TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

WEDA - NEW ARCHITECTURAL STYLE FOR WORLD-WIDE-WEB ARCHITECTURE

Report of the PhD Thesis

Study programme: P2612 – Electrotechnology and informatics
Study branch: 2612V045 – Technical Cybernetics

Author: **Ing. Jitka Hübnerová**
Supervisor: doc. RNDr. Pavel Satrapa, Ph.D.

Scope of work:
Number of pages: 168
Number of figures: 66
Number of tables: 18

Liberec 2014

Abstract

In this work, the proposal of duplex event-based architectural style is described and tested. This style helps with usability, processing and performance of web services especially in conjunction with monitoring and alerting applications or geographic systems. This work has a contribution in the area of software architectures, formal modelling and performance engineering.

Thesis has theoretical and practical parts. I first give a brief review of SOA 2.0, EDA and ESB ensemble theory. Then I present a detail analysis of our Weda architecture style and its enhancements and discuss for its essence, characteristics, applicable conditions, applicable scope and application domain. With Weda, service oriented architecture can evolve to event-driven-architecture, while preserving capabilities to communicate over the World-Wide-Web. For this purpose we will introduce new Web-Event-Driven-Architecture (Weda) architectural style, protocol and developed API, which can be established easily into existing web services stack, so millions of web services can be extended and are not forced to be completely rewritten.

As tests show, impacts of the Weda architectural style is better throughput for web services and benefit of possibility to extend communication to duplex level with client contract without needing the client to publish a public endpoint over the Internet. Thanks to this discovery, we will describe a Weda's event processing capabilities such us complex event processing and we will bring them to event propagation patterned WWW ESB proposal, which can be integrated into HTTP servers and as well to SaaS model of public clouds. We will mention a practical use case scenario. Work provides quality attributes measurements conducted on our implementation.Weda is modeled as a network of timed automata which results in a compact and intuitively appealing specification and provides its formal validation and verification.

Keywords

software architecture, web services, SOA 2.0, event processing, web socket, sensor web, GIS, performance, timed automata

Abstrakt

V této disertační práci je popsán a otestován návrh duplexního událostně řízeného architektonického stylu. Tento styl rozšiřuje použitelnost webových služeb a zlepšuje jejich výkon a zpracovávání, a to zvláště ve spojení s monitorovacími a pohotovostními aplikacemi nebo geografickými systémy. Práce přispívá zejména do oblasti softwarových architektur, formálního modelování a řízení výkonu.

Disertační práce má teoretické i praktické části. Nejprve je předložen přehled teorie architektur SOA 2.0, EDA a ESB. Následně je detailně analyzována koncepce architektonického stylu Weda a jeho rozšíření, diskutována jeho podstata, charakteristiky, podmínky, rozsah a aplikační doména. S použitím architektonického stylu Weda se SOA snadno rozšíří na událostně řízenou architekturu, a to při zachování schopnosti komunikovat přes WWW. Za tímto účelem práce představuje jak specifikaci tohoto stylu, tak protokol a v neposlední řadě vyvinuté API, které může být snadno zařazeno do zásobníku vrstev stávajících služeb tak, že miliony služeb mohou být pouze rozšířeny bez nutnosti jejich přeprogramování.

Jak ukazují testy, dopadem architektonického stylu Weda je vyšší propustnost webových služeb a možnost rozšíření komunikace na obousměrnou aniž by byl klientský uzel nucen publikovat veřejný port. Dále práce popisuje, jak lze do globálně přístupného informačního systému zahrnout schopnost komplexního zpracovávání událostí (samostatný obor) a transformuje tento návrh do návrhu ESB sběrnice s vzorem propagace událostí, jejíž běh je možný přes WWW. Je tak integrovatelná do HTTP serverů a také SaaS modelu veřejných cloudů. Prakticky popíšeme a ukážeme případ užití na příkladu senzorových webů. Práce nám poskytne výsledky měření různých atributů kvality získaných vlastním měřícím programem na vlastní implementaci. Tyto výsledky jsou analyzovány s použitím teorie pravděpodobnosti a matematické statistiky. Weda styl je v práci formálně modelován, a to jako síť časových automatů, čímž práce zaručuje jeho kompaktní specifikaci, formální validaci a verifikaci.

Klíčová slova

sw architektury, webové služby, SOA 2.0, řízení událostí, web socket, senzorový web, GIS, měření výkonu, časové automaty

Contents

1. Introduction

1.1 Problem statement and motivation

Web services are increasingly gaining acceptance as a framework for facilitating application-to-application interactions within and across enterprises. SOA is widely used architecture mainly because of spread of web services. **SOA 2.0 is theoretical concept** which is not practically used today mainly because of technological and security limits. As WebsocketAPI becomes W3C and browser standard in the near future, we stay in front of the task of using new communication channel for SOA 2.0 (event-driven SOA) and try to create more decoupled World-Wide-Web based distribution system. The world of application integration over the public Internet is very conservative, and it is logical output of number of interested sides, which have to communicate with each other. From this assumption we come out to the concept of Weda. Weda tries to deal with the trend of growing amount of data transferred via WWW from the application architecture point of view and growing number of functionally coupled nodes in the network. Presented style has been strictly designed to fill a gap in layered structure of web service's protocol stack enabling a new features whilst preserving the existing. Such a concept is **backward compatible** and has bigger chance for success. It is also capable for dealing with new hosting environments (public clouds) and extensions such as possibility to extend communication to duplex level with client contract without needing the client to publish a public endpoint over the Internet. Programming-language specific API's can be built to establish a Weda into the stack.

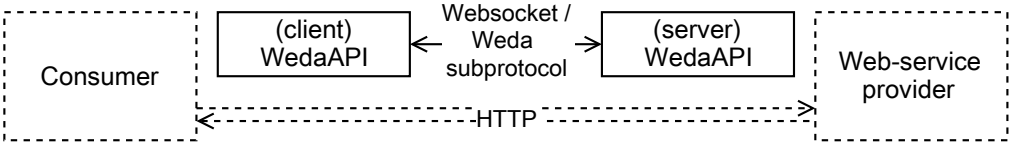

Figure 1.1: Blackbox overview of Weda

A motivation use-case for such a solution is building the part of complex GIS systems and/or public sensor web. Simply, any **alerting or monitoring solution can benefit from Weda architectural style**.

1.2 Contribution of dissertation

1.2.1 Contributions in the area of software architectures

The main objective of thesis is to provide a practical solution for building SOA 2.0 based distributed systems which are interconnected via World-Wide-Web and deployable to Cloud. Second objective is to find its advantages and disadvantages and opportunities by dealing with its performance, scalability, extensibility and internal issues. When given a name Weda, **a coordinated set of architectural constraints becomes an architectural style**. Informal description can be translated to RFC document.

1.2.2 Contributions in the area of formal modelling

Another goal is to provide a methodology on how to create a checkable model and its formal verification of Weda components and Websocket subprotocol. Websocket protocol as new protocol **isn't formally modeled** at the moment. One can use the results to model another subprotocol specification or Websocket extension. Results prove that the system of Weda components is viable and **creates a compact and intuitively appealing specification**.

1.2.3 Contributions in the area of performance engineering

Third objective is to measure referential implementation and experimental system. One can use these results to ensure that his adoption of system is efficient and scalable. While no public tool is available for such performance analysis, I developed a multithreaded **load tester SW tool** for dealing with the task. Measurement results were used to find a **methodology for predicting a RTT for any WebSocket subprotocol** and to provide a random number generator that simulates the results for other theoretical studies. Work presents the results for Weda architecture style and its subprotocol.

1.3 Organization

Original thesis document is designed to enable us to do comprehensive picture of the whole architectural style. We will talk briefly about the conceptual and also about the technical infrastructure and we will bring practical application view of the design. This report document gives a comprehensive overview and skips details and some chapters (e.g. adopting existing applications to the API, API implementation description or background research). The report is organized in the following way:

- Chapter 2 contains overview part of **informal description of the architectural style**. In thesis, the details of each component are given. In future it

can be translated to RFC draft and used for dealing with implementation issues.

- Chapter 3 presents a framework that can be used to **model and verify Weda architectural style formally**. Main automata are described here.
- Chapter 4 show experimental systems that were built upon the implementation.
- Chapter 5 contains an application of **theory of probability and mathematical statistics to study performance issues**. Input data were gained from long running test and the results are used to find a **prediction formula of system's response time**.
- Chapter 6 presents **results of selected of measurements** that compare performance quality attributes of Weda against SOAP and REST systems.
- Chapter 7 concludes the **lessons learned** and asks questions for future work.

2. Informal description of Weda architectural style

2.1 Overview

An ontology is a data model that represents a set of concepts within a domain and the relationships among those concepts. The semantics of WWW-based architecture ontology is defined as follows [2], [7]:

$$EAType \subseteq Infrastructure(t) \cup Management(t) \cup Process(t) \cup Configuration(t) \cup Component \cup Connector \cup Port \cup Role \cup QA \quad (2.1)$$

in which, *EAType* denotes enterprise architecture type, for example Weda. Thesis describes in detail main architectural ontology entities – infrastructure, management, service, service consumer and data – constrained in their relationships in order to achieve a desired set of architectural properties.

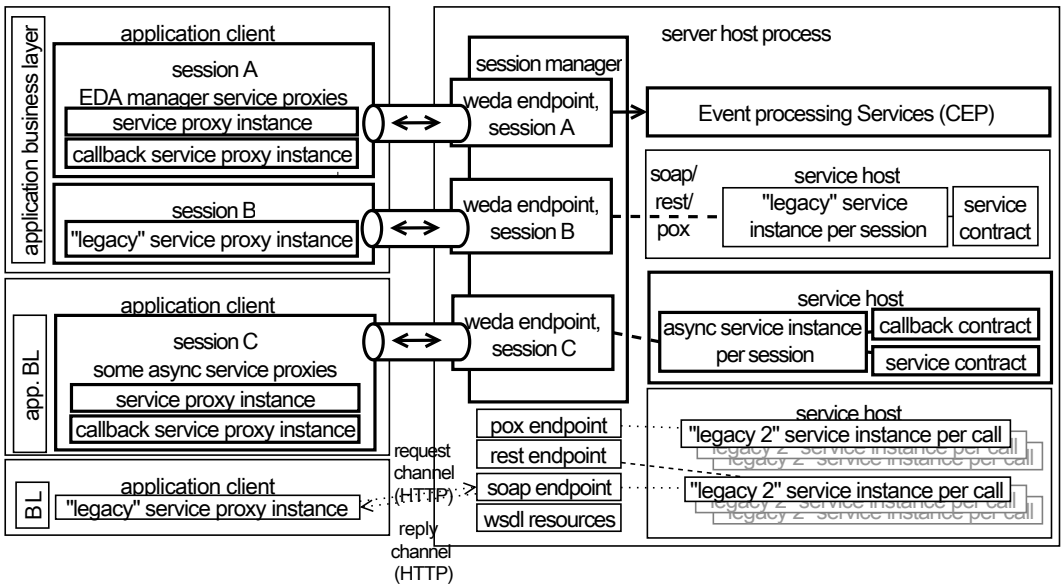

Figure 2.1: High-level process view of Weda infrastructure

2.2 Comparison of Web services architectural styles

We can identify three classes of Web services:

- REST-compliant Web services, in which the primary purpose of the service is to manipulate XML representations of Web resources using a uniform set of stateless operations
- RPC-compliant Web services, in which the service may expose an arbitrary set of operations.
- WEDA-compliant Web services, in which the service can use asynchronnous message passing which can provide us eventing behaviour as well as call & return.

In table 2.1 I compare attributes of identified styles.

Table 2.1: Comparison of Web services architectural styles

attribute	WEDA-style	REST-style	RPC-style
architecture	SOA2.0 (event-based SOA)	SOA	SOA
distributed system type	hybrid (message passing and call / return)	call / return	call / return
addressability	multiple endpoints (address, client address+identifier) per service (clients, server)	unique URI address per resource	one endpoint (address, contract) per service
common transport	HTML5 WebSockets	HTTP	HTTP
state	statefull	stateless	stateless
flow control	asynchronnous	synchronnous	synchronnous (over common transport)
process communication models	one-to-one, one-to-many, many-to-many	one-to-one	one-to-one
latency	bad (without specific flow control mechanism) or best with admission and flow control	best	good
throughput	best	bad	bad
instance context	per session	per call	per call (over common transport)
scalability	bad for horizontal, best for vertical scaling, best in terms of concurrent clients	good	good
coupling	loose (only event type definitions in duplex contracts)	functionally tightly coupled (MIME types in self-descriptive resource representations)	functionally tightly coupled (operations and data types in contract)
data interface	inherited (no restriction)	generic (e.g. HTTP verbs, MIME)	service description (e.g. WSDL)
common data format	inherited (no restriction)	HTTP resource representation, XML, JSON	SOAP
deployment topologies	enterprise service bus	hub and spoke (centralized)	hub and spoke (centralized)
coordination	ESB's native functions for orchestration and choreography, no scheduler	resource-oriented workflows (theoreticall - atom, rss, dynamic hyperlinks in practice)	service-oriented workflows, scheduler required

3. Model based analysis and formal verification

3.1 Background

Some common languages for system representation are PROMELA for SPIN or temporal logic formulas. K.I.F. Simonsen and L.M.Kristensen did a basic modeling of Websocket protocol using Petri Nets [4]. UPPAAL is a tool suite for validation and symbolic model-checking of real-time systems and uses a subset of CTL (computation tree logic i.e. class of temporal logic) as the basis for the query language.Expressive and popular formalism for modeling real-time systems is known as Timed automata [6]. SPIN can be used for untimed and UPPAAL for timed systems.

Definition 1 ***(Timed automaton)*** *is a tuple* $\mathcal{A} = (Q, q_0, \sum, E, I, C)$ *where*
Q *is finite set of discrete states (locations)*
$q_0 \in Q$ *is the initial discrete state*
$\sum$ *is set of events*
C *is finite set of clock*
$E \subseteq Q \times \mathcal{B}(C) \times \sum \times 2^C \times Q$ *is set of timed edges*
where $\mathcal{B}(C)$ *is set of boolean clock constraints involving clocks from* C *and* 2^C *is powerset of* C
$I : Q \to \mathcal{B}(C)$ *is a function associated with each discrete state* Q*. System can remain in the same location as long as the invariant is true. Invariants* I *are downwards closed,*

Table 3.1: Correspondence between CTL and UPPAAL query language syntax

CTL Formula	UPPAAL form
$A\Box\ \varphi (A\Diamond\varphi)$	A[] φ (A <> φ)
$E\Box\ \varphi (E\Diamond\varphi)$	E[] φ (E <> φ)
$\varphi \rightsquigarrow \psi$	φ –> ψ
not φ	not φ
φ *and* ψ	φ and ψ
φ *or* ψ	φ or ψ
$\varphi \Rightarrow \psi$	φ imply ψ

in the form: $x \leq n$, where n is natural number and $x \subset C$ is clock variable.

We shall write $q \xrightarrow{g,a,r} q'$ when $\langle q, g, a, r, q' \rangle \in E$.

Definition 2 ***(Guards)*** *Let C be a set of real valued clocks and I a set of integer valued variables. A guard g over C and I is a formula generated by the following syntax*

$g ::= c | g \wedge g$, where $c \in (C_c(C) \cup C_i(I))$

C_c is set of all clock constraints over C, C_i is set of all integer constraints over I. $\mathcal{B}(\mathcal{C}, \mathcal{I})$ stands for the set of all guards over C and I.

In order to study compositionality problems we introduce a parallel composition of timed automata. In order to get the kind of parallel composition we want, we have to introduce the notion of co-actions, which is done by defining a synchronization function τ.

Definition 3 ***(Synchronization Set)*** *Let Act be the set of visible actions (a,b...range over Act).*

Let $\tau \subseteq Act \times Act$ be a set of pairs such that:

$$\langle a, b \rangle \in \tau \Rightarrow \langle b, a \rangle \in \tau \text{ for all } a, b \in Act$$

Definition 4 ***(Parallel Composition)***. *Let A_1, A_2 be two timed automata. Then, the parallel composition $(A_1|A_2)$ is a timed automaton $\langle Q, q_0, \sum, E, I, C \rangle$ where $(q_1|q_2) \in Q$ whenever $q_1 \in Q_1$ and $q_2 \in Q_2$, $q_0 = (q_{1,0}|q_{2,0})$. The set of edges E is defined as follows:*

- $(q_1|q_2) \xrightarrow{g,\tau,r} (q_1'|q_2')$ *if* $(q_1 \xrightarrow{g_1,a_1,r_1} q_1') \wedge (q_2 \xrightarrow{g_2,a_2,r_2} q_2') \wedge (g = g_1 \cup g_2) \wedge (a_1, a_2 \in \tau) \wedge (r = r_1 \cup r_2)$
- $(q_1|q_2) \xrightarrow{g,\tau,r} (q_1'|q_2)$ *if* $(q_1 \xrightarrow{g,\tau,r} q_1')$
- $(q_1|q_2) \xrightarrow{g,\tau,r} (q_1|q_2')$ *if* $(q_2 \xrightarrow{g,\tau,r} q_2')$

Note that parallel composition is commutative and associative.

Model checking tools face a combinatorial blow up of the state-space, commonly known as the state explosion problem.

3.2 Modeling the Weda architectural style

Weda model consists of three parts: *Server* (with Weda gateway), *Client* and *Weda Channel*. Modeling an entity such as a *Client* or *Server* is a complex and time consuming task. We have decided to keep more abstraction levels to make model a little bit easier to understand and also to decrease the number of reachable states because of state explosion problem. For example no client proxy or Weda wire layer was modeled at the client group. On the other side, server model is a little bit more complicated to show main tasks to be performed when implementing the style. To make the understanding of the idea behind the implementation easier, the Server model has been divided into four different sub-partitions as follows:

- *WedaGateway_HostInitialization*
- *WedaGateway_SessionChannelListener*
- *WedaGateway_SessionChannelManager*
- *WedaGateway_SessionChannelWire*

Weda channel model is described by *WedaChannelInput* and *WedaChannelOutput* timed automata and *Client* is modeled via *WedaGateway_SessionChannelFactory* and *Client* timed automata. We can find automata descriptions and code listing in the original document.

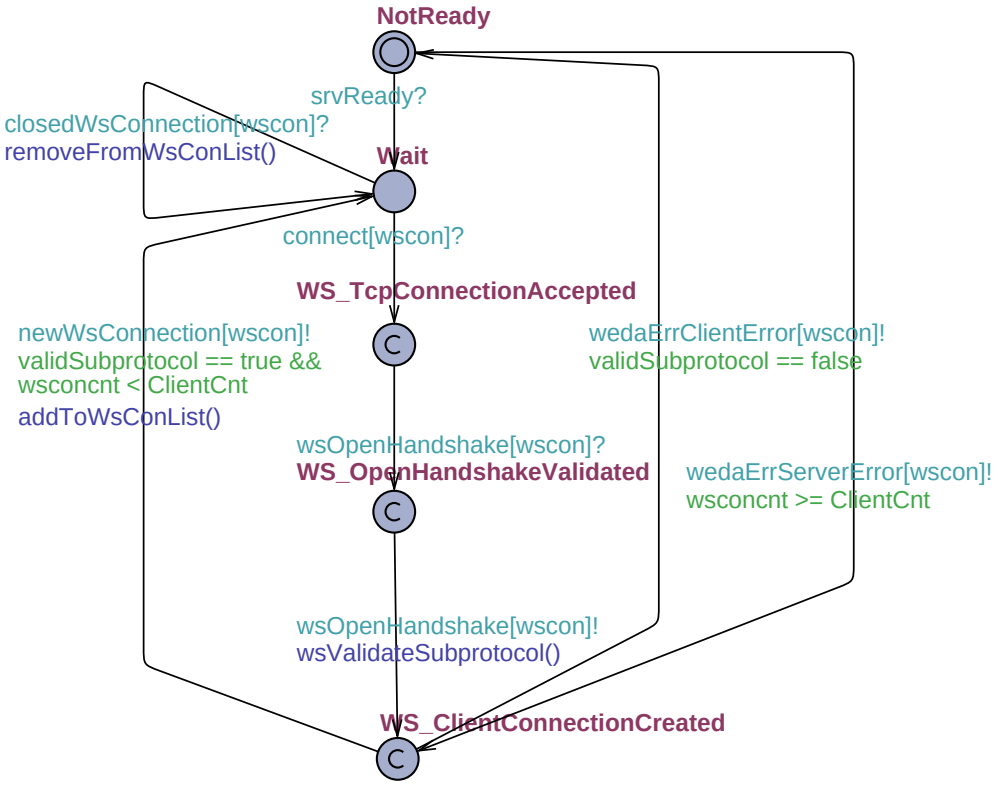

Figure 3.1: Weda Gateway Session Channel Listener model.

3.3 Verification

The finite model achieved after the modeling stage is still complex with respect to the memory cost. Thus, a restriction to a small number of processes is required in order to check correctness with UPPAAL.We restricted ourselves up to 50 client processes. Despite that, some properties are difficult to check even for one client as it requires a lot of memory space as well as time such as safety properties. For a safety properties we have created simplified model where one client is connecting to the server only. Here I list some properties

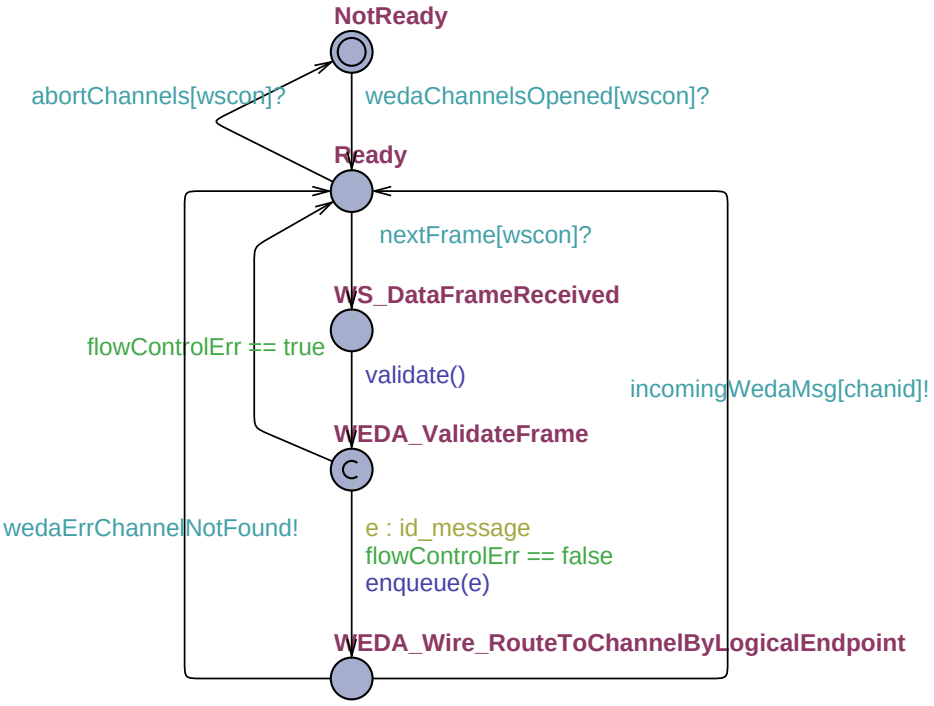

Figure 3.2: Session Channel Wire model.

has been proved to be satisfied. Next reachability property checks that client can successfully reconnect and send messages again.

$$E\Diamond(Client(0).turns > 0 \;\&\&\; Client(0).Opened \;\&\&\; WedaGateway\_SessionChannelWire(0).Ready \;\&\&\; Client(0).noRequest > 0) \quad (3.1)$$

Next liveness property verify that every request can obtain its response.

$$A\Diamond\; Client(0).noRequest > Client(0).noResponse \; imply \; Client(0).Receiving \quad (3.2)$$

$$A\Diamond\; WedaGateway\_SessionChannelFactory(0).WS\_OpeningTcpConnection \; imply \; WedaGateway\_SessionChannelListener(0).WS\_TcpConnectionAccepted \quad (3.3)$$

Previous formula verify that opening the WS connection by client leads to be accepted by the server.

$$E\Diamond\; WedaGateway\_SessionChannelWire(0).Ready \;\&\&\; Client(0).noRequest < Client(0).cntMsg \; imply \; WedaGateway\_SessionChannelWire(0).WS\_DataFrameReceived \quad (3.4)$$

Previous formula confirms that the system will try to send all messages in the cycle and they are to be received by the Weda server.

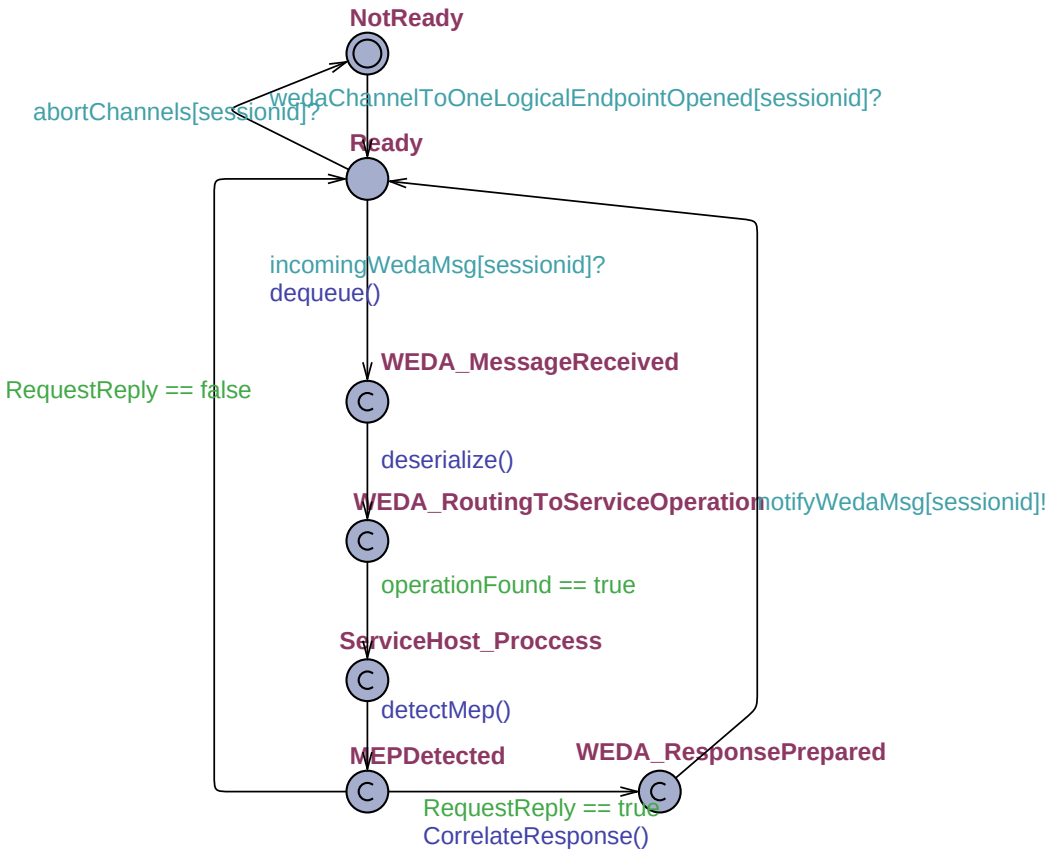

Figure 3.3: WedaChannelInput model

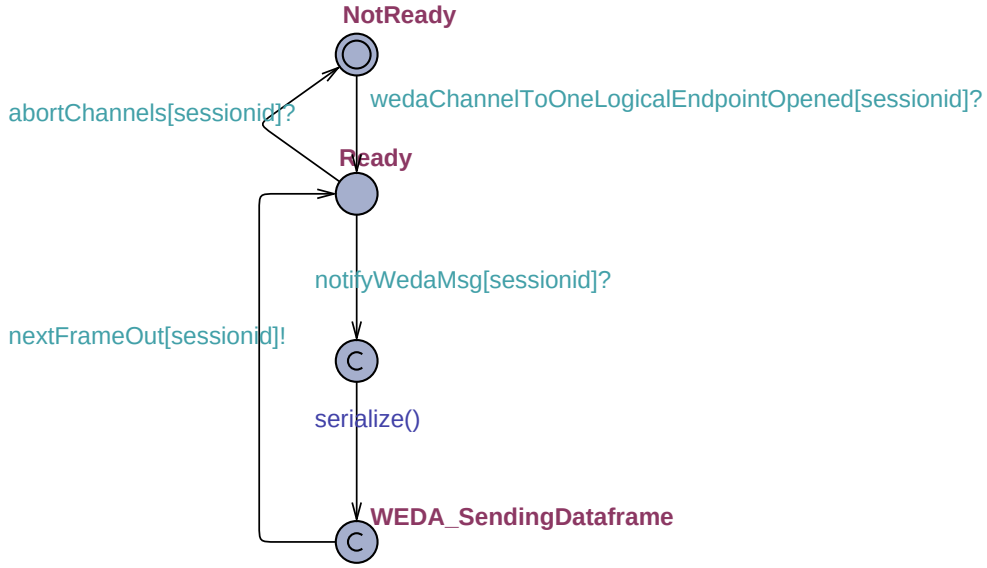

Figure 3.4: WedaChannelOutput model.

4. Experimental systems and case studies

Author implemented Weda architectural style into alpha version of Weda API, described in the thesis. With this API two experimental systems were built. Experiences and the data obtained from these experiments were used for calibrating the model. The use case for the experiments is building web based geospatial system based on standards of SWE technology that enables the implementation of Sensor Webs. Because of technology limitations, this web service stack cannot be used for real-time monitoring and alerting in particular. On figure 4.1 one can see a context model of experimental systems that were built. In thesis we describe the server side of both experimental systems, thin client side and thick client side (load testing tool used for benchmarking). Whole server's solution consists of 13 projects with number of classes.

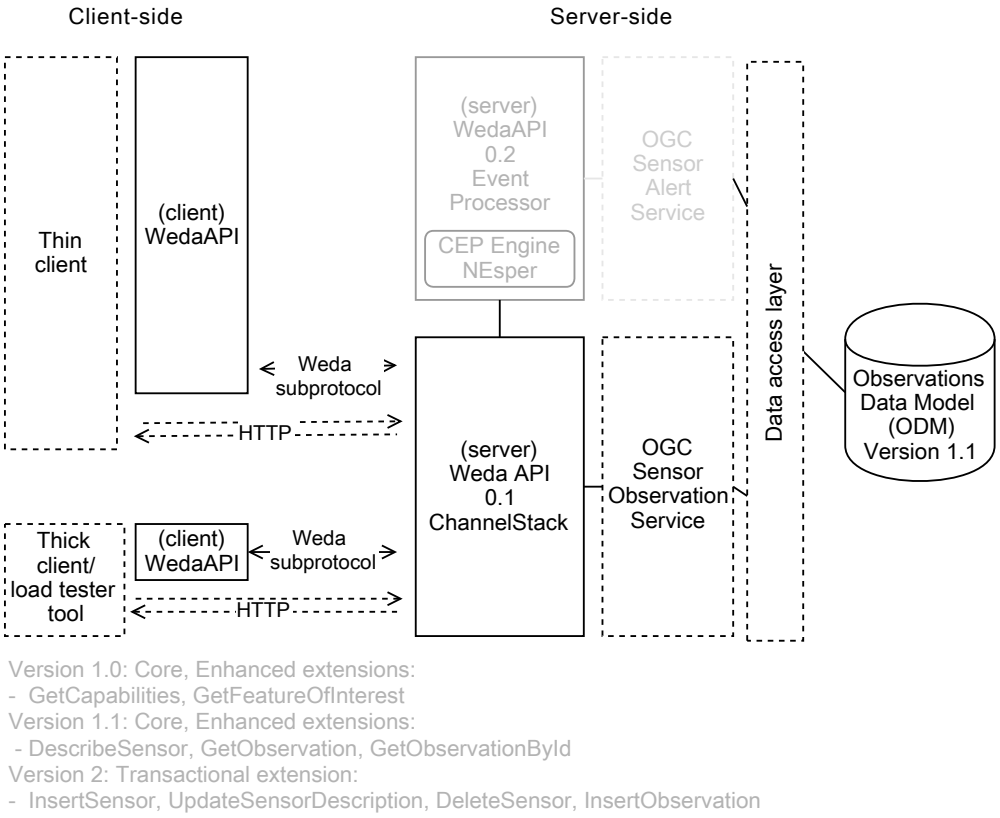

Figure 4.1: Overview of experimental systems

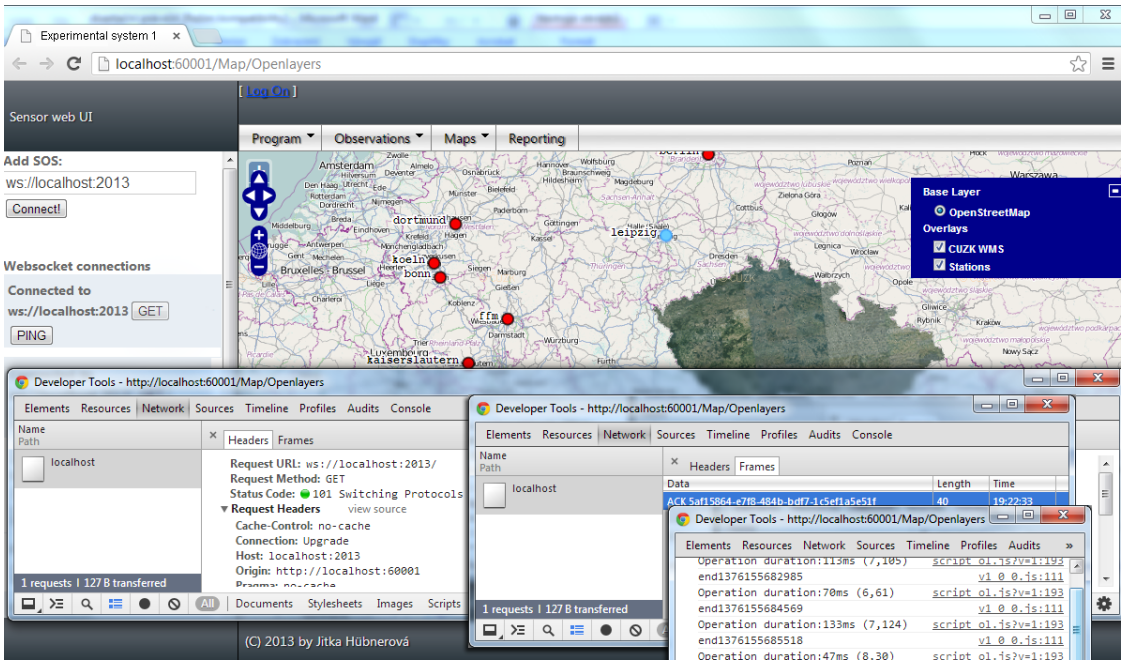

Figure 4.2: Experimental system with thin client interface

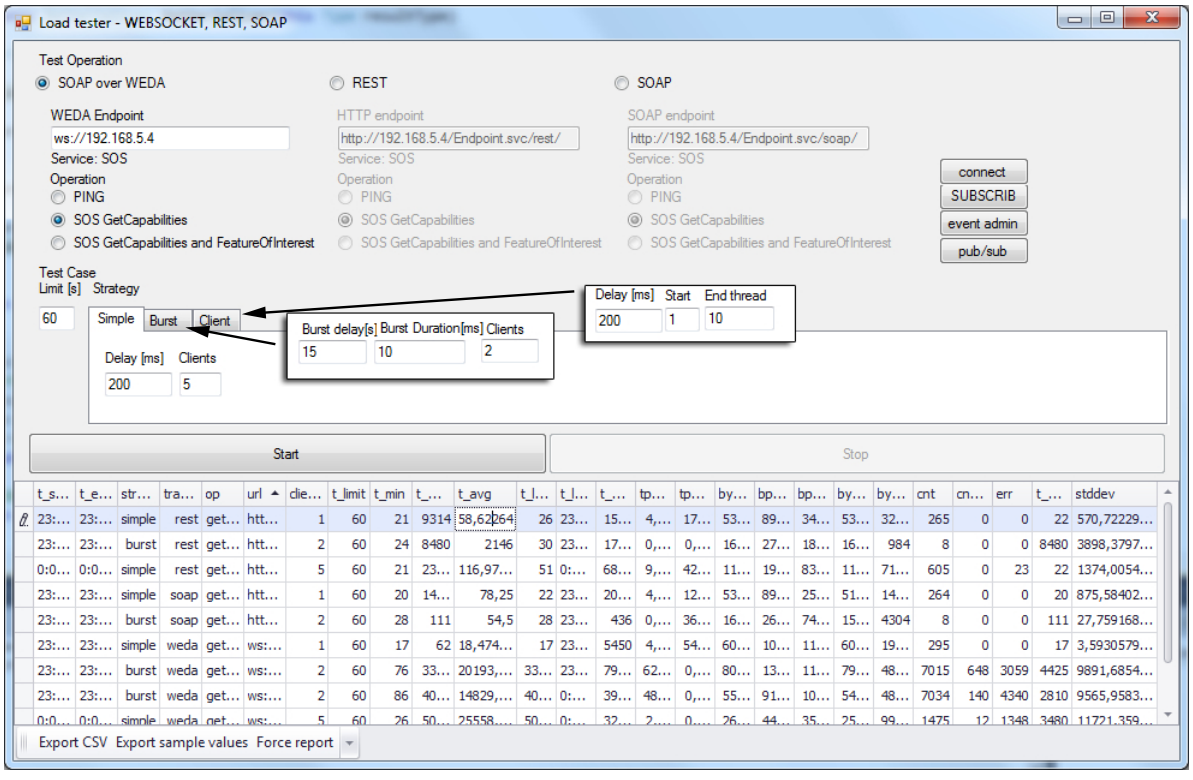

Figure 4.3: Experimental system with thick client interface

5. Distribution of response time instability

5.1 Methods and settings

The response time is the most visible performance index to users of computer systems. End-users see individual response times, not the average. Therefore the distribution of response times is important in performance evaluation and capacity planning studies. The inability of the webservices involved to guarantee a certain response time and performance and the instability of the communication medium can cause timing failures, when the response time or the timing of service delivery (i.e., the time during which information is delivered over the network to the service interface) differs from the time required to execute the system function. In result some users may get a correct service, whereas others may perceive incorrect services of different types due to timing errors. These timing errors can become a major cause of inconsistent failures usually referred as the Byzantine failures [5]. Uncertainity of performance can be refered as the unknown, unstable, unpredictable, changeable characteristics of measured system. Issues of response time analysis are becoming more important as models are developed that make explicit predictions about the shape of the response time distribution. Results for such a measuring can be used for making decision about the sytem architecture or configuration such as setting timeouts. For example today's knowledge in the area shows us that the RTs are not normally distributed [9] and they are highly skewed.

We measured the request processing time (RPT) (by a service) and the network round trip time (RTT), i.e. response time RT = RPT + RTT in store and forward devices (RFC 1242). In our work we followed methodologies described at [8, 9, 3]. Then we investigated how instability changed during 3 hours. With collected data we found the way to predict and represent performance uncertainity of Weda by employing one of the theoretical distributions, used to describe such random variables like Weda's response time.

We measured response time instability of Weda by invoking number of requests from the thick client application and collecting the responses with metadata about server processing times. The load generator was hosted on 4xIntel Xeon running at 2.5 Ghz, Windows 8, 2GB of RAM, 1Mbps downlink network connection and 100Kbps uplink network connection. The location of the load generator was 4 networks hops away from the server hosting the service. Reverse proxy (no caching) was placed between the client and server. The average

packet round trip time was 33 ms and constituted less than 1% of the service time. Server was hosted on Intel Core i7 2670qm running at 2.2 Ghz, 4GB DDR3 665MHz, Windows 7 professional sp1 64bit. Database server (MSSQL 2008 R2) was running at the same host as Weda server so its latency is included in total amount of RPT. It was found that RPT times are mainly consisted of latency of data access layer (99%).

Test case for measuring response time instability has been defined with constant payload of GetCapabilities operation invoked at OGC SOS webservice. Every 10s for a 3 hours a request was sent and results measured what gave us more than 1000 samples. Server processed each request by proper serialization at each layer up to the bottom data access layer. Backward propagation of results was packed into response frames by Weda API and metadata about server processing times were glued into the response. Illustration of setup and measuring points can be viewed at figure 5.1.

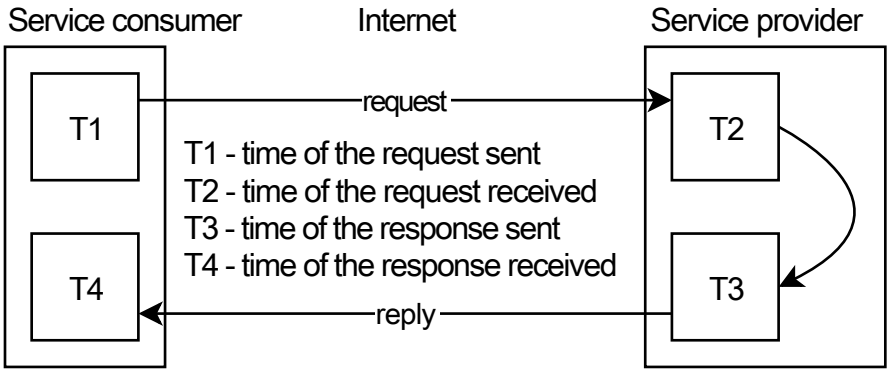

Figure 5.1: Benchmark setup

Following formulas describe what is evident from the figure, so how the request processing time (RPT) and network round trip time (RTT) was collected. RTT includes a time for request forwarding achieved by our reverse proxy. This was used to simulate such a device's delay.

$$RT = T4 - T1 \tag{5.1}$$

$$RPT = T3 - T2 \tag{5.2}$$

$$RTT = (T2 - T1) + (T4 - T3) = (T4 - T1) - (T3 - T2) = RT - RPT \tag{5.3}$$

5.2 Performance analysis

Performance trends and uncertainity results are shown at figure 5.2.

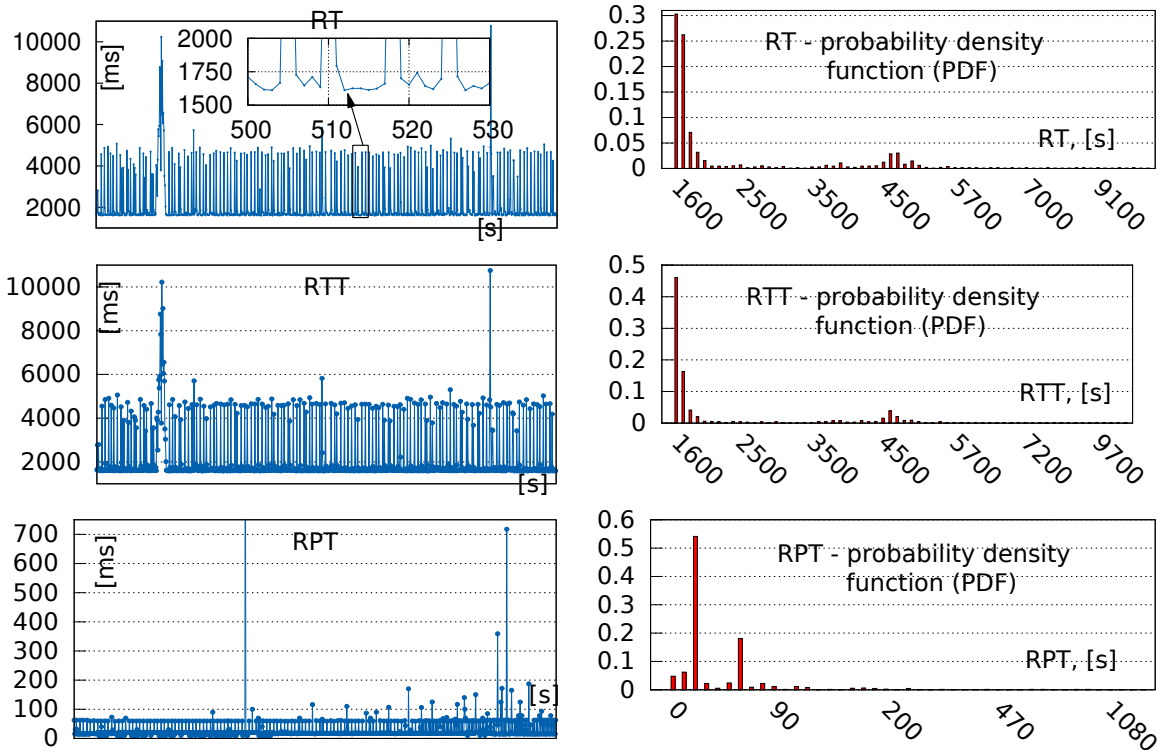

Figure 5.2: Performance trends and probability density of RT, RTT, RPT

It can be seen that response times are constant for 50% (1600-1700ms) samples. 73% of sample's RT were close to 95th percentil. 26% of samples have uncertain response time varying from 2s to 11s (four times more then average value). RTT is main part of RT value. It's distribution is very similar to RT as 48% of RTT's are between 1600 and 1700ms. One small peak can be found at 4,5s where 7% of samples are situated.

Table 5.1 shows statistics of the test. A ratio between standard deviation and average value is used as uncertainty measure. From the table we get that RT and RTT have relatively small uncertainity but RPT has significant instability however not much affecting final RT. Very small amount of samples are significantly affected by RPT giving more than 1s to total RT. From this point there is no chance to significantly improve performace by improving serialization technique (except adding the compression) or dealing much with the implementation [1].

5.3 Hypothesis checking technique and Goodnes-Of-Fit analysis

Collected experimental data can be used to obtain an approximation of the response time distribution of instability in the distribution function. We verify several hypotheses that experimental data are the most suitable one of these

Table 5.1: Performance statistics: RT, RPT, RTT

	Min [ms]	Max [ms]	Avg [ms]	95th [ms]	Std.dev.	Std.dev / Avg [%]
RPT	10	1280	31	18	54	940
RTT	1580	10754	2197	1630	1244	56.6
RT	1608	10773	2258	1669	1243	55.8
ping RTT	32	367	40	35	3	1.7

distribution functions: Normal, Exponential, Gamma, Wibull, Generalized extreme value, t Location-Scale or Log-Logistic distribution. Consequently, it is necessary to answer whether the distribution function that best approximates stochastic processes, as well simulates the overall response time, individual parameters. When parameters simplify response time RTT from two independent parts and TP (processing time), then the overall response of the distribution function consists of a composition of the independent variables RTT and TP follows:

$$g(R) = g(\dot{TP}, \dot{RTT}) = f_1(\dot{TP}) f_2(\dot{RTT}) = \int_{-\infty}^{+\infty} f_1(\dot{TP}) f_2(R - \dot{RTT}) \mathrm{d}\dot{TP} = \int_{-\infty}^{+\infty} f_1(R - \dot{RTT}) f_2(\dot{RTT}) \mathrm{d}\dot{RTT} \quad (5.4)$$

Kolmogorov-Smirnov test can test the hypothesis that known distribution Ψ equals to another distribution Θ. That means, that this test null-hypothesis

$$H_0 : \Psi = \Theta \quad (5.5)$$

against alternative hypothesis

$$H_1 : \Psi \neq \Theta \quad (5.6)$$

Kolmogorov-Smirnov statistics for the two-sample Kolgomorov-Smirnov test is as follows:

$$D_{(n,n')} = sup_x |F_{1,n}(x) - F_{2,n'}(x)| \quad (5.7)$$

where F_1 and F_2 are empiric distribution functions of first and second sample. Null-hypothesis is rejected at level α when

$$K_\alpha > \sqrt{\frac{nn'}{n+n'} D_(n, n')} \quad (5.8)$$

In our work we used Matlab numeric computing environment (http://www.mathworks.com). We also used the GNU Octave opensource (http://www.gnu.org/software/octave/) but we have found the results inapropriate and pure.

The technique of hypothesis checking consist of two basic procedures. First, values of distribution parameters are to be estimated by analyzing experimental sample. This step is crucial for finding the distribution and can be very time consuming. The question is how to estimate the unknown parameters of a distribution given the data from this distribution. And how good are these estimates and are they close to the actual 'true' parameters? For example an iterative approach allows the parameter space to be searched and the parameter values that best fit a frequency distribution to be estimated. Second step is the null hypothesis that experimental data have a particular distribution with certain parameters (Goodness-Of-Fit). A probability density function (PDF) represents the distribution of values for a random variable. The area under the curve defined the density (or weight) of the function in a specific range. The density of a function is very similar to a probability. To perform hypothesis checking itself we used the "kstest" function:

$$[h, p - value] = kstest(x, Name, Value)$$

This function starts a Kolmogorov-Smirnov test of the null hypothesis that the sample x comes from the (continuous) distribution *Name* with estimated parameters of *Value*. I.e., if F and G are the CDFs corresponding to the sample and *Name*, respectively, then the null is that F == G. The *p-value* of the test is returned. One often "rejects the null hypothesis" when the *p-value* is less than the predetermined significance level which is often 0.05 or 0.01, indicating that the observed result would be highly unlikely under the null hypothesis (i.e., the observation is highly unlikely to be the result of random chance). A significance level of 0.05 would deem extraordinary any result that is within the most extreme 5% of all possible results under the null hypothesis. In this case a *p-value* less than 0.05 would result in the rejection of the null hypothesis at the 5% (significance) level.

From figure 5.2 we have found that each of our distribution (RT, RTT, RPT) is bimodal, with two relative maxima. In this case the mean and median are not very useful since they give only a "compromise" value between the two peaks. Such a behavior makes it very hard to find some matching theoretical distribution. The option to get some relevant results here is to decompose the bimodal distribution into the unimodal components. Bimodal distribution can indicate that the mean of the process has been shifted over the period covered by the data. In our dataset the bimodality wasn't caused by shifting the mean in time or space and it is constantly distributed accross the dataset. This behavior is specific to the Websocket transport protocol.

In following work we have tried to check the number of hypothesis that experimental data conform to Normal, Exponential, Gamma, Weibull, Generalized extreme value, t Location-Scale or Log-logistic distribution. Distribution is then described by its parameters, for example by mean μ and standard deviation σ. Parameters for standard distributions were extracted from "dfittool","gamfit" and "wblfit" Matlab's functions.The alpha value used for all tests was the default 0.05.

To check the hypothesis that the vector *data* has a normal distribution

$$y = f(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

following test is to be run: $[h,p] = kstest(data, [datanormcdf(data,\mu,\sigma)])$. Another distributions are tested similarily. To check if it has Exponential distribution

$$y = f(x|\mu) = \frac{1}{\mu} e^{-\frac{1}{\mu}}$$

we used $[h,p] = kstest(data, [data\ expcdf(data,\mu)])$. To check if it has Gamma distribution

$$y = f(x|a,b) = \frac{1}{b^a \Gamma(a)} x^{a-1} e^{\frac{-x}{b}}$$

we used $[h,p] = kstest(data, [data\ gamcdf(data,a,b)])$. To check if it has Weibull distribution

$$f(x|a,b) = \frac{b}{a}(\frac{x}{a})^{b-1} e^{-(x/a)^b}$$

we used $[h,p] = kstest(data, [data\ wblcdf(data,a,b)])$. To check if it has Generalized extreme value distribution with shape parameter $k \neq 0$

$$y = f(x|k,\mu,\sigma) = (\frac{1}{\sigma} exp(-(1+k\frac{(x-\mu)}{\sigma})^{-\frac{1}{k}})(1+k\frac{x-\mu}{\sigma})^{-1-\frac{1}{k}}$$

for $1 + h\frac{x-\mu}{\sigma} > 0$: $[h,p] = kstest(data, [data\ gevcdf(data,k,\sigma,\mu)])$. To check if it has t Location-Scale distribution

$$\frac{\Gamma(\frac{v+1}{2})}{\sigma\sqrt{v\pi}\Gamma(\frac{v}{2})}[\frac{v + (\frac{x-\mu}{\sigma})^2}{v}]^{-(\frac{v+1}{2})}$$

we used $pd = makedist('tLocationScale', 'mu', \mu, 'sigma', \sigma, 'nu', nu)$
$[h,p] = kstest(data, [data\ cdf(pd,data)])$. To check if it has Loglogistic distribution we check if $ln(x)$ has logistic distribution

$$\frac{a^{\frac{x-\mu}{\sigma}}}{\sigma(1+e^{\frac{x-\mu}{\sigma}})^2}$$

by $[h,p] = kstest(data, [data\ cdf('loglogistic', data, \mu, \sigma)])$

Hypothesis checking results can be seen at the tables 5.2, 5.3, 5.4. Main finding was that none of the distributions fits well to the whole dataset, mainly because of its binomality. Other finding was that the better approximation can be achieved by providing less samples to the test(!).The deviation of experimental data significantly affects goodness of fit. Because of binomality, we divided the dataset into two groups that were analyzed separatelly. As we didn't want to get the good looking approximation for every price, we

Table 5.2: RT Goodness Of Fit approximation

	All	Group1	Group2
Normal	$\mu = 2878.21$ $\sigma = 3935.31$ $p = 1.1387e-147$	$\mu = 1782.83$ $\sigma = 361.03$ $p = 2.23 * 10 - 83$	$\mu = 7038.07$ $\sigma = 7217.27$ $p = 8.3123e-25$
Exponential	$\mu = 2878.21$ $p = 6.8135e-194$	$\mu = 2959$ $p = 1.8899e-147$	$\mu = 7038.07$ $p = 4.2567e-39$
Gamma	$a = 2.18991$ $b = 1314.31$ $p = 1.6679e-118$	$a = 35.0228$ $b = 50.9047$ $p = 2.0579e-73$	$a = 2.68135$ $b = 2624.83$ $p = 1.4613e-22$
Weibull	$a = 3106.32$ $b = 1.18815$ $p = 1.1047e-142$	$a = 1934.56$ $b = 3.94769$ $p = 7.8788e-123$	$a = 7796.64$ $b = 1.32664$ $p = 3.6056e-23$
Generalized extreme value	$k = 1.81816$ $\sigma = 91.9042$ $\mu = 1657.77$ $p = 5.1726e-10$	$k = 1.06328$ $\sigma = 40.9807$ $\mu = 1640.55$ $p = 0.0033$	$k = 0.734909$ $\sigma = 806.742$ $\mu = 4584.11$ $p = 7.5435e-06$
t Location-Scale	$\mu = 1645.32$ $\sigma = 36.3157$ $nu = 0.434106$ $p = 1.2788e-103$	$\mu = 1650.59$ $\sigma = 37.7363$ $nu = 0.929007$ $p = 1.1563e-48$	$\mu = 4647.45$ $\sigma = 161.458$ $nu = 0.517398$ $p = 1.8071e-05$
Log-Logistic	$\mu = 7.60309$ $\sigma = 0.265017$ $p = 1.2371e-97$	$\mu = 7.43857$ $\sigma = 0.0568205$ $p = 3.6172e-63$	$\mu = 8.55282$ $\sigma = 0.217237$ $p = 2.2766e-11$

decided that we don't want to achieve better p-values by separating the groups to smaller and smaller datasets since such a results don't make any practical sense. Uncertainity which exists here means that generally we can't predict a response times and describe them by analytical formula especially for the distributions tested. Only prediction we try to make for RTT vector in the next section.

From the results it is remarkable, that the Exponential distribution in our case describes experimental data worst of all. Generalized extreme value distribution gave better approximation than the other six ones for RT and RTT. For *Group 1* we cannot reject the hypothesis at 1% significance level since p-value is grater than 0.01. Because $k > 0$, the GEV distribution is the type II, or Frechet, extreme value distribution. Like the extreme value distribution, the generalized extreme value distribution is often used to model the smallest or largest value among a large set of independent, identically distributed random values representing measurements or observations. For *Group 2* we cannot reject the hypothesis that the RTT vector *data* has a t Location-Scale distribution at significance leve 5%. Because of its deviation, RPT is worse to predict than the RTT of Weda as Websocket subprotocol. Results of the Goodness Of Fit can be visually verified by looking at Figure 5.3.

Table 5.3: RTT Goodness Of Fit approximation

	All	Group 1	Group 2
Normal	$\mu = 2197.11$ $\sigma = 1244.08$ $p = 5.8449e-133$	$\mu = 1662.99$ $\sigma = 172.165$ $p = 2.3842e-71$	$\mu = 4734.21$ $\sigma = 981.372$ $p = 4.7577e-13$
Exponential	$\mu = 2197.11$ $p = 8.5869e-207$	$\mu = 1662.99$ $p = 2.1338e-244$	$\mu = 4734.21$ $p = 3.7437e-38$
Gamma	$a = 5.05036$ $b = 435.04$ $p = 1.0069e-119$	$a = 136.526$ $b = 12.1807$ $p = 2.6133e-61$	$a = 33.0144$ $b = 143.399$ $p = 2.7013e-10$
Weibull	$a = 2497.49$ $b = 1.92818$ $p = 7.3111e-109$	$a = 1751.04$ $b = 5.35065$ $p = 9.6702e-129$	$a = 5138.3$ $b = 3.83308$ $p = 1.5029e-13$
Generalized extreme value	$k = 1.25514$ $\sigma = 55.6306$ $\mu = 1624.01$ $p = 2.7715e-13$	$k = 0.642105$ $\sigma = 24.0013$ $\mu = 1611.7$ $p = 0.0110$	$k = 0.107772$ $\sigma = 507.259$ $\mu = 4373.43$ $p = 3.9086e-05$
t Location-Scale	$\mu = 1609.72$ $\sigma = 14.3186$ $nu = 0.425903$ $p = 3.2866e-62$	$\mu = 1611.94$ $\sigma = 16.8082$ $nu = 0.930244$ $p = 9.2490e-31$	$\mu = 4619.14$ $\sigma = 110.453$ $nu = 0.781641$ $p = 0.2163$
Log-Logistic	$\mu = 7.48992$ $\sigma = 0.187296$ $p = 5.9267e-93$	$\mu = 7.4003$ $\sigma = 0.0245429$ $p = 1.4337e-35$	$\mu = 8.43106$ $\sigma = 0.0679032$ $p = 9.4798e-04$

Table 5.4: RPT Goodness Of Fit approximation

	All	Group 1	Group 2
Normal	$\mu = 31.4705$ $\sigma = 54.255$ $p = 2.2663e-94$	$\mu = 17.9119$ $\sigma = 5.43757$ $p = 2.0506e-56$	$\mu = 78.7223$ $\sigma = 101.303$ $p = 5.8951e-32$
Exponential	$\mu = 31.4705$ $p = 1.0056e-71$	$\mu = 17.9119$ $p = 2.6429e-190$	$\mu = 78.7223$ $p = 1.7663e-49$
Gamma	$a = 1.85158$ $b = 16.9966$ $p = 5.6492e-105$	$a = 13.8501$ $b = 1.29327$ $p = 3.2528e-41$	$a = 3.66054$ $b = 21.5057$ $p = 9.5675e-28$
Weibull	$a = 33.2591$ $b = 1.12114$ $p = 5.7279e-75$	$a = 19.8199$ $b = 2.99727$ $p = 3.3529e-59$	$a = 87.07$ $b = 1.30218$ $p = 1.5302e-36$
Generalized extreme value	$k = 0.577439$ $\sigma = 7.6873$ $\mu = 17.3101$ $p = 3.7324e-52$	$k = -0.00838105$ $\sigma = 4.04473$ $\mu = 15.7567$ $p = 1.3543e-37$	$k = 1.0481$ $\sigma = 1.64982$ $\mu = 60.2331$ $p = 1.3088e-10$
t Location-Scale	$\mu = 17.3695$ $\sigma = 1.64654$ $nu = 0.570363$ $p = 1.6232e-47$	$\mu = 17.5933$ $\sigma = 1.80677$ $nu = 1.63181$ $p = 8.7519e-18$	-
Log-Logistic	$\mu = 3.05335$ $\sigma = 0.542686$ $p = 7.9606e-63$	$\mu = 2.86082$ $\sigma = 0.122485$ $p = 2.6621e-25$	$\mu = 4.14736$ $\sigma = 0.113534$ $p = 3.8192e-22$

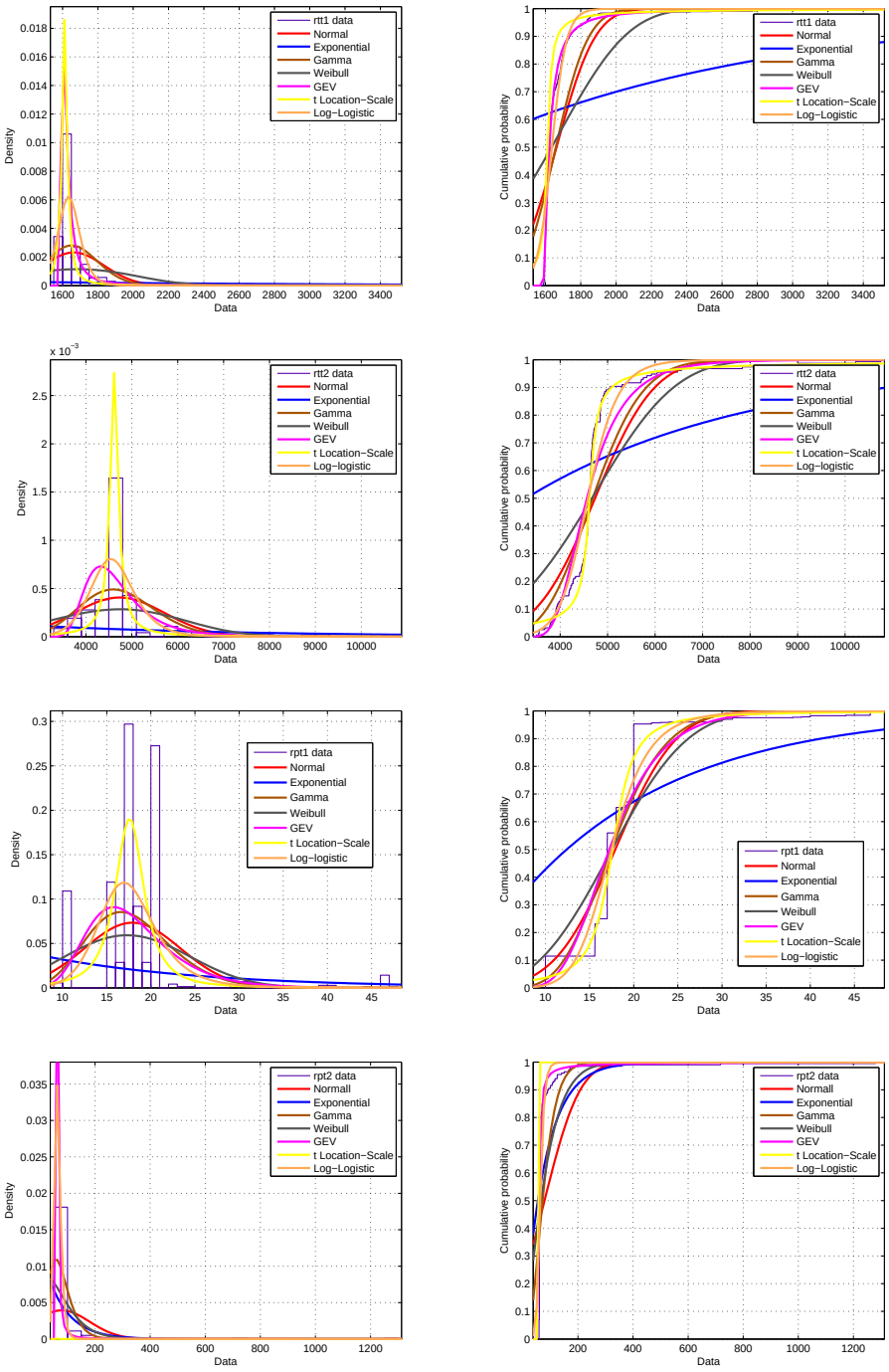

Figure 5.3: PDF and CDF fit on Group1 and 2 of RT, RTT, RPT

5.4 Performance simulation

In theoretical studies of Websocket subprotocol's performance one may want to simulate its response time. We can do this well when we know a distribution law describing the random variable.

As we didn't find any good fitting theoretical distribution for a whole dataset, in approach of simulation we must deal with the composition of two distribution laws $f_1(Group1)$ and $f_2(Group2)$ for each data vector (RT, RTT or RPT). Normally the joint probability distribution of two random variables is specified by a function of two variables, often a cumulative probability distribution function or a probability density function. Our observations are mutually exclusive. So $Group1 + Group2$ could be the answer for our composition rule. Please note that this is not true for other cases where distributions are not mutually exclusive. We can combine datasets in this way only because when $Group1$ makes an observation, there is no corresponding observation from $Group2$. There is no overlap. In this case, the intersection $Group1 \cap Group2$ is empty, leading to the conclusion: $g(Group1 \cap Group2) = 0$. This explains why, for the mutually exclusive case, $g(Group1 \; or \; Group2) = f(Group1) + f(Group2)$. Finally we can write:

$$g(data) = g(Group1, Group2) = f_1(Group1) + f_2(Group2)$$

When trying to simulate RTT distribution for example (which was the only one where the hypothesis couldn't be rejected because it's relativelly small standard deviation), the $f_1(RTT_1)$ can be a Generalized extreme value distribution with shape parameter $k \neq 0$ and the $f_2(RTT_2)$ can be a t Location-Scale distribution with individual parameters.

Our simulation approach results then in formula:

$$\begin{aligned} g(RTT) = & f(RTT_1|k,\mu_1,\sigma_1) + f(RTT_2|v,\mu_2,\sigma_2) = \\ & (\frac{1}{\sigma_1} exp(-(1+k\frac{(RTT_1-\mu_1)}{\sigma_1})^{-\frac{1}{k}})(1+k\frac{RTT_1-\mu_1}{\sigma_1})^{-1-\frac{1}{k}} + \\ & \frac{\Gamma(\frac{v+1}{2})}{\sigma_2\sqrt{v\pi}\Gamma(\frac{v}{2})}[\frac{v+(\frac{RTT_2-\mu_2}{\sigma_2})^2}{v}]^{-(\frac{v+1}{2})} \end{aligned} \tag{5.9}$$

for $1 + k\frac{RTT_1-\mu_1}{\sigma_1} > 0$

By combining data, additional "noise" is introduced into the data set. The extra noise increases the variance and therefore, also increases the uncertainty and the size of the confidence interval. In such cases, it is worse to combine the data sets than to analyze them separately. One must try both approaches and choose the one which is most appropriate for his needs.

To simulate a Websocket subprotocol response time, you can use a Matlab

function which has been found to be good approximation:

$$g(rndRTT) = horzcat(gevrnd(k, \sigma_1, \mu_1, m, n_1), \\ random('tlocationscale', \mu_2, \sigma_2, v, m, n_1)). \quad (5.10)$$

It generates m by n vector of random numbers chosen from the generalized extreme value (GEV) distribution with shape parameter k, scale parameter σ_1, and location parameter, μ_1 and t Location scale distribution with location parameter μ_2, scale parameter $\sigma_2 > 0$, and shape parameter $v > 0$.

For example you can simulate RTT distribution by running this command in Matlab:

$$rndRTT = horzcat(gevrnd(0.642105, 24.0013, 1611.7, 1, 740), \\ random('tlocationscale', 4619.14, 110.453, 0.781641, 1, 157)) \quad (5.11)$$

An accuracy of simulated RTT as compared to actual data obtained experimentally can be evaluated by use of the kstest2(x, y) function. This function performs a two-sample Kolmogorov-Smirnov test to compare the distributions of values in the two data vectors x and y. The null hypothesis for this test is that x and y have the same continuous distribution. If we try to check the hypothesis that our random variable has our analyzed composite distribution, we run this test in Matlab:

$$[h, p] = kstest2(rndRTT, RTT)$$

P-value for our random vector is 0.0745, so hypothesis that the random variable vector has **our composite distribution cannot be rejected at significance level 5%** an we can **confirm** that our Websocket subprotocol's RTT **can be simulated by developed method**.

5.5 Conclusion

We have found that the response time is mainly consisted of RTT delays while RPT plays a very small role in the RT although it has significant standard deviation.Then we prove that the probability density of RT, RTT and RPT is binomial distribution. It was shown that the composite function can be divided to disjont sets where first set has Generalized extreme value distribution and another set has t Location-Scale distribution, which models the smaller peak (for RTT).

These results provide us the opportunity to predict RTT of WebSocket subprotocol by developed formula (5.9). At the end we give an example of random variable generator of such predicted RTTs (5.10) and we test the correctness of prediction of generated data against the experimental benchmark results using a two sample Kolmogorov-Smirnov test.

6. Performance testing

6.1 Overview

To demonstrate the ability of architecture to communicate in real time between the client and server and to investigate the benefits and the bottlenecks of Weda I performed experiments to compare the Weda-style against conventional usage of web service with RPC-style (SOAP over HTTP) or REST-style. Test scenarios are outlined in table 6.1.

6.2 Measurement results

Test case 1 – constant-time bursts from 1 client

The performance test in the local area was performed on the LAN with a 0.2 milliseconds ping round trip time (RTT). Each request was consisted of call to GetCapabilities operation of SOS webservice. I first measured the message response time for middle-sized messages with 1 client and relatively high load peaks to simulate the behaviour of the system which we try to overhelm. This test case lasts for 180s and burst duration limit was set to 5ms. During the 5ms a lot of requests were invoked (since generator was not blocked by awaiting the responses in Weda-style). The parallel transport of Weda begins to show its benefit, in which Weda transport is not restricted by the bandwidth of a single HTTP request and does use the bandwidth of a duplex channel, thus boosting the transfer rate. As we can see from the results in Figure 6.1, Weda's throughput is significantly bigger (40 times) than the throughput of SOAP over HTTP or REST web service. SOAP/REST responds by one turn per second due its synchronnous processing. This great result has an other side of coin. End-user response time is also bigger as server deals with huge number of concurrent requests simultaneously. For this test case, where burst duration is 5ms, our response times are still less than timeout limit.

Test case 2 – variable publish rate from 1 client

The next measurement shows the results of test case 2 during which a server was interviewed uniformly by one client. A multiple measurements were made at minimal possible sample rates to simulate the behavior which is very close to one with connected sensor asking to SOS's InsertObservation operation at its measurement rate. As we can see in Figure 6.2, a very low publish rates can improve throughput two-times.

Table 6.1: Test scenarios

Test case	Strategy	Duration	Clients	Delay	Other parameters
1	Burst (constant client count, constant burst duration)	180s	1	15s	5ms burst duration
2	Simple (constant client count, variable publish rate)	60s	1	var	10-100 turns / s
3	Burst (constant sample count, variable client count)	120s	1-10	1s	10 samples per burst
4	Burst (burst duration, variable client count)	6/120s	1-10	1s	5ms burst duration
5	Simple (constant publish rate, variable client count)	300s	1-50	1s	

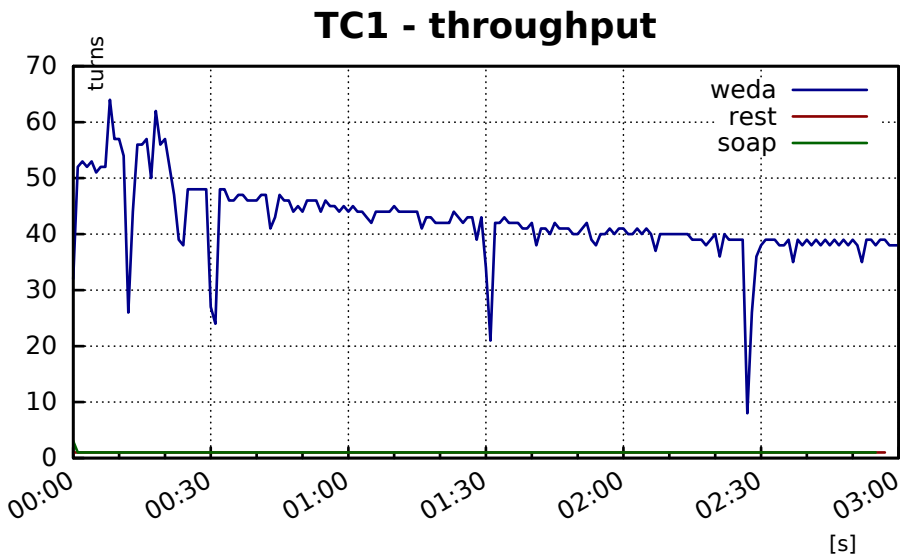

Figure 6.1: TC1 (constant burst) - throughput

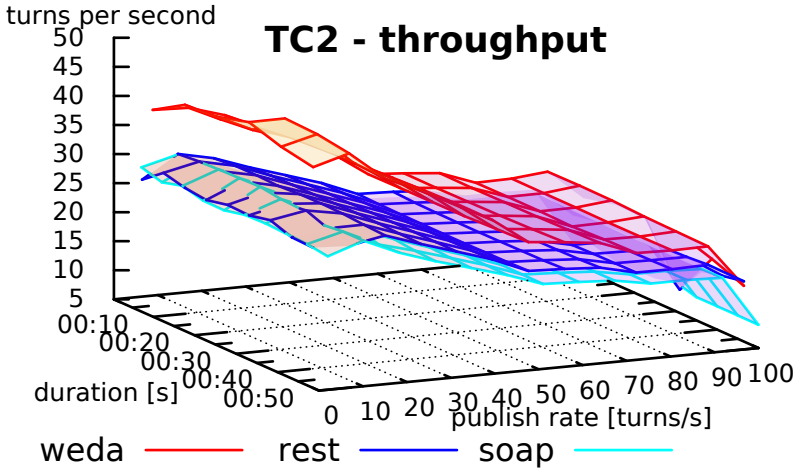

Figure 6.2: TC2 (variable rate) - throughput

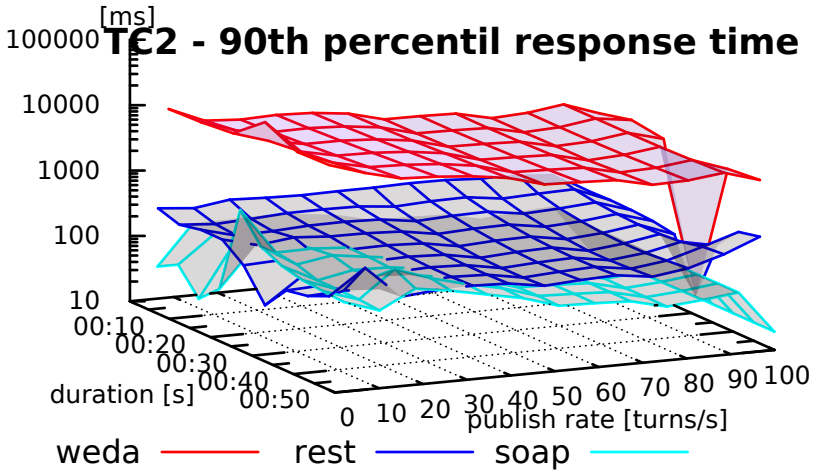

Figure 6.3: TC2 (variable rate) - 90th percentil - response time

Test case 3 – constant sample count from growing number of clients

Other test case (3) shows the information about the system's behaviour when a new client is connecting every 12s and then it sends a burst of exactly 10 samples to the server each 1s. This burst strategy is different between the others because it generates strictly a constant amount of samples per burst, no matter how much time it takes.This will make a pretty same conditions to all tested bindings as Weda cannot generate more asynchronnous requests than SOAP and REST in such a test case. As we can see from the Figure 6.4, throughput is growing exponentially with the clients for Weda. It is opposite to the REST and SOAP.

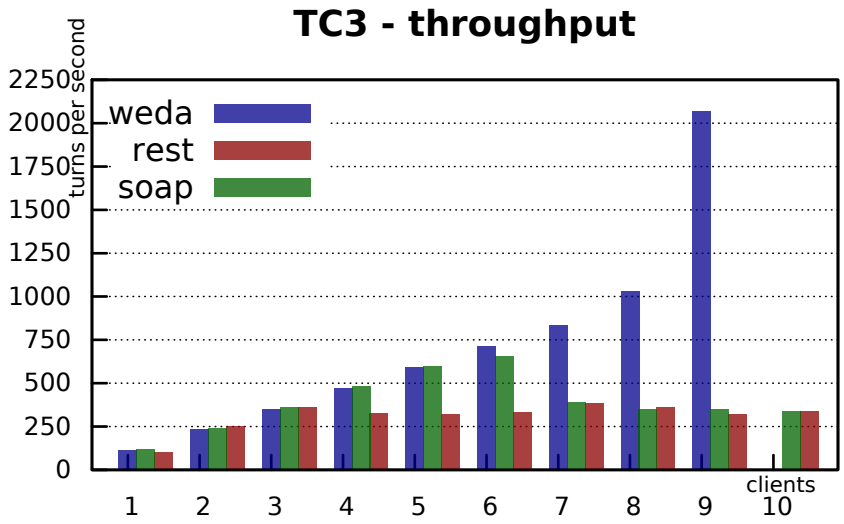

Figure 6.4: TC3 (variable clients and constant burst) - throughput

Test case 4 – variable clients and constant-time bursts

This test case gives us illustration of the worst test case for Weda-style without proper control mechanism. Test case 4 shows the situation at which a variable client count sends a burst of requests during the burst duration limit. Weda's test duration had to be shorten because in such a test case, server was overhelmed.

Test case 5 – asynchronicity differences suppressed

The last presented measurement is a test case 5 with simple strategy, constant publish rate and variable client count that is incremented up to 50 simultaneous clients each 6s, every client requesting the server every 1s. As we can see from Figure 6.5 a Weda's responsiveness is very good and constant even with the maximum number of clients in opposite of SOAP and REST. Theirs response times gets worse with each client connected. Throughput in turns per seconds is very similar for all three configurations and copies the linear curve. Throughput in KB/sec shows Weda's smaller overhead and that it has a very low impact on system's performance.

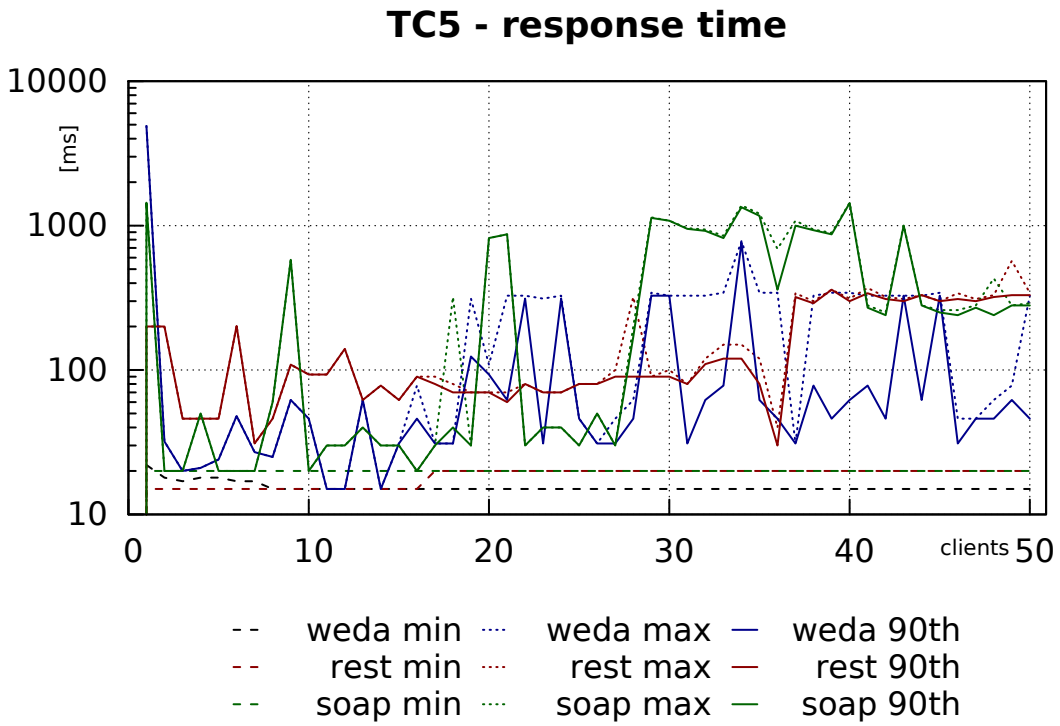

Figure 6.5: TC5 (variable clients) - response time

6.3 Conclusion

Findings on the throughput attribute

From burst-based test cases we can learn that synchronous styles (SOAP over HTTP and REST) can only achieve a small amount of turns against the Weda-style. Weda-style has 40times bigger throughput, but as such it is more susceptible to DDoS attacks without robust flow control mechanism (WebSocket supports 1,000 concurrent sessions). We have to mention that the tests ran without any flow control mechanism implemented and we see how the behaviour of the transport binding can result into impropriate response times for other clients during the high load from one attacker. So we highly recommend to implement some robust flow control mechanism in future work starting with the proposed one. A great opportunity to deal with overhelming issues is to add an admission control mechanism at each input queue. It is on discussion if such a mechanism should be required directly in WebSocket specification (not in Weda-style). Then WebSocket-standard server implementations would be forced to implement the mechanism. As we can see here, there are still questions to discuss in WebSocket specification itself.

Findings on the scalability attribute

Very interesting results we obtained from the test case 3 which suppressed the differences caused by asynchronous or synchronous transport. We saw that throughput increases exponentially with number of clients for Weda-style.

RPC and REST-styles have their peak-throughput relatively low at 6 clients count (each client invoked exactly 10 samples per burst every 1s). Weda-style proves there that is more scalable in the terms of concurrent clients. Weda's minimum response time (in TC3) shows us that there is opportunity for implementation of Weda API to perform better than other styles but big deviation caused that 90th percentil was worse than for RPC and REST-styles. Still this is from a burst-based test case which makes the differences in asynchronous/synchronnous server processing (testing server consisted of async begin/end operations processed trully asynchronously in terms of transport and server calculation as well).

Findings on the response time attribute

As we mentioned, response times were negativelly affected in test cases 1-4 because of asynchronnous processing which increased peak-throughputs for Weda-style. To suppress this behaviour we prepared test case 5 which gave us another view on Weda-style responsiveness. Conditions were set in a way leading to a very similar throughput behaviour (we prevented Weda-style to send / process more samples than other styles). With this in mind we can take a closer look on response time parameters for Weda-style in constant load. From the results we obtained that Weda's 90th percentil response time is lowest and unaffected by incrementing client count unlike the RPC and REST-style. This measurement gave us a good news for decisions about Weda-style responsiveness quality attribute. We can see, that bad results given in TC1-4 are affected by implementation of asynchronous processing and missing flow and admission control mechanism. These results have shown our main objectives for work in the future. TC5 shows that a Weda-style responsiveness is a little bit better than for RPC and REST-style.

7. Conclusions and future work

7.1 Summary

In this work author is presenting new **Web-Event-Driven-Architecture (Weda)** architectural style, protocol and developed API, which can be established easily into existing web services stack, so millions of web services can be extended, but are not forced to be completely rewritten. I have shown its strengths as being **firewall friendly web standards based solution** that can be plugged into existing applications and also I implemented Weda architectural style into the **Weda API** (0.1). The considerations about usability of WebSocket protocol for messaging purposes were presented together with additional constraints that must be made. **Informal description** of architecture style was written to be easily convertible to RFC or IANA draft. Architecture was **modeled and verified** by model checker UPPAAL, theory of timed automata and temporal logic and the work can be used to model WebSockets and its subprotocols or extensions in the future. There is no previous modelling work done before on Websockets subprotocol.

Practically the architecture was studied with the use of two **GIS-based experimental systems**. Event processing capabilities of proposal of Weda architectural style are interesting in conjunction with web and can change a user experience and coupling of applications in distributed system intensively. Presented framework allows addition to Weda to provide **complex event processing via the World-Wide-Web**. On basic concepts, a proposal of World-wide-web based **ESB topology** was built and an example of usage scenario has been given. As ESBs can be implemented inside a private cloud optimized IaaS architecture today, there is an opportunity to deploy WWW friendly ESB cloud native container over a **public cloud** even with PaaS or **SaaS architecture**. Moreover applications built as such cloud-enabled application platform don't rely on cloud management services and can be deployed on dedicated web servers too, especially if services are not necessarily to be distributed across multiple providers and load-balanced.

Lastly a performance study is presented. I studied Weda's quality attributes and especially response time instability with the help of **theory of probability and mathematical statistics**. I found a **prediction formula** of system's response time. I also offer a random number generator for other theoretical studies. To be able to collect the input data I had to develop a framework

where all middleware could be tested which leads to self-made **multithreaded load tester tool** as no such a tools exists for WebSocket subprotocols today. I then wanted to see how conventional architecture style middleware (SOAP, REST) and Weda performs against each other in terms of throughput, scalability, response time and network traffic load. I show that Weda architecture style has **great impact on the throughput** quality attribute. Server overhelming issues can have a bad impact on end-user latency. During the high load, special limits have to be set to Weda-style to eliminate susceptibility to DDoS attacks. I also deal with the question if such an admission control mechanism should be an integral part of WebSocket specification. To deal with such an issue for collecting a representative benchmark data I present the last test case at which response time attribute was studied with approach to suppress limits mentioned. This finding leads to conclusion that Weda-style **scale much better** than other styles and it is **a little bit faster** in terms of response time.

I can recommend Weda-style for

- "shared" systems crossing organization boundaries
- read & write data (not read-mostly)
- non-idempotent events or operations
- need of server invoking clients independetly (pushing)
- realizing SOA 2.0 behaviour
- after dealing with flow and admission control as alternative to RPC or REST architectural styles.

I cannot recommend Weda-style for

- load-balanced applications and read-mostly systems
- use cases with small amount of turns / messages per client

7.2 Future work

One of the main aspects of the future work should concentrate on improving the proposed architecture style by addressing the following issues:

- Stabilization of Weda API
- Adding more features to Weda-style (session reestablishment, better flow control, configuration parameters, admission control ...)
- Adding more features to Weda API (ESB features, event processor layer improvements ...)
- Transforming informal specification to the RFC or IANA draft.

- Extension of formal model to show duplex service behavior
- Adding gzip or MTOM compression to the transport and measure improvement of performance.
- Measure performance of use-case "WMS over Weda"
- Adding new implementations in more programming languages

7.3 Publications

Articles

1. J. Hübnerová, *Towards Solution for the Public Web-based GIS Monitoring and Alerting System*, Proceedings of International Conference GIS Ostrava 2014 - Geoinformatics for Intelligent Transportation, January 2014, Czech republic, ISBN 978-80-248-3311-8

    - Presented at the International Conference GIS 2014 - Geoinformatics for Intelligent Transportation. (oral presentation)

2. J. Hübnerová, *Model based analysis and formal verification of WEDA architectural style*, Proceedings of IEEE International Conference on Informatics & Applications (ICIA), September 2013, Poland, ISBN 978-1-4673-5255-0

    - Presented at the 2nd IEEE International Conference on Informatics & Applications (ICIA 2013), Lodz, Poland. (oral presentation)

3. J. Hübnerová, *Weda - new architectural style for world-wide-web architecture*, Proceedings of ISAT 2013, 34th International Conference Information Systems Architecture and Technology, September 2013, Poland, ISBN 978-83-7493-804-4

    - Presented at the 34th International Conference Information Systems Architecture and Technology (ISAT 2013), Szklarska Poreba, Poland. (oral presentation)

4. J. Hübnerová (Dařbujanová), *High performance mobile device RIA using binary MOM and SOA over WebSockets*, Proceedings of CSE'2010, International Scientific Conference on Computer Science and Engineering, Slovakia, ISBN 978-80-8086-164-3

    - Presented at the 8th International Scientific Conference on Computer Science and Engineering (CSE 2010), Kosice, Slovakia. (oral presentation)

5. J. Hübnerová, *Towards event propagation ESB for realizing SOA 2.0 in public cloud*, International journal on Cloud Computing: Services and Architecture (IJCCSA), 2013, submitted

- Submitted journal article
- Under review

Workshops

1. J. Hübnerová, *Optimalizace přenosu a zpracování dat v SOA architektuře a RIA aplikacích*, doctorand seminar, Liberec 2009
2. J. Hübnerová, *Aplikace Action Message Format 3 na komplexní datové struktury, měření a redukce latence přenosových protokolů v SOA architektuře*, doctorand seminar, Liberec 2010
3. J. Hübnerová, *High performance RIA using binary MOM and SOA over WebSockets*, doctorand seminar, Liberec 2010
4. J. Hübnerová, *Sensor Web Enablement and migration of SOS transport binding from SOAP/REST to WebSocket protocol*, doctorand seminar, Liberec 2011
5. J. Hübnerová, *Towards universality and quality attributes in World-Wide-Web architectures used for inter-process communication*, doctorand seminar, Liberec 2012
6. J. Hübnerová, *WEDA - new architectural style for World-Wide-Web Architecture*, doctorand seminar, Liberec 2013

Bibliography

[1] D. Davis and M. Parashar. Latency performance of soap implementations, 2002.

[2] D. Garlan, R. Allen, and J. Ockerbloom. Exploiting style in architectural design environments. *SIGSOFT Softw. Eng. Notes*, 19(5):175–188, Dec. 1994.

[3] A. Gorbenko, V. Kharchenko, S. Mamutov, O. Tarasyuk, Y. Chen, and A. Romanovsky. Real distribution of response time instability in service-oriented architecture. In *Proceedings of the 2010 29th IEEE Symposium on Reliable Distributed Systems*, SRDS '10, pages 92–99, Washington, DC, USA, 2010. IEEE Computer Society.

[4] K.I.F.Simonsen and L. Kristensen. Towards a cpn-based modelling approach forreconciling verification and implementation of protocol models. In *27th IEEE/ACM International Conference on Automated Software Engineering*, 2012.

[5] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.

[6] K. Michal. *Fault diagnostic based on temporal analysis*. Doctoral dissertation, UNIVERSITÉ JOSEPH FOURIER-GRENOBLE and BRNO UNIVERSITY OF TECHNOLOGY, 2006.

[7] C. Pahl, S. Giesecke, and W. Hasselbring. Ontology-based modelling of architectural styles. *Inf. Softw. Technol.*, 51(12):1739–1749, Dec. 2009.

[8] C. Walck. *Hand-book on STATISTICAL DISTRIBUTIONS for experimentalists*. Particle Physics Group, Fysikum, University of Stockholm, Dec. 1996.

[9] T. Zandt. How to fit a response time distribution. *Psychonomic Bulletin & Review*, 7(3):424–465, 2000.

This short list contains only publications relevant to the content of this report. A complete list of references is part of the dissertation document.