Subscribable Web

Studiengang: BSc in Informatik | Vertiefung: Mobile Computing
Betreuer: Dr. Reto Koenig
Experte: Dr. Joachim Wolfgang Kaltz

Seit seiner Entstehung vor knapp 30 Jahren, hat das World Wide Web einen beispiellosen Siegeszug erlebt. Wenn man aber die ursprünglichen Ziele der Entwickler des Web betrachtet, fällt auf, wie stark sich seither die Anforderungen an das Web gewandelt haben. Heutzutage verlangen wir nicht nur statische Dokumente, sondern auch Realtime-Informationen. Hier hat die vorliegende Arbeit angesetzt und untersucht, wie ein «Subscribable Web» aussehen könnte.

Sollen Dokumente über das Internet zur Verfügung gestellt werden, wird heute in den allermeisten Fällen das World Wide Web verwendet. Der Zugriff auf Dokumente im Web basiert auf einem Request-Response-Modell. Das bedeutet, dass Benutzer das gewünschte Dokument von einem Server anfordern müssen, indem ein Request an den Server gesendet wird. Der Server sendet dann das Dokument als Response an die Benutzer zurück. Falls das Dokument nicht auf dem Server vorhanden ist, oder Benutzern die Berechtigung fehlt, das Dokument zu erhalten, wird anstelle des Dokuments eine Fehlermeldung als Response zurückgesendet.
Das Request-Response-Modell des Web eignet sich gut für statische Dokumente, die sich nie ändern. Benutzer wissen immer, dass sie die korrekte Version des Dokuments besitzen, da nur eine Version des Dokuments existiert. Wenn sich aber ein Dokument im Laufe der Zeit verändern kann, ist nie klar, ob die Version des erhaltenen Dokuments nicht bereits veraltet ist.
Da die Kommunikation im Request-Response-Modell immer vom Benutzer aus geht, muss der Server immer wieder angefragt werden, ob sich das Dokument seit der letzten Anfrage geändert hat. In den meisten Fällen wird sich das Dokument nicht geändert haben und die Anfrage wird somit nur Ressourcen verschwenden. Der Vorgang, bei dem der Server regelmässig vom Benutzer abgefragt wird, ist als Polling bekannt. Wenn die Zeitabstände des Pollings sehr kurz gewählt werden, kann die Netzwerkbelastung stark ansteigen. Werden deshalb die Zeitabstände aber vergrössert, kann eine Änderung des Dokuments lange unentdeckt bleiben.
Eine Lösung für dieses Dilemma bietet das Publish-Subscribe-Modell. Bei diesem Modell verlangen die Benutzer nicht einmalig ein einzelnes Dokument, sondern abonnieren sich auf das Dokument. Dadurch erhalten die Benutzer bei jeder Änderung des Dokuments automatisch die neue Version, ohne selber aktiv werden zu müssen.

Beim Publish-Subscribe-Modell kommt statt eines Servers ein Vermittler zwischen Dokumentbesitzer und Benutzer zum Zug, der sogenannte Broker. Wenn der Benutzer ein Dokument erhalten möchte, sendet er einen Subscribe für dieses Dokument an den Broker. Der Dokumentbesitzer sendet die jeweils aktualisierte Version des Dokuments als Publish ebenfalls an den Broker. Die Aufgabe des Brokers ist es, diese Relation korrekt zu bedienen und so die Benutzer stets auf dem aktuellen Stand zu halten.
In dieser Arbeit wurde untersucht, ob im heutigen Web ein Publish-Subscribe-Modell verwendet werden könnte. Um dieses Subscribable Web zu simulieren, wurde ein Proof-of-Concept erstellt, der es erlaubt, sich auf Dokumente im Web zu subscriben. Zusätzlich konnten so verschiedene Arten von Webauftritten untersucht und evaluiert werden, wie weit sie sich für das Publish-Subscribe-Modell eignen würden.
Für den Proof-of-Concept wurde MQTT als Publish-Subscribe-Protokoll verwendet. Da heutige Browser nur mit dem auf Request-Response basierenden HTTP umgehen können, musste per JavaScript eine Protokollumsetzung geschrieben werden, damit mit einem MQTT-Broker kommuniziert werden kann.
Um untersuchen zu können, wie sich Webauftritte unter einem Publish-Subscribe-Modell verhalten würden, musste ein Proxy-Server erstellt werden. Dieser Proxy-Server ist in der Lage, über Request-Response zur Verfügung gestellte, Dokumente im Web zu pollen. Wenn sich diese Dokumente ändern, werden sie vom Proxy-Server an den MQTT-Broker gesendet.
Der Proof-of-Concept konnte zeigen, dass viele Webauftritte ohne grossen Aufwand über Publish-Subscribe zur Verfügung gestellt werden könnten. Durch die direkte Verwendung des Publish-Subscribe-Modells könnten viele Workarounds wegfallen, die genau darauf ausgelegt sind, ein solches Publish-Subscribe über client-seitige Scripts zu emulieren.

Stephan Schär