfen. Es sind dies das Offline Verhalten, die Kosten und die Sicherheit, bzw. Datenschutz.

Offline

Im Gegensatz zu einer normalen Website wird eine Tomsite nicht immer online sein. Es ist eine typische Eigenschaft einer mobilen Site, dass sie oft nicht ansprechbar ist, weil z.B. das Navigationsgerät nicht in Betrieb oder die entsprechende Verbindung in das Internet nicht aufgebaut ist. Wäre die Tomsite direkt aus dem Internet adressierbar, hätte der Entwickler fast keine Möglichkeiten diesen offline Modus benutzerfreundlich und aussagekräftig darzustellen. «Server not found» ist in dieser Situation die normale Fehlermeldung.

Dank der Umleitung über den Gateway aber, er wird als normaler Webserver auf eine hohe Verfügbarkeit ausgelegt, kann mit der vorliegenden Infrastruktur eine benutzergerechte Antwort gegeben werden. Aufgrund der An- und Abmeldung weiss der Gateway wann eine Tomsite online ist. Im offline Status kann der Gateway diesen Zustand mit einer aussagekräftigen HTML Response beantworten.

Findet ein Verbindungsunterbruch ohne Abmeldung statt, erkennt dies der Gateway erst bei einem konkreten Aufruf der entsprechenden Tomsite. In diesem Falle kann ein Timeout abgewartet werden, um anschliessend alle vorhandenen Informationen aus dem Gatway zu entfernen und die Tomsite als offline zu markieren.

Kosten

Die Verbindung aus dem Telekomnetz kostet. Die Kosten berechnen sich aus der kommunizierten Datenmenge.

In einem Feldversuch wurde deshalb das System über eine Zeitperiode von 2 Stunden aktiv genutzt. In einer ersten Phase wurde alle 30 Sekunden eine Abfrage abgesetzt; total 120 * 2 = 240 Abfragen. Der anschliessenden Stresstest mit 80 Anfragen kurz hintereinander, führten zu insgesamt 320 Tomsite Anfragen. Das Mobiltelefon musste in dieser Zeitspanne 0.7 Megabytes (MB) senden und 0.3 MB empfangen, das sind 1 MB Datenverkehr. Mit den Tarifen von Sunrise (Erste Hälfte 2007) würde dies ca. Fr. 3.50 kosten.

Sicherheit, Datenschutz

Da das TomTom Navigationsgerät persönliche Daten speichern kann, ist es wichtig, dass der Besitzer die Kontrolle darüber hat, wer seine Tomsite nutzen darf und wer nicht. Deshalb sollte die Tomsite ein entsprechendes Administrationsinterface zur Verfügung stellen, über das der Benutzer eine einfache Benutzeradministration zur Verfügung hat. Da die vorliegende Arbeit jedoch als Machbarkeitsstudie ausgelegt ist, wurde das Thema Sicherheit, Datenschutz bewusst aus der Aufgabestellung gestrichen.

Zusammenfassung

In dieser Arbeit wurde gezeigt, dass es ist möglich ist, einen voll funktionsfähigen Webserver mit dynamischer Unterstützung auf einem TomTom GO Gerät zu installieren und auszuführen. Mit Hilfe eines Gateways, der die Verbindung zwi-

schen dem Netz des Telekomanbieters und dem Internet herstellt, wird diese sogenannte Tomsite wie eine normale Website von einem Browser aufrufbar.

Es stellt sich hier die Frage, ob der Gateway, der wie der mobile Webserver ebenfalls einen HTTP Port für die HTTP Requests zur Verfügung stellen muss, nicht auch die Webserver Funktionalität des mobilen Gerätes übernehmen kann. Der Gateway und das mobile Geräte nutzen eine Socket Verbindung, um die Kommunikation zwischen Telekomnetz und Internet sicherstellen zu können. Diese Verbindung kann teuer sein, da sie über die kommunizierte Datenmenge abgerechnet wird. Bei der Entwicklung der mobilen Applikation muss man deshalb den notwendigen Datentransfer beachten. Im Extremfall kann die Generierung der HTML Seite deshalb auch auf den Gateway ausgelagert werden. Das TomTom wird in einem solchen Falle nur die Positionsdaten und nicht die gesamte Webseite an den Gateway senden, was den Datentransfer zwischen diesen beiden Partner stark reduziert. Dieses Vorgehen ist aber nur dann sinnvoll, wenn alle über den Gateway angeschlossenen mobilen Geräte die gleiche internetfähige Applikation bereitstellen. Sobald verschiedene Anwendungen über den Gateway abgehandelt werden, wird der Aufwand auf dem Gateway gross, die unterschiedlichen Inhalte performant aufzubereiten und zu verwalten. In dieser Situation ist ein dezentralisierter Ansatz und eine klare Aufgabenteilung sinnvoller, da er einfacher wartbar, erweiterbar und flexibler ist. Ebenfalls ist der Einsatz eines mobilen Webservers unumgänglich, wenn das mobile Endgerät auch über WLAN eine Verbindung in das Internet aufbauen kann und in diesem Falle ein Gateway nicht mehr notwendig ist.

Der mobile Webserver auf einem Navigationsgerät ist vor allem für kontextsensitive Applikationen interessant, welche die Positionsinformationen des Gerätes nutzen möchten. Dies wurde beispielhaft mit einer Integration der Fahrzeugposition in Google Maps demonstriert.

Leider hat TomTom in der Zwischenzeit die Entwicklung des Software Development Kit (SDK) für die TomTom GO Plattform eingestellt. Das SDK stellte dem Programmierer eine umfangreiche Bibliothek zur Verfügung, um auf die spezifischen Funktionen des Linux-basierten Navigationssystems zugreifen zu können, um zum Beispiel die GPS Position auszulesen.

Wir sind jedoch überzeugt, dass solche Ideen und Ansätze in Zukunft vermehrt Anwendung finden werden. Diese Arbeit hat uns gezeigt, dass die technischen Möglichkeiten bereits heute vorhanden sind.

Referenzen

[Pra03] Pratistha D., Nicoloudis N., Cuce S., A Micro-Services Framework on Mobile Devices, International Conference on Web Services, 2003

[Wik06] Wikman J., Dosa F., Tarkiainen M., Personal Website on a Mobile Phone, Nokia Research Center Helsinki, 2006

[Han07] Handschin P., TomSite, Semesterarbeit Studiengang Informatik, Fachhochschule FHNW, 2007

[Klone] KoanLogic, Klone Embedded Webserver, http://koanlogic.com/klone/index.html

[Nokia] Nokia, Mobile Web Server, http://mymobilesite.net/

[TomTom] TomTom International, http://www.tomtom.com