Universität Salzburg, UNIGIS

ÜBERMITTLUNG VON SENSORDATEN ZU WEBANWENDUNGEN

Stefan Scheuber

Zusammenfassung: Dieser Beitrag vergleicht zwei Varianten, die es ermöglichen, ortsbezogene Sensordaten zu einer Webanwendung zu übermitteln. Bei den beiden Übertragungsvarianten handelt es sich einerseits um den Sensor Observation Service (SOS) und andererseits um das WebSocket-Protokoll. Vergleichsmessungen haben gezeigt, dass beide Varianten zuverlässig die Daten übertragen. Bei Performance-Messungen wies die WebSocket-Variante aber eine kleinere Latenzzeit, eine geringe Menge übertragener Daten und einen höheren maximalen Meldungsdurchsatz auf als die SOS-Variante. Die WebSocket-Variante überzeugte somit bei den Vergleichsmessungen in den Punkten Echtzeitfähigkeit und Performanz.

Schlüsselwörter: Nah-Echtzeit, ortsbezogene Sensordaten, Sensor Observation Service (SOS), WebSocket

SENSOR DATA TRANSMISSION TO WEB APPLICATIONS

Abstract: This paper presents two solutions, which allow to transmit spatial sensor data to a web application. The first solution is based on a Sensor Observation Service (SOS), the second solution uses a WebSocket to transmit the sensor data to the web application. Comparative measurements have shown, that both solutions are able to transmit data reliably. The comparison has further shown, that the WebSocket solution has a smaller latency, the amount of transmitted data is reduced and the maximum message rate is greater than that of the SOS solution. Thus, the comparative measurements have demonstrated, that the WebSocket solution has advantages in real-time capability and performance.

Keywords: Near-Realtime, spatial sensor data, Sensor Observation Service (SOS), WebSocket

Autor
B. Sc. CS / M. S.c GIS Stefan Scheuber
Schafmattweg 13
CH-4102 Binningen
E: dev@stue.ch

1 EINLEITUNG

Im Jahre 2011 nannte Goodchild in einem Beitrag über zukünftige Entwicklungen in Geoinformationssystemen die Echtzeitverarbeitung und -Darstellung von ortsbezogenen Daten als einen der möglicherweise bedeutendsten Trends (Goodchild 2011). Er beschreibt darin den Fortschritt vom Analysieren statischer Daten hin zu deren dynamischer Verarbeitung, was einen Wandel in der Echtzeitentscheidungsfindung und -überwachung herbeiführen könnte. Der von Goodchild beschriebene Trend hat sich in den letzten Jahren durch immer kleiner und günstiger werdende Sensoren, deren Einsatz stetig in weiteren Bereichen Anwendung findet, verstärkt. Durch die Anbindung von Sensoren an das Internet werden Daten nicht einmalig erhoben und ausgewertet, vielmehr werden durch die Sensoren konstant Daten erzeugt und übertragen. Möchte man die Sensordatenströme einer breiten Öffentlichkeit zugänglich machen, bietet es sich an, diese auf einer Webanwendung darzustellen. Der Anwender kann somit, ohne zusätzliche Programme zu installieren, die Sensordaten visualisiert auf einer Webseite betrachten. Die rasant fortschreitende Entwicklung von Webtechnologien erlaubt es mittlerweile, Webanwendungen zu schreiben, die vom Verhalten und Funktionsumfang her Desktopapplikationen ähneln (Fraternali et. al 2010). Der Nutzen dieser Webanwendungen wird zudem durch die Verwendung möglichst verlässlicher und aktueller Daten erhöht. Verschiedene Technologien wurden hinsichtlich der Verarbeitung von Sensordatenströmen als auch für die Nah-Echtzeitübertragung zu einer Webanwendung entwickelt.

Das Open Geospatial Consortium (OGC) hat sich mit der Sensor Web Enablement (SWE)-Initiative zum Ziel gesetzt, webzugängliche Sensornetzwerke und archivierte Sensordaten mithilfe von offenen Standards aufzufinden, den Zugriff auf die Daten und Sensoren sicherzustellen, wenn möglich die Sensoren ferngesteuert zu kontrollieren und den Zugriff auf die Sensoren und Sensordaten mit offenen Standards zu vereinheitlichen (Botts et. al 2008). Mit dem Sensor Observation Service (SOS) wurde eine generell gehaltene Serviceschnittstelle spezifiziert, die für das Speichern und Laden von Sensormetadaten und -messwerten verwendet werden kann (Na & Priest 2007). Die Kommunikation eines Clients mit einem SOS erfolgt synchron, wobei der SOS nie die Kommunikation initiiert. Möchte ein Client somit Daten von einem SOS beziehen, muss eine entsprechende Anfrage an den Service gestellt werden, worauf der SOS mit den angeforderten Daten antwortet. Ist der Client an den jeweils aktuellsten Daten interessiert oder möchte er beispielsweise wissen, ob ein Zielwert erreicht wurde, muss er laufend Anfragen an den SOS stellen, um die aktuellsten Daten zu erhalten. Diese Methode, an aktuelle Daten zu gelangen, wird *Polling* genannt. Eine Webanwendung kann mit entsprechenden Serviceaufrufen Daten direkt von einem SOS abfragen und weiterverarbeiten.

Mit dem Sensor Alert Service (SAS) und dessen Weiterentwicklung, dem Sensor Event Service (SES), ist eine asynchrone Kommunikation möglich (Simonis 2006, Echterhoff & Thomas 2008). Durch diese Art der Kommunikation können das Polling und dadurch unnötige Serviceaufrufe verhindert werden. Hierfür registriert sich ein Client mit einer Anfrage bei einem SAS oder SES und wird ab diesem Zeitpunkt laufend mit neuen Werten benachrichtigt (notifiziert). Zudem kann bei der Registrierung ein Filter definiert werden, sodass nur bei bestimmten Ereignissen Notifikationen zum Client gesendet werden. Sowohl der SAS als auch der SES eignen sich aber nicht als Service für eine Webanwendung, da Notifikationen, die beim SAS über das Extensible Messaging and Presence Protocol (XMPP) und beim SES an einen Callback-Service gesendet werden, von Internetbrowsern nicht unterstützt und verarbeitet werden können. Eine geeignete Lösung, um Ereignisse ohne Polling von einem Server zu einer Webanwendung zu senden, war für lange Zeit nicht oder nur mit Einschränkungen verfügbar, da ursprünglich im Hypertext Transfer Protocol (HTTP) keine asynchrone Kommunikation vorgesehen war. Ein Workaround wurde durch HTTP-Long-Polling und HTTP-Streaming geschaffen, die durch eine weite Auslegung des HTTP eine „asynchrone" Kommunikation zu Webanwendungen ermöglichen (Loreto et. al 2011). Die weite Auslegung des HTTP-Standards beim Einsatz von HTTP-Long-Polling und HTTP-Streaming, die manche sogar als missbräuchlich betiteln, kann jedoch zu Verzögerungen in der Übertragung als auch zu Performanzproblemen führen.

Im Jahr 2011 wurde schließlich das WebSocket-Protokoll veröffentlicht. Dieses erlaubt eine asynchrone bidirektionale Kommunikation zwischen einem Server und einer Clientanwendung (Fette & Melnikov 2011). Das WebSocket-Protokoll wird von den gebräuchlichsten Internetbrowsern unterstützt (Deveria 2015), wodurch es Webanwendungen eine asynchrone Kommunikation mit einem Server erlaubt.

In diesem Beitrag werden zwei Varianten vorgestellt, mit denen Sensordaten möglichst in Echtzeit zu einer Webanwendung übertragen und dort auf einer Kartenkomponente grafisch dargestellt werden können. Die Übertragung wurde bei der einen Variante über einen SOS umgesetzt, wobei die Datenbeschaffung der Webanwendung durch Polling bei einem SOS erfolgt. Bei der anderen Variante erfolgt die Kommunikation zwischen Server und Webanwendung asynchron über das WebSocket-Protokoll. Anschließend wird durch einen Parallelbetrieb der beiden Varianten eine Auskunft über die Zuverlässigkeit der beiden Varianten gegeben und in einem Vergleich die Nah-Echtzeitmöglichkeiten, der maximale Meldungsdurchsatz sowie die Übertragungsmenge verglichen.

2 ARCHITEKTUR DER VARIANTEN

Die beiden betrachteten Varianten zur Übermittlung von Sensordaten unterscheiden sich grundlegend im technologischen Aufbau, den verwendeten Datenformaten und in der Art, wie die Meldungen vom Zeitpunkt des Empfangs beim Serversystem zum Client übertragen werden.

2.1 SOS-VARIANTE

Die betrachtete Variante der Übertragung von Sensordaten zu einer Webanwendung mittels SOS orientiert sich am Konzept, das Jirka & Bredel (2011) erarbeitet haben, um Schiffsposition in Nah-Echtzeit zu visualisieren. Die empfangenen Daten werden, wie auch bei Jirka & Bredel, in einen SOS eingefügt, wodurch diese über eine standardisierte Schnittstelle laufend wieder bezogen werden können.

Wie in Abbildung 1 aufgezeigt, werden bei dieser Variante die auf dem Server empfangenen Sensormesswerte mittels einer *SOS:InsertObservation*-Meldung an den SOS-Server gesendet und auf diesem abgespeichert. Mittels einer *SOS:GetObservation*-Meldung können diese wieder abge-

fragt werden. Die für diesen Beitrag umgesetzte Webanwendung führt bei dieser Variante in einem definierbaren Intervall SOS:*GetObservation*-Anfragen an den SOS-Server aus um aktuelle Daten anzufordern. Für die Kommunikation zwischen Server und SOS-Server sowie zwischen Webanwendung und SOS-Server werden die Messwerte gemäß dem *Observations and Measurements* (O&M)-Datenformat beschrieben (Cox 2011a). Die Messwerte werden schließlich als XML-Zeichenketten, die gemäß der O&M XML-Implementierung formatiert sind, übertragen (Cox 2011b).

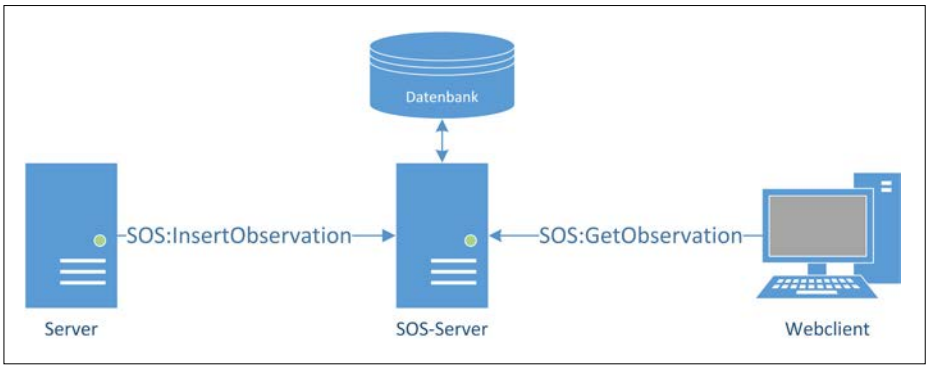

Abbildung 1: Übersicht SOS-Übertragung

2.2 WEBSOCKET-VARIANTE

Pimentel & Nickerson (2012) haben eine WebSocket-Lösung vorgestellt, mit der Echtzeitdaten eines Windsensors über WebSocket übertragen werden. Die zweite in diesem Beitrag entwickelte Variante zur Übertragung von Sensordaten zu einer Webanwendung orientiert sich an dieser Lösung.

Abbildung 2: Übersicht Übertragung WebSocket

Die auf dem Server empfangenen Sensordaten werden dabei direkt durch das WebSocket-Protokoll an die Webanwendung gesendet. Wie in Abbildung 2 dargestellt, steht in der WebSocket-Variante der Client direkt mit dem Server in Verbindung. Die Webanwendung muss lediglich initial eine Verbindung über das WebSocket-Protokoll mit dem Server herstellen. Anschließend sendet der Server direkt Daten zur Webanwendung, ohne dass diese in regelmäßigen zeitlichen Abständen Abfragen auf dem Server durchführen muss.

In der für diesen Beitrag umgesetzten Variante werden serverseitig die empfangenen Sensormesswerte in GeoJSON-Zeichenketten umgewandelt und anschließend in das WebSocket geschrieben. GeoJSON (Butler et. al 2008) basiert auf den Formatierungsregeln JSON (Crockford 2006). Im Vergleich zum XML-Standard weist JSON bei der Übertragung von Daten als auch bei deren Verarbeitung erhebliche Performancevorteile auf (Nurseitov et. al 2009).

3 UMSETZUNG VERGLEICHSSYSTEM

Das für diesen Beitrag verwendete System, mit welchem die in Abschnitt 2 vorgestellten SOS- und WebSocket-Varianten verglichen werden sollen, besteht aus einem Server- und einem Clientteil. Die Aufgabe des Serverteils ist es, die von einem Sensor empfangenen Daten in das System zu integrieren und über die beiden Varianten zur Webanwendung zu übertragen. Aus diesem Grund wird das Serversystem fortan als Integrationssystem bezeichnet. Die Aufgabe des Clientteils ist es, die Daten über beide Varianten in einer Webanwendung zu empfangen und darzustellen. Der Rechner, auf welchem die Webanwendung ausgeführt wird, wird als Webclient bezeichnet. Das in diesem Beitrag verwendete Integrationssystem als auch die Webanwendung entstanden im Rahmen der Masterthesis des Autors und stehen zur freien Nutzung zur Verfügung (Scheuber 2015). Die Quelltexte können auf GitHub (GitHub 2016) bezogen werden und unterliegen der liberalen MIT Open-Source-Lizenz (MIT 1988).

3.1 INTEGRATIONSSYSTEM

Das verwendete Integrationssystem ist in der Programmiersprache Java geschrieben und dient der Transformation der Sensordaten in ein einheitliches Format sowie deren anschließender Verteilung über die SOS- und WebSocket-Variante. Wie in Abbildung 3 illustriert wird, ist das Integrationssystem grundsätzlich in Datenempfangs- und Datenversandmodule unterteilt. Die Aufgabe der Datenempfangsmodule ist es, die Anbindung an eine Datenquelle sicherzustellen sowie die empfangenen Daten in ein einheitliches sowie generisches Format zu konvertieren. Diese Objekte werden zur Weiterverarbeitung an die Datenversandmodule übergeben. Dort können diese Objekte mithilfe entsprechender Technologien übertragen werden (z. B. zu einer Webanwendung). Somit enthalten nur die Empfangsmodule domänenspezifische Verarbeitungsroutinen. Das Integrationssystem kann durch die Einbindung weiterer Module nach Bedarf ergänzt werden, um Daten von weiteren Quellen/Sensortypen einzulesen und/oder über weitere Technologien

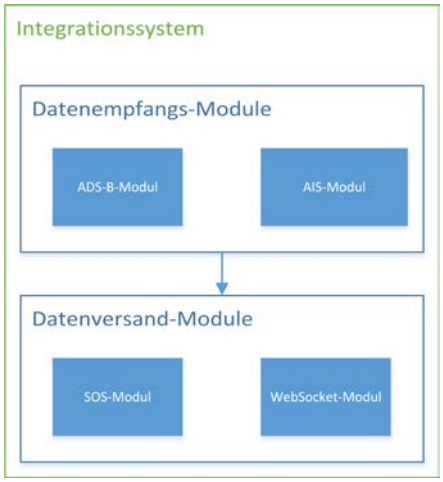

Abbildung 3: Aufbau Integrationssystem

Attribut	Typ	Einheit	Beispiel
Identifikation	Zeichenkette	–	4CA4E5
Sendezeitpunkt	Zeitpunkt	Unix Timestamp	1432233926692
Rufzeichen	Zeichenkette	–	RJA1118
Höhe	Zahlenwert	Fuss	37000
Geschwindigkeit	Zahlenwert	Knoten	350
Flugrichtung	Zahlenwert	Grad (0-360)	254
Position	Position	WGS84-Koordinate (lon/lat)	7.573060/47.536420

Tabelle 1: Übertragene Attribute

zu versenden. Zudem können durch die gewählte Konfiguration des Integrationssystems einzelne Module für den Betrieb deaktiviert resp. aktiviert werden.

Für das betrachtete Integrationssystem wurden bisher zwei unterschiedliche Datenempfangsmodule implementiert. Das Automatic Dependent Surveillance-Broadcast (ADS-B)-Modul dient dem Empfang und der Weiterverarbeitung von Flugzeugpositionsdaten. Mit dem AIS-Modul können Schiffspositionen und Informationen, die durch ein Automatic Identification System (AIS) versendet wurden, decodiert und in generische Objekte umgewandelt werden.

Für die beiden Übertragungsvarianten wurden bislang zwei Datenversandmodule, das SOS- und das WebSocket-Modul, implementiert. Jede dem SOS-Modul übergebene Meldung wird umgehend an den SOS-Server gesendet. Hierfür werden dem SOS-Modul übergebene generische Objekte in Zeichenketten umgewandelt, die eine *SOS:InsertObservation*-Anfrage repräsentieren. Die resultierenden Zeichenketten werden direkt als Anfragen an den SOS-Server gesendet.

Im WebSocket-Modul werden die übergebenen generischen Objekte umgehend durch *Marshalling* in GeoJSON-Text-Repräsentation umgewandelt. Die JSON-Zeichenketten werden an alle Clients, die eine offene WebSocket-Verbindung mit dem Integrationssystem hergestellt haben, versendet.

3.2 WEBANWENDUNG

In der verwendeten Webanwendung, die in der Programmiersprache JavaScript geschrieben ist, werden die durch das Integrationssystem verteilten Daten empfangen und auf einer OpenLayers3-Komponente visualisiert. Die Webanwendung kann sowohl Daten von einem SOS-Server beziehen sowie WebSocket-Daten empfangen. Der Empfang der Daten über die beiden Varianten wird durch eine clientseitige Empfangsschicht abstrahiert, wodurch die Visualisierung unabhängig von der verwendeten Übertragungstechnologie erfolgen kann. Die Webanwendung beinhaltet keinen domänenspezifischen Code und kann somit für beliebige Einsatzgebiete verwendet werden. Zudem ist deren Ausführung mit allen gebräuchlichen Internetbrowsern möglich.

4 VERGLEICHSMESSUNGEN

Die Vergleichsmessungen zwischen der SOS- und der WebSocket-Variante wurden aufeinanderfolgend auf demselben Testsystem durchgeführt. Als Messwerte wurden Flugzeugpositionsdaten verwendet, die je nach Vergleichstest simuliert oder als Live-Daten von einem ADS-B-Sensor ausgelesen wurden. In Tabelle 1 werden die Attribute aufgelistet, die von einem ADS-B-Sensor empfangen und zum Integrationssystem übertragen wurden. Sowohl bei den empfangenen als auch bei den simulierten Daten wurden dieselben Attribute verwendet.

Wie in Abbildung 4 illustriert, wurden der ADS-B-Sensor für den Empfang von Flugzeugpositionsdaten, das Integrationssystem, der SOS-Server und der Client, auf dem die Webanwendung (Webclient) läuft, über einen 1 Gbit/s Layer-3-Netzwerk-Switch in einem geschlossenen lokalen Netzwerk direkt miteinander verbunden. Zudem wurden die Uhren der verschiedenen Geräte über einen separaten Network Time Protocol (NTP)-Server synchronisiert. Dadurch können die Uhren in einem lokalen Netzwerk im Submillisekundenbereich miteinander synchronisiert werden (Huston 2015).

Insgesamt wurden auf dem Testsystem vier Vergleichsmessungen durchgeführt, mit welchen jeweils unterschiedliche Aspekte der Datenübertragung verglichen wurden.

4.1 ÜBERTRAGUNGSQUALITÄT

Mit dieser Vergleichsmessung soll geprüft werden, ob die Flugzeugpositionsdaten,

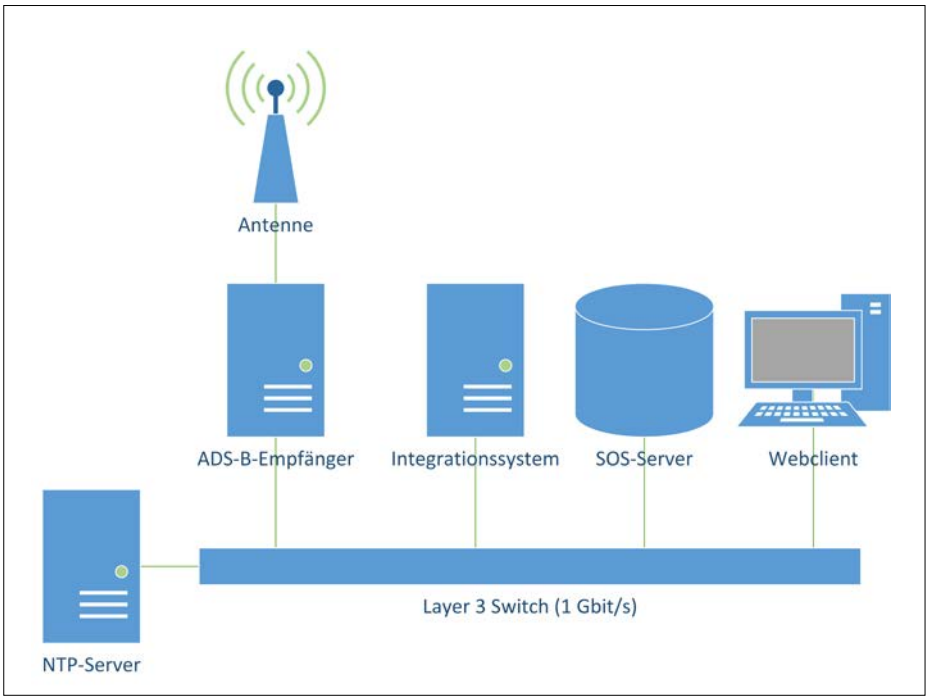

Abbildung 4: Aufbau Testsystem

die auf dem Integrationssystem empfangen werden, über beide Varianten korrekt zur Webanwendung übertragen werden. Dafür wurde der ADS-B-Sensor 90 Minuten lang an das Integrationssystem angeschlossen und die Daten wurden über beide Varianten parallel verteilt. Auf dem Webclient wurde die Webanwendung sowohl für die SOS- wie auch die WebSocket-Variante gestartet. Serverseitig als auch clientseitig wurden alle Meldungen in ein Logfile geschrieben und zur Auswertung der Inhalt der Server-Logdatei mit den beiden Client-Logdateien verglichen.

4.2 MINIMALE LATENZZEIT

In dieser Vergleichsmessung soll die minimale Übertragungszeit von ADS-B-Daten vom Integrationssystem bis zur Visualisierung in der Weboberfläche verglichen werden. Dafür wurden vom Integrationssystem simulierte ADS-B-Meldungen in einem Intervall von einer Meldung pro Sekunde übertragen. Der Vergleich wurde für beide Varianten nacheinander während 60 Minuten mit simulierten Daten einmal ausgeführt. Es wurden somit insgesamt rund 3.600 Meldungen je Variante zur Webanwendung übertragen. Zur Berechnung der Latenzzeit wurde beim Simulieren der Daten im Datenempfangsbereich des Integrationssystems der aktuelle Zeitstempel als Sendezeitpunkt der Meldung gesetzt. Clientseitig wurde der Zeitstempel nach der Umwandlung in ein generisches Format mit der aktuellen Uhrzeit verglichen. Das Delta der beiden Zeitpunkte entspricht der Latenzzeit. Bei der SOS-Variante wurde die Webanwendung so konfiguriert, dass nach jeder erhaltenen Antwort umgehend eine neue Anfrage an den SOS-Server gesendet wird.

4.3 ÜBERTRAGUNGSMENGE

Diese Vergleichsmessung soll einen Vergleich der übertragenen Datenmenge, die pro Variante über das System versendet wird, ermöglichen. Da die Bandbreite eines Serversystems in einer produktiven Umgebung nicht unbegrenzt ist, wurde zusätzlich die übertragene Datenmenge, die zwischen Webclient und Server anfällt, gesondert betrachtet. Je kleiner die Datenmenge pro Webclient ist, umso mehr Clients können sich bei gleichbleibender Bandbreite zum Server verbinden. Dieser Vergleich wurde für beide Varianten nacheinander mit simulierten Daten ausgeführt.

	Empfangene Daten	Gesendete Daten
Integrationssystem	12.867 KB	71.781 KB
SOS-Server	100.374 KB	32.875 KB
Webclient	19.149 KB	28.424 KB

Tabelle 2: Übertragungsmenge SOS

	Empfangene Daten	Gesendete Daten
Integrationssystem	1.274 KB	4.891 KB
Webclient	5.245 KB	1.322 KB

Tabelle 3: Übertragungsmenge WebSocket

Dafür wurden vom Integrationssystem eine Stunde lang in einem Intervall von 200 ms neue Meldungen erzeugt und übertragen, womit pro Variante insgesamt rund 18.000 Meldungen übertragen wurden. Die Webanwendung des SOS-Clients wurde so konfiguriert, dass diese pro Sekunde fünf Anfragen an den Server sendet. Dadurch sollte in der Regel nach jeder Anfrage an den SOS ein neuer Datensatz empfangen werden. Die übertragene Datenmenge wurde aus den Statistiken des Netzwerk-Switches ausgelesen.

4.4 MELDUNGSDURCHSATZ

Mit diesem Vergleich soll der maximale Durchsatz an Meldungen pro Sekunde bei beiden Varianten innerhalb dieses Testsystems gemessen werden. Der Vergleich wurde wiederum für beide Varianten nacheinander mit simulierten Daten ausgeführt. Es wurde während den ersten 30 Sekunden pro Sekunde eine Meldung vom Integrationssystem erzeugt. In Intervallen von jeweils 30 Sekunden wurde die Anzahl Meldungen um eine Meldung pro Sekunde erhöht. Auf dem Clientsystem wurde die Anzahl empfangener Meldungen pro Minute gemessen. Kann der Client die Menge an Meldungen nicht mehr verarbeiten, sinkt oder stagniert die empfangene Meldungsmenge pro Minute. Zudem wurde die durchschnittliche Latenzzeit betrachtet. Der Versuch wurde pro Variante während 50 Minuten durchgeführt, womit im letzten Intervall des Vergleichs 100 Meldungen pro Sekunde vom Integrationssystem erzeugt wurden.

5 ERGEBNISSE

5.1 ÜBERTRAGUNGSQUALITÄT

Insgesamt wurden in den 90 Minuten 88.923 Meldungen vom ADS-B-Empfän-

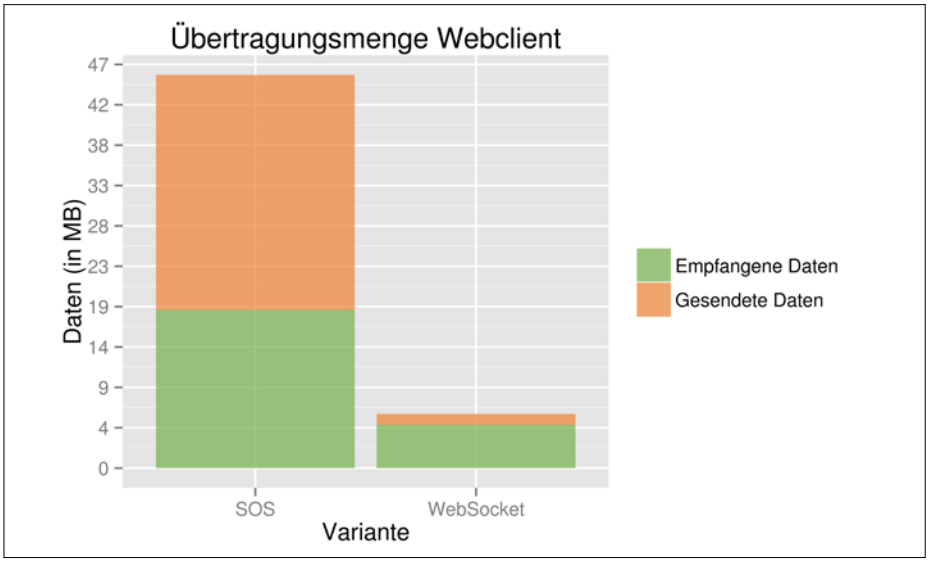

Abbildung 5: Übertragungsmenge Webclient je Variante

ger über beide Varianten übertragen. Der Vergleich der Inhalte der drei Logdateien ergab keine Abweichungen der Messwerte. Die Daten wurden somit in beiden Varianten zu 100% korrekt übertragen.

5.2 MINIMALE LATENZZEIT

Die durchschnittliche Latenzzeit bei dieser Vergleichsmessung lag bei der SOS-Variante bei 91,1 ms, wobei der kleinste Wert bei 41 ms lag. Der Maximalwert lag mit einem Faktor von etwa 1,5 zum Durchschnittswert bei 141 ms.

Bei der WebSocket-Variante war die durchschnittliche Latenzzeit bei diesem Vergleich mit 4,1 ms deutlich tiefer als bei der SOS-Variante. Die kürzeste Latenzzeit lag bei rund einer Millisekunde. Die größte Latenzzeit bei der WebSocket-Variante war mit 18 ms noch immer fünfmal kleiner als die durchschnittliche Latenzzeit der SOS-Variante.

5.3 ÜBERTRAGUNGSMENGE

Die Ergebnisse dieser Vergleichsmessungen können den Tabellen 2 und 3 entnommen werden.

Die gesamthaft versendete Datenmenge der SOS-Variante lag bei dieser Vergleichsmessung bei rund 130 MB und jene der WebSocket-Variante bei etwas mehr als 6 MB.

Betrachtet man die gesendeten und empfangenen Daten die zu resp. vom Webclient übertragen wurden, hat die SOS-Variante mit rund 47 MB in etwa 41 MB mehr Daten übertragen als die WebSocket-Variante (ca. 6 MB).

5.4 MELDUNGSDURCHSATZ

Wie dem Liniendiagramm in Abbildung 6 entnommen werden kann, skaliert die SOS-Variante bis 26 Meldungen pro Sekunde linear. Ab 26 Meldungen pro Sekunde bleibt die Anzahl verarbeiteter Meldungen in etwa konstant, die Latenzzeit wird ab diesem Zeitpunkt stetig größer.

Da die WebSocket-Variante die 100 Meldungen pro Sekunde verarbeiten konnte, ohne die Latenzzeit erheblich zu beeinflussen, wurde die Messung für die WebSocket-Variante mit vierfacher Meldungsmenge wiederholt.

Bei diesem Versuch brach die Kommunikation bei 196 Meldungen pro Sekunde abrupt ab (vgl. Abbildung 7). Bis zu diesem Zeitpunkt konnten die Meldungen ohne

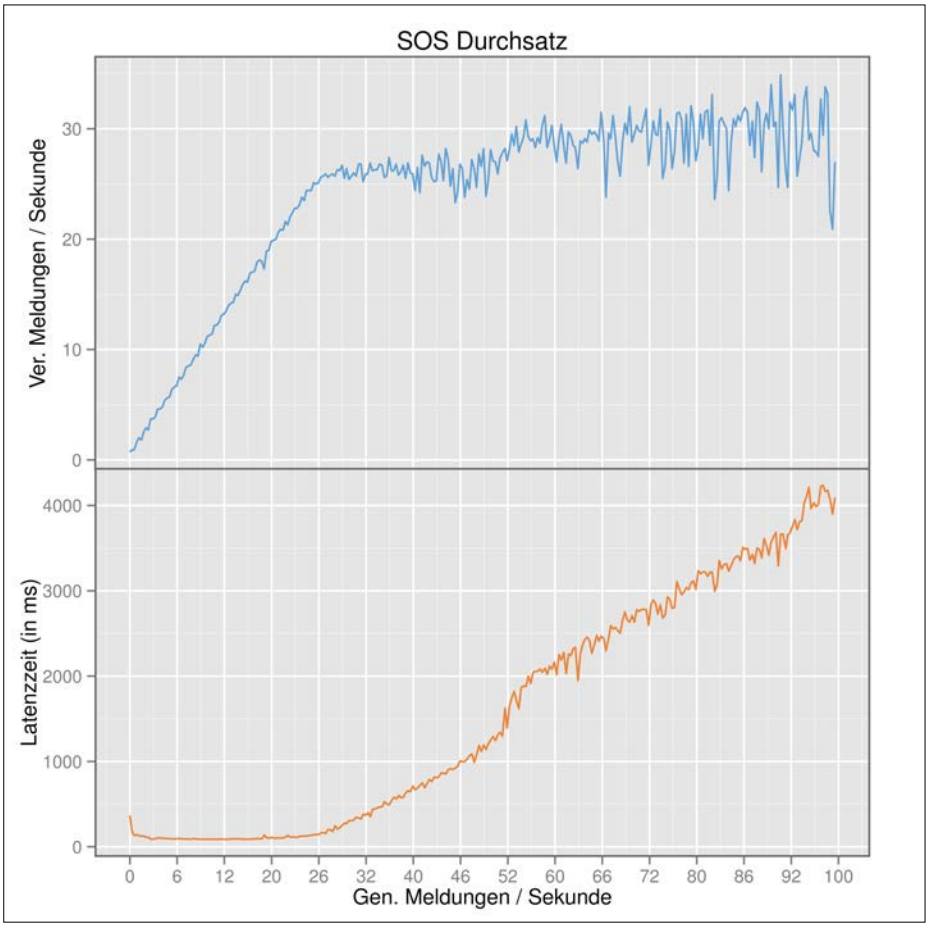

Abbildung 6: SOS-Durchsatz

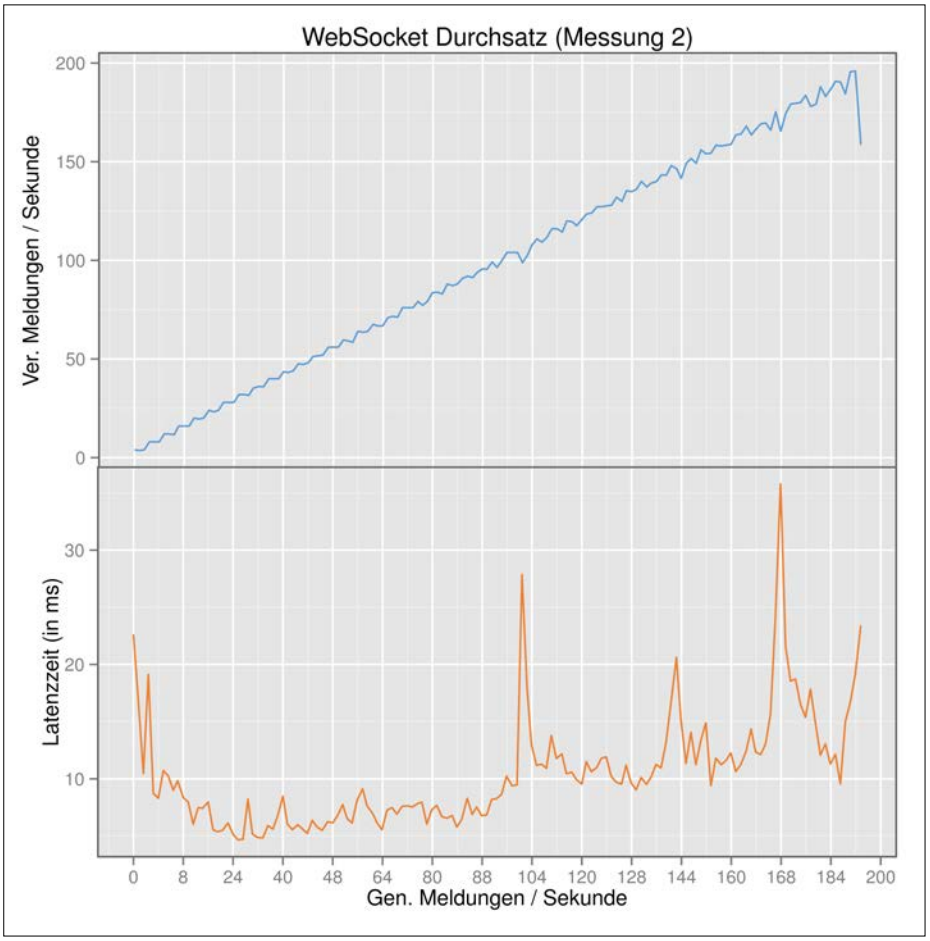

Abbildung 7: WebSocket-Durchsatz (Messung 2)

grobe Beeinflussung der Latenzzeit erfolgreich übertragen und verarbeitet werden.

6 DISKUSSION

Die Vergleichsmessung der Übertragungsqualität hat ergeben, dass in beiden Varianten in einem parallelen Betrieb während 90 Minuten alle ADS-B-Livedaten, bei einer durchschnittlichen Meldungsmenge von rund 16.5 Meldungen pro Sekunde, vom Integrationssystem korrekt auf die Webanwendung übertragen werden konnten. Somit kann festgehalten werden, dass sowohl die WebSocket- wie auch die SOS-Variante für die Übertragung von Daten, die von einem ADS-B-Empfänger stammen, zuverlässig eingesetzt werden können.

Grundsätzlich kann durch die direkte und asynchrone Kommunikation der WebSocket-Variante erwartet werden, dass die Latenzzeit dieser Variante bedeutend geringer ist als diejenige der SOS-Variante. Die Vergleichsmessung hat diese Erwartung bestätigt. Die durchschnittliche Latenzzeit der WebSocket-Variante war mit rund 4 ms erheblich kürzer als der durchschnittliche Wert der SOS-Variante (91 ms). Die kürzeste Latenzzeit, die mit der WebSocket-Variante erreicht wurde, lag mit 1 ms sehr viel näher bei der Echtzeit als der kürzeste Wert der SOS-Variante mit 41 ms, der sogar deutlich über dem Höchstwert der WebSocket-Variante liegt. Folglich ermöglicht es die WebSocket-Variante dem Anwender, durch eine minimale Latenzzeit näher zur Echtzeitdarstellung von Messwerten zu gelangen als dies mit dem SOS möglich ist.

Die Betrachtung der übertragenen Datenmenge muss differenziert betrachtet werden, da bei der SOS-Variante die Kommunikation über einen SOS-Server erfolgt. Die Messwerte werden zunächst zum SOS-Server gesendet, wo diese gespeichert werden, ehe sie von der Webanwendung abgefragt und zu dieser übertragen werden. So werden bei der SOS-Variante die Werte doppelt übertragen. Bei Betrachtung auf Stufe des Gesamtsystems hat die SOS-Variante rund 21-mal mehr Daten für die gleiche Anzahl Meldungen übertragen als die WebSocket-Variante.

Wird die vom Webclient verursachte Datenmenge isoliert betrachtet, können für die WebSocket-Variante dennoch geringere Datenmengen erwartet werden. Diese Erwartung stützt sich auf den geringeren Overhead einer WebSocket-Meldung, das performante Datenformat und insbesondere die asynchrone Kommunikation mit dem Server. In diesem Fall war denn auch die bei der WebSocket-Variante gemessene Datenmenge rund 7-mal kleiner, womit theoretisch mit gleichbleibender Bandbreite rund 7-mal mehr Clients bedient werden könnten als mit der SOS-Variante.

Beim Meldungsdurchsatz kann aufgrund der bereits im vorangehenden Abschnitt aufgeführten Eigenschaften erwartet werden, dass die WebSocket-Variante mehr Meldungen als die SOS-Variante verarbeiten kann. Die Vergleichsmessung zur Evaluation des Meldungsdurchsatzes bestätigt diese Erwartung. Die SOS-Variante konnte bis zu 26 Meldungen pro Sekunde gut skalieren wobei höherer Meldungsdurchsatz auf Kosten der Latenzzeit ging. Die WebSocket-Variante konnte bis 196 Meldungen pro Sekunde gut skalieren, womit der Meldungsdurchsatz unter Berücksichtigung einer kleinen Latenzzeit entscheidend größer war als bei der SOS-Variante.

7 FAZIT UND AUSBLICK

Mittels eines Vergleichssystems wurde die Übermittlung von Sensormesswerten über zwei Varianten zu einer Webanwendung verglichen. Wie den Ergebnissen der Vergleichsmessungen entnommen werden kann, ist sowohl die SOS- als auch die WebSocket-Variante fähig, die ADS-B-Flugzeug-Positionsdaten in Nah-Echtzeit zuverlässig zu übertragen. Die WebSocket-Variante weist jedoch bedeutende Performancevorteile auf. Der Einsatz von WebSocket eignet sich somit sehr gut für die Übertragung von Nah-Echtzeitdaten und kann für diesen Einsatzzweck empfohlen werden.

Sowohl das Integrationssystem als auch die Webanwendung wird laufend auf GitHub weiterentwickelt (GitHub 2016). Damit diese Client-/Server-Lösung noch flexibler eingesetzt werden kann, könnten folgende Erweiterungen entwickelt respektive ausgearbeitet werden.

7.1 WEBSOCKET-FILTER

Die WebSocket-Variante könnte um eine serverseitige Filterung erweitert werden. Dadurch könnte ein Client Filter-Befehle an den Server senden, sodass nur die gewünschten Daten zu ihm übertragen werden. Die Filter könnten sowohl ortsbezogen (z. B. alle Flugzeuge im Umkreis von 10 km um Basel) und/oder auf Attributebene wirken (z. B. Fluggeschwindigkeit > 573 km). Das Integrationssystem würde ab dem Zeitpunkt, ab welchem der Filter-Befehl eingegangen ist, nur Meldungen durch das WebSocket zum Client senden, welche dem Filter entsprechen. Ein Proof-of-Concept dieser Erweiterung wurde bereits erfolgreich umgesetzt. Eine spezifizierte Implementierung dieser Variante steht jedoch noch aus.

7.2 SES-ANBINDUNG

Für das Integrationssystem könnte ein neues Datenempfangsmodul entwickelt werden, das Notifikationen eines SES verarbeiten kann. Dadurch könnten SES-Notifikation umgehend zu einer Webanwendung übertragen werden. Die entwickelte Client/Server-Lösung würde mit dieser weiteren Schnittstelle die SWE-Umgebung besser für Webanwendungen zugänglich machen.

7.3 HYBRIDE LÖSUNG

Beide Übertragungsvarianten haben gegenüber der anderen Variante spezifische Vorteile. Der große Vorteil der WebSocket-Variante besteht, wie in diesem Beitrag dargelegt, in der Performanz, wohingegen ein wesentlicher Vorteil der SOS-Variante in der Möglichkeit der Historisierung der Daten liegt. Denkbar ist eine Lösung, welche die Nutzung der Vorteile beider Varianten in einem parallelen Betrieb erlaubt. Ein Client könnte für die Echtzeitwiedergabe die WebSocket-Variante verwenden. Durch Umschalten in einen Wiedergabemodus auf dem Client könnten zudem historisierte Daten geladen und dargestellt werden. Dieser Wiedergabemodus würde Anfragen an den SOS mit dem gewünschten Zeitraum stellen, um die Daten vom angegebenen Zeitpunkt zu laden und diese zu visualisieren. Mit einer solchen hybriden Lösung würden sich die beiden Varianten optimal ergänzen. Die technische Umsetzung dieser Lösung wurde bereits durchdacht, die Implementierung ist jedoch zurzeit noch ausstehend.

Literatur

Botts, M.; Percivall, G.; Reed, C.; Davidson, J. (2008): OGC® sensor web enablement: Overview and high level architecture. In: GeoSensor networks, 2008, S. 175-190. Springer, Berlin/Heidelberg.

Butler, H.; Daly, M.; Doyle, A.; Gillies, S.; Schaub, T.; Schmidt, C. (2008): The GeoJSON format specification. http://geojson.org/geojson-spec.html, Zugriff 01/2016.

Cox, S. (2011a): Abstract Specification – Observations and measurement (10-004r3). Open Geospatial Consortium Inc. Specification, 2010, Version 2.0.

Cox, S. (2011b): Observations and measurements-xml implementation (10-025r1). Open Geospatial Consortium Inc. Specification, 2011, Version 2.0.

Crockford, D. (2006): The application/json media type for JavaScript object notation (json). IETF, RFC 4627. https://tools.ietf.org/html/rfc4627, Zugriff 01/2016.

Deveria, A. (2015): Can I Use ... WebSockets. http://caniuse.com/WebSockets, Zugriff 01/2016.

Echterhoff, J.; Thomas, E. (2008): OpenGIS Sensor Event Service Interface Specification (proposed) (08-133). Open Geospatial Consortium Inc. Discussion Paper, 2008, Version 0.3.0.

Fette, I.; Melnikov, A. (2011): The WebSocket protocol. RFC 6455 (Proposed Standard), Internet Engineering Task Force, 2011. http://www.ietf.org/rfc/rfc6455.txt, Zugriff 01/2016.

Fraternali, P.; Rossi, G.; Sánchez-Figueroa, F. (2010): Rich internet applications. In: Internet Computing IEEE, 14 (4), S. 9-12.

GitHub (2016): GitHub – Where software is built. http://github.com/stues, Zugriff 01/2016.

Goodchild, M. (2011): Looking forward: Five thoughts on the future of GIS. Esri, ArcWatch 02/2011. http://www.esri.com/news/arcwatch/0211/future-of-gis.html, Zugriff 01/2016.

Huston, G. (2015): Protocol Basics: The Network Time Protocol. The Internet Protocol Journal, 15 (4), S. 2-12.

Jirka, S.; Bredel, H. (2011): Building Tracking Applications with Sensor Web Technology. In: Geoinformatik 2011 – Geochange, ifgiPrints, 41, S. 135-137.

Loreto, S.; Saint-Andre, P.; Salsano, S.; Wilkins, G. (2011): Known issues and best practices for the use of long polling and streaming in bidirectional. IETF, RFC 6202. https://tools.ietf.org/html/rfc6202, Zugriff 01/2016.

MIT (1988): The MIT License. https://opensource.org/licenses/MIT, Zugriff 11/2015.

Na, A.; Priest, M. (2007): Sensor Observation Service (06-009r6). Implementation Standard. Open Geospatial Consortium Inc. 2007, Version 1.0.

Nurseitov, N.; Paulson, M.; Reynolds, R.; Izurieta, C. (2009): Comparison of JSON and XML Data Interchange Formats: A Case Study. In: Proceedings of the International Conference on Computers and Their Applications (CAINE-2009), S. 213-218.

Pimentel, V.; Nickerson, B. G. (2012): Communicating and displaying real-time data with WebSocket. Internet Comput. IEEE, 16 (4), S. 45-53.

Scheuber, S. (2015): Nah-Echtzeit von Sensordaten im Webumfeld. Masterthesis, Universität Salzburg (UNIGIS). http://www.unigis.at/index.php/club-unigis/abschlussarbeiten/article/103116-nah-echtzeit-von-sensordaten-im webumfeld, Zugriff 01/2016.

Simonis, I. (2006): OGC Sensor Alert Service Candidate Implementation Specification (06-028r3). Open Geospatial Consortium Inc. OGC Best Practices, 2006, Version 0.9.