Response-Time Measurements using the Sun Java Adventure Builder

Philipp Reinecke
preineck@informatik.hu-berlin.de

Sebastian Wittkowski
wittkows@informatik.hu-berlin.de

Katinka Wolter
wolter@informatik.hu-berlin.de

Humboldt-Universität zu Berlin
Unter den Linden 6
10099 Berlin, Germany

ABSTRACT

Extensive data sets are an indispensable prerequisite for modelling the behaviour of service-oriented systems. As part of an ongoing research effort we are developing a SOA testbed that allows for fault-injection at various levels of the system. Within our current testbed, we utilise the Java Adventure Builder reference application as an example service-oriented system. We inject IP packet loss and measure response times for three load levels. In this paper, we report on preliminary results and present data from the experiments.

Categories and Subject Descriptors

C.4 [**Performance of systems**]: Fault tolerance, Performance attributes, Modeling techniques

General Terms

Experimentation, measurement, performance

Keywords

Service-Oriented Architectures, Phase-type distributions, Models, Fault-Injection

1. INTRODUCTION

Quality of service (QoS) of service-oriented systems becomes increasingly important, and, consequently, methods to assure QoS levels enjoy increasing attention. The development and evaluation of such methods requires general models for the performance of service-oriented systems under realistic use conditions. In particular, our focus lies on models that reflect user-perceived system behaviour when the system is confronted with common faults and disturbances, such as IP packet loss and high load situations. We will employ these models in abstract formalisms (e.g. queueing systems) to develop methods that assure QoS levels for a wide range of service-oriented systems. In order to correctly reflect system behaviour, and to foster a deeper understanding of the characteristics of service-oriented systems, such models must be parameterised based on real data, which can be obtained either in experiments on real systems or in testbeds with fault injection.

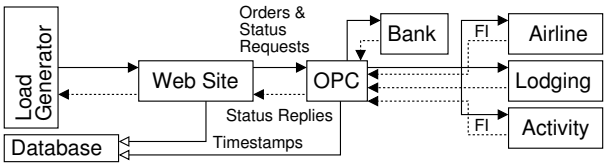

Figure 1: Structure of the testbed.

Unfortunately, neither data nor ready-made models for the performance of service-oriented systems are commonly available. In fact, apart from [16], we are aware of only a handful of other studies that consider performance in common working conditions at all: [9] and [4] present data on publicly available Web Services, while [3, 8, 10, 23] measure response times in testbeds, but do not inject faults. In [5], fault-injection is employed, however, the paper does not focus on performance implications. Apart from [10] and [16], none of the listed papers provides models for the response times.

In order to address this lack of models and data, we recently proposed a multi-level fault-injection toolkit (ML-FIT) for service-oriented systems [17]. The purpose of the testbed is to study the behaviour of service-oriented systems under realistic operating conditions, as perceived by the user. We focus on the response time of individual requests, since we consider response time the primary metric of interest from a user's point of view. Based on the measurements we will build and parameterise the required abstract models. In order to capture realistic behaviour, ML-FIT is intended to inject faults at various levels in the system (e.g. in the network, in the operating system, or in the application server). As faults are handled by the various fault-handling mechanisms present in a system (e.g. retransmissions in the network), they are transformed to other faults (e.g. loss is transformed to timing faults) and finally manifest as realistic response times.

In this paper we present some preliminary results from our first implementation of the testbed. The paper is organised as follows: The next section describes the experiment

QUASOSS'09, August 25, 2009, Amsterdam, The Netherlands.

Run	Mean	SD	c^2	Min	50%	95%	99%	Max
1	55.74	39.76	0.51	29	48	100	168	810
2	55.26	31.40	0.32	29	48	107	155	526
3	52.85	36.31	0.47	28	45	93	149	790
4	54.75	25.81	0.23	28	47	100	163	250
5	53.94	47.86	0.79	28	46	93	143	944
6	53.0	36.28	0.47	29	46	94	136	824
7	53.31	33.13	0.39	29	46	95	134	740
8	53.6	28.19	0.28	28	47	99	137	424
9	52.16	20.17	0.15	29	47	92	125	209
10	52.26	25.01	0.23	29	46	92	139	399

Table 1: Airline response times in ms, no packet loss, 10 Users

setup. Section 3 presents data from our measurements, and in Section 4 we derive phase-type distributions that model the raw data and can be used in further studies. Section 5 concludes the paper and gives an outlook on further work.

2. EXPERIMENT SETUP

The overall structure of our testbed is illustrated in Figure 1. A request is depicted as solid arrow, while the reply is shown as a dashed arrow. Solid arrows with hollow heads indicate the collection of time-stamps.

We utilise Sun's Java Adventure Builder Reference application [19] as an example of a service-oriented system. The Java Adventure Builder provides a web site interface to a SOA consisting of four service providers (called Bank, Airline, Lodging, and Activity). On the web site, users can customise adventures, each consisting of a flight, lodging, and an activity. When the user books an adventure, the web site submits an order to the Order Processing Centre (OPC). The OPC calls each of the providers and allows to retrieve the order status. The OPC implements the following workflow: First, the Bank is called by a synchronous web service invocation. Afterwards, the OPC calls the Airline, Lodging, and Activity suppliers in an asynchronous manner. Calls to these three suppliers are submitted all at the same time. The OPC then waits for replies from each of the suppliers before setting the order status to *Completed*.

We applied a few modifications to the original Adventure Builder application, in order to render the system suitable for use in the testbed. First, we introduced an immediate status notification on the web site. In our testbed, when the user submits a booking, the web site stalls (for this user) till the OPC has finished work on the order. To this end, the web site polls the OPC for the order status. The original Adventure Builder returns immediately, offering the user an option to manually request the status, and also sends an email notification upon order completion. For the experiments presented here, we chose web-only notification because it allows us to measure response times immediately noticeable to the user. We defer the study of response times with e-mail notifications to future work.

Second, in order to emulate more realistic delays, we added a simple database access operation to the Airline, Lodging, and Activity suppliers.

Third, we introduced code to store request/reply timestamps at the begin and end of each booking into a database. Timestamps for complete bookings are stored by the web site, while the OPC stores timestamps for the calls to the four service providers. We minimise the impact of the measurement process on the results by deferring the database

Run	Mean	SD	c^2	Min	50%	95%	99%	Max
1	385.56	122.69	0.10	155	372	579	764	1236
2	389.02	133.74	0.12	148	369	616	923	1306
3	375.08	122.39	0.11	146	362	550	750	1557
4	380.13	118.23	0.10	141	366	585	740	1231
5	388.68	129.42	0.11	145	374	604	772	1646
6	372.20	107.53	0.08	155	366	534	664	1235
7	392.5	132.52	0.11	143	377	606	833	1462
8	397.93	142.34	0.13	133	379	640	1017	1187
9	376.35	109.54	0.09	142	366	552	709	1243
10	373.18	110.39	0.09	141	366	551	658	1297

Table 2: OPC response times in ms, no packet loss, 10 Users

Run	Mean	SD	c^2	Min	50%	95%	99%	Max
1	942.0	1429.98	2.31	25	366	3472	6543	14202
2	693.50	1216.40	3.08	24	165	3595	5566	12708
3	667.57	1453.63	4.74	25	176	2561	8875	15112
4	624.93	947.05	2.30	24	195	2823	4265	6744
5	596.73	1097.52	3.38	25	151	2888	5651	9792
6	732.25	1025.63	1.96	24	301	2873	4456	8356
7	701.87	1289.99	3.38	25	187	3326	5399	13853
8	689.09	1217.96	3.12	25	205	3132	5309	13547
9	409.71	905.51	4.85	25	106	1753	3182	21710
10	523.49	829.44	2.51	25	139	2267	3504	9051

Table 3: Airline response times in ms, no packet loss, 50 Users

Run	Mean	SD	c^2	Min	50%	95%	99%	Max
1	5801.01	5196.8	0.80	175	4183	16675	23026	42623
2	6675.13	5128.31	0.59	191	5720	15108	22974	50264
3	5427.53	4239.88	0.61	124	4288	12937	20370	37633
4	5565.08	4253.46	0.58	220	4422	13106	19153	37699
5	5275.31	4833.28	0.84	168	4060	14841	22631	46791
6	5346.84	4210.91	0.62	196	4356	13068	19978	33712
7	6040.43	4287.29	0.50	163	5342	13605	19544	29440
8	5372.27	4164.95	0.60	158	4670	12122	20862	33300
9	5883.04	5565.85	0.90	238	4184	16265	31152	40935
10	6424.93	4717.26	0.54	182	5411	15515	21793	47307

Table 4: OPC response times in ms, no packet loss, 50 Users

Run	Mean	SD	c^2	Min	50%	95%	99%	Max
1	122.98	393.64	10.25	29	50	306	840	6181
2	132.93	453.42	11.63	28	49	311	3037	7606
3	115.78	468.42	16.37	29	48	293	633	8516
4	143.77	516.70	12.92	28	51	294	3034	7308
5	164.76	630.14	14.63	29	51	314	3266	7333
6	84.81	154.15	3.30	28	48	281	514	3562
7	95.99	204.44	4.54	29	50	286	659	3071
8	125.34	670.55	28.62	29	48	276	689	12056
9	98.64	197.55	4.01	28	51	286	391	3522
10	119.11	319.24	7.18	28	51	317	959	4476

Table 5: Airline response times in ms, packet loss on Airline-OPC link, 10 Users

Run	Mean	SD	c^2	Min	50%	95%	99%	Max
1	444.75	399.85	0.81	129	392	704	1379	6364
2	433.17	450.64	1.08	138	384	583	3274	7898
3	434.26	468.08	1.16	132	383	658	1134	8690
4	450.54	512.93	1.30	125	380	689	3301	7439
5	476.60	616.05	1.67	140	389	709	3660	7412
6	401.13	177.24	0.20	122	381	620	871	3826
7	426.29	235.93	0.31	137	392	666	1141	3365
8	439.93	668.95	2.31	144	380	595	1237	12279
9	418.59	224.33	0.29	144	388	662	946	3763
10	454.59	353.54	0.61	122	395	793	1864	4703

Table 6: OPC response times in ms, packet loss on Airline-OPC link, 10 Users

Run	Mean	SD	c^2	Min	50%	95%	99%	Max
1	854.34	1361.39	2.54	26	316	3735	5870	16875
2	570.41	1119.58	3.85	26	163	2679	5138	15320
3	814.38	1501.75	3.40	24	300	3370	6036	18624
4	953.52	1810.10	3.60	26	368	3583	6731	23587
5	740.07	1716.92	5.38	24	223	3297	6491	35597
6	891.28	1597.27	3.21	24	291	3858	6831	19690
7	768.37	1159.08	2.28	25	312	3213	4856	12829
8	714.36	1416.05	3.93	25	258	2952	5208	29616
9	455.86	1261.55	7.66	26	85	2029	6718	15587
10	1076.41	2504.43	5.41	25	261	4548	14984	26264

Table 7: Airline response times in ms, packet loss on Airline-OPC link, 50 Users

operations required for storing time-stamps to a background task. Our measurements of response times in Section 3.1 are obtained from these time-stamps in off-line analysis. The recorded times are not multiples of the polling interval as requests are generated in a random process.

Fourth, we changed the OPC to return the status *Pending* also on non-existent orders. This addresses a problem that can occur if an order is not yet stored in the OPC when the first status check occurs. Our experiment setup ensures that each order that may be requested has been generated previously, and consequently we cannot run into the situation that we access a non-existent order.

Finally, we added the necessary components to inject network faults and to generate load. We use the Linux NetEm module [24] to inject IP packet loss, and we generate load using Apache JMeter 2.3.2 [21].

The OPC, Airline, Activity and Lodging services are each deployed on an Intel Pentium 4 computer (3 GHz, Hyperthreading enabled) with 512 MB of RAM and Linux 2.6.22. Due to limitations in the available hardware, the Bank and OPC services currently share the same server. The Website is installed on an Intel Core 2 Duo machine with 2 GHz and 2 GB RAM running under Mac OS X 10.5.6. The JMeter load generator runs on an AMD Athlon 64 (2 GHz, 1 GB RAM) with Linux 2.6.24. All computers are connected by a dedicated 100 Mbit network routed through an Intel Pentium 3 machine (700 MHz, 384 MB RAM, Linux 2.6.12) on which fault injection takes place. We use the Sun GlassFish application server 9.1_02, build b04-fcs [18].

3. MEASUREMENTS

In our experiments, we measure response times for three different load scenarios (10, 25, and 50 simulated users). The workload consists of repeated requests with delays following a Normal distribution with parameters $\mu = 300$ ms and $\sigma =$

Run	Mean	SD	c^2	Min	50%	95%	99%	Max
1	6099.66	4986.61	0.67	188	4722	16075	22442	48994
2	5647.03	4407.69	0.61	160	4576	14041	20337	44101
3	5596.60	4834.83	0.75	209	4093	14345	20635	44103
4	7773.61	6563.63	0.71	183	6085	23745	30284	46666
5	5917.18	5065.41	0.73	136	4717	15896	26243	38837
6	5900.61	4739.55	0.65	149	4807	14531	23320	37285
7	5056.15	3844.13	0.58	234	3810	11951	17684	31676
8	5799.99	4668.77	0.65	142	4753	14949	22461	47444
9	5957.10	4914.41	0.68	204	4585	15304	26516	50706
10	7614.09	7058.60	0.86	160	5662	19753	37136	72429

Table 8: OPC response times in ms, packet loss on Airline-OPC link, 50 Users

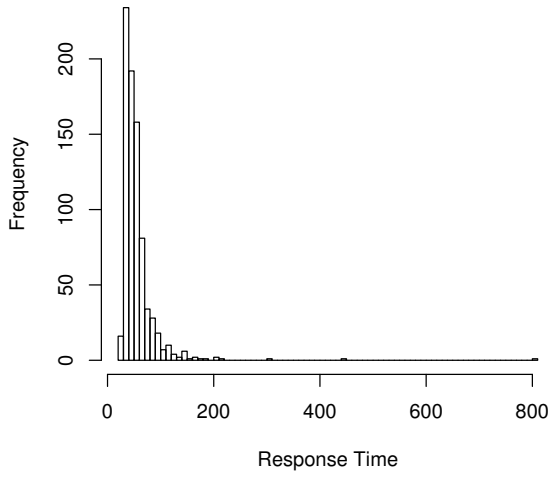

Figure 2: Distribution of response times for the Airline (no packet loss, 10 users)

100. We run experiments without packet loss and with 3% packet loss on the network link between the OPC and the Airline supplier, and the network links between the OPC and the Airline and Activity suppliers, respectively (indicated by 'FI' in Figure 1). In this work we report on preliminary results for the 10 and 50 users scenario, without and with loss on the link between the Airline and the OPC.

We repeat each experiment ten times, restarting the system before each run in order to avoid software aging artifacts. Since the system exhibits pronounced startup and breakdown phases (see Figure 6, for instance), we crop the data sets to the stable phase prior to analysis.

Each experiment run with 10 users results in 1000 observations of which we drop the first 150 and last 50 measurements leaving us with 800 observations for our analysis. The experiment runs with 50 users result in 5000 observations of which the first 1500 and last 500 are deleted.

3.1 Results

Tables 1–8 present statistical summaries of our results. Since we observe strong variations between the individual runs, we opted to treat the results separately in this preliminary analysis.

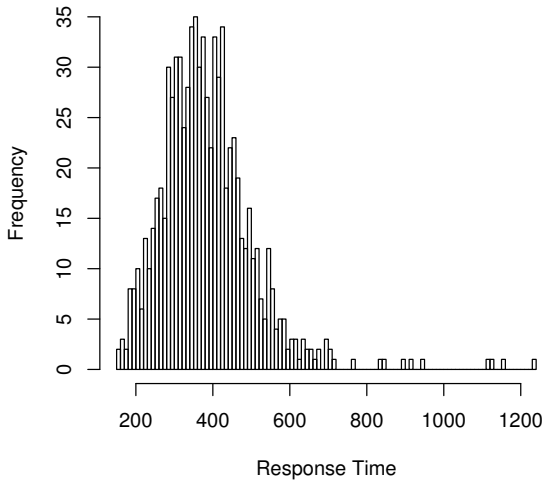

Figure 3: Distribution of response times for the OPC (no packet loss, 10 users)

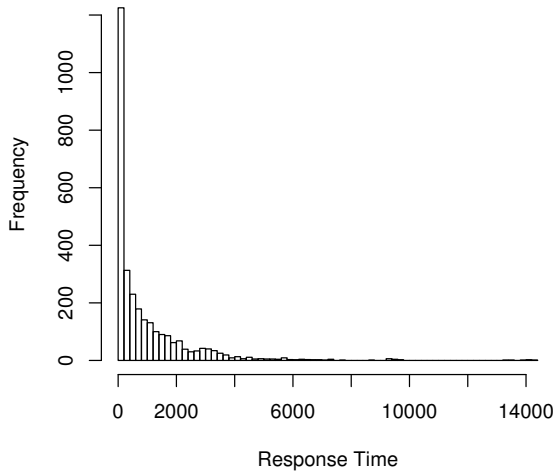

Figure 4: Distribution of response times for the Airline (no packet loss, 50 users)

Tables 1–4 show response times for the Airline and the OPC in the scenarios with 10 and 50 users. In these experiments, no packet loss was injected. In the low-load scenario (10 users), both the Airline and the OPC have low response times. We also note that the response times exhibit low variance, as indicated by the standard deviation SD and the squared coefficient of variation $c^2 = \sigma^2/\mu^2$. The 95 and 99 percent quantiles as well as the maximum show that most service invocations finish very fast. This is corroborated by the histograms of response times, shown in Figures 2 and 3 (we selected the first run of each scenario to draw the histograms).

In contrast, in the high-load scenario (50 users, Tables 3 and 4) both the mean response time and the variance are much higher for the Airline and the OPC. Furthermore, from the histograms we note that the response time distributions in these scenarios tend to develop a long tail. We attribute these changes to the fact that with 50 customers the system was already driven towards an overload situation, possibly caused by problems in the software itself and by the low-end hardware. In particular, in the graphs for the response times over time (Figures 6 and 7) we observe periodic increases of response times. We suspect that these spikes are due to the Java garbage collector attempting to free memory by de-allocating unused objects (a similar effect has been observed in [8]). In our experiment setup, the garbage collector interval was set to 3600 ms, which appears to coincide with the intervals of the spikes. However, further work is required in order to establish the effects of the garbage collector on completion times.

Please note that the variability in all data of the OPC is much lower than in the data for the Airline. This can be explained by the polling mechanism which introduces time slots and hence reduces randomness. This effect is most pronounced in the low-load scenarios.

We now turn our attention to response times with 3% IP packet loss on the network link between the Airline and the OPC (Tables 5–8 and Figures 8–11). First, we note that increasing the load from 10 to 50 users again results in an increase of the variance, as should be expected. More important, however, is the observation that with loss the variance is higher than without loss, even in the low-load scenario. Furthermore, we see that in the low-load scenario there are three runs (2, 4, and 5) where the 99% quantile is above 3000 ms. From previous work [12, 13, 15] we know that packet loss during the connection-setup phase of the TCP results in message delays in the order of the RTO timeout values, i.e. 3 s, 6 s, 9 s, ..., and consequently we attribute these observations to packet loss during the connection-setup phase. On the other hand, we note that the bulges around the RTO timeouts are not as pronounced here as in our previous work. This difference will be studied in future work. Among the possible reasons are the comparably low number of observations in our current experiments (also corroborated by the large variation in c^2 over the experiment runs), larger message sizes (which allow the TCP to apply more efficient fault-handling mechanisms), and also a different packet loss model (cf. [15], where we used a Gilbert model).

Considering the high-load scenario with packet loss, we observe that response times are very large. While mean response times for the Airline are still around 800 ms, response times for the OPC are already in the order of several seconds. Furthermore, response times vary greatly, as indicated by the large standard deviation and the 95 and 99 percent quantiles as well as the maximum. In one case (run 10), the maximum response time of the OPC was above 60 s. With the Airline, response time quantiles also seem to indicate the effect of the TCP RTO timeout, as discussed above, although upon closer inspection we could not identify a spike at e.g 3 s in the histogram (Figure 10).

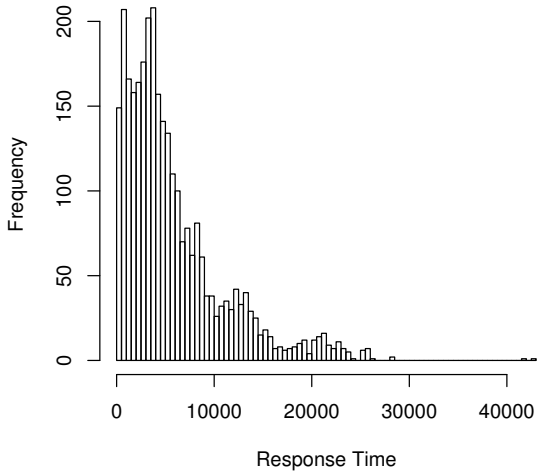

Figure 5: Distribution of response times for the OPC (no packet loss, 50 users)

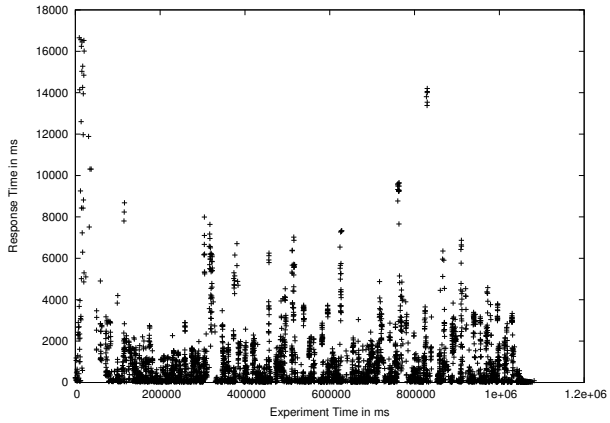

Figure 6: Response times over time for the Airline (no packet loss, 50 users)

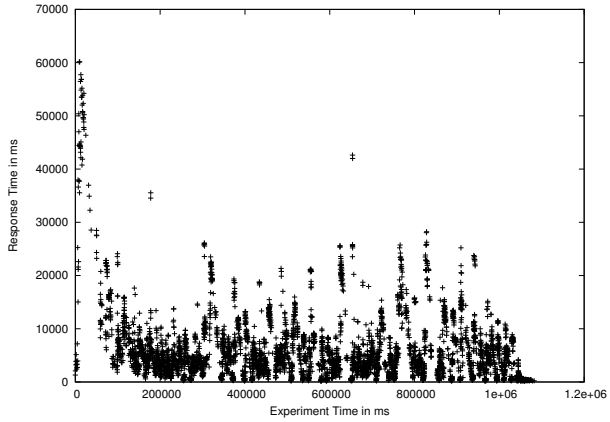

Figure 7: Response times over time for the OPC (no packet loss, 50 users)

4. PHASE-TYPE MODELS FOR RESPONSE TIMES

In order to use the observed data in further theoretical studies, we must derive appropriate models for the response times. As a first step, we choose to describe system behaviour by phase-type models for the response times. In this section we first give a basic background on phase-type distributions, and Hyper-Erlang distributions in particular. We then derive phase-type distributions to model the response times observed in the experiments.

4.1 Background

Continuous phase-type (PH) distributions represent the time to absorption in a CTMC with one absorbing state [11]. PH distributions are commonly specified as a tuple $(\boldsymbol{\alpha}, \mathbf{Q})$ of the initial probability vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ and the sub-generator matrix $\mathbf{Q} = (\lambda_{ij})_{1 \leq i,j \leq n}$, which contains the transition rates λ_{ij}. The probability density function, cumulative density function, and kth non-central moment, respectively, are defined as follows [11, 7, 20]:

$$\begin{aligned} f(x) &= \boldsymbol{\alpha} \mathrm{e}^{\mathbf{Q}x} \mathbf{q}, \text{ where } \mathbf{q} = -\mathbf{Q}\mathbb{1}, \\ F(x) &= 1 - \boldsymbol{\alpha} \mathrm{e}^{\mathbf{Q}x} \mathbb{1}, \\ E\left[X^k\right] &= k! \boldsymbol{\alpha} (-\mathbf{Q})^{-k} \mathbb{1}. \end{aligned}$$

Within the general PH class, acyclic phase-type distributions (ACPH) have received particular attention for practical applications [7, 22]. Hyper-Erlang distributions (HErD) form a sub-class of the ACPH class. A HErD is a mixture of b Erlang distributions with rates $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_b)$ and lengths $\mathbf{m} = (m_1, \ldots, m_b)$ The probability of entering each Erlang branch is specified by $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_b)$.

4.2 Response Time Models

We employ the moment-matching method from [20] to derive ACPH(2) models of the response times. The algorithm attempts to compute $(\boldsymbol{alpha}, \boldsymbol{Q})$ such that the first three moments of the corresponding phase-type distribution match those of the empirical data. Where the first and second moment can be matched, but the third moment is out of bounds for an ACPH(2), the resulting ACPH(2) only approximates the third moment. As the ACPH(2) class cannot match the first three moments of every distribution, we also employ the G-FIT tool [22] to fit Hyper-Erlang distributions to those data sets where no ACPH(2) can be found. Due to time constraints we only present approximations for the data from the first run of each experiment. Models for all data sets will be made available at [14].

Table 9 presents the resulting ACPH(2) models. The third moment of the response time distributions of the Airline in the run with 10 users and no packet loss, and those for the OPC with packet loss could only be approximated (marked by an asterisk in Table 9). We note that for the data sets with low c^2 the approximation tends towards a mixture between an Erlang(2) and an Exponential distribution. In particular, for the first data set the approximation is very similar to an Erlang(2), as $\lambda_1 \sim \lambda_2$ and α_1 is close to 1.

Table 9 does not contain an entry for OPC response times with 10 users and no packet loss. Recall from Table 2 that these have a squared coefficient of variation $c^2 = 0.10$. Since

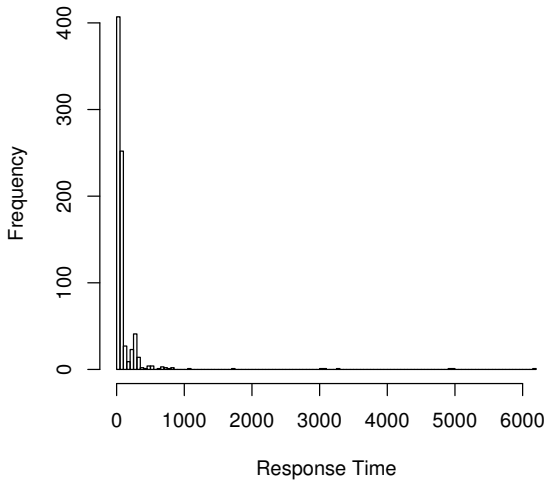

Figure 8: Distribution of response times for the Airline (packet loss, 10 users)

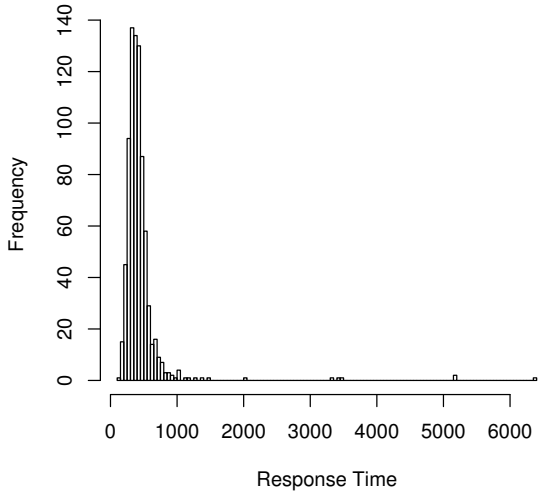

Figure 9: Distribution of response times for the OPC (packet loss, 10 users)

Data Set	α_1	λ_1	λ_2
nl-10-airline-1*	0.98199	0.035568	0.035557
nl-50-airline-1	0.25552	0.000489	0.002390
nl-50-opc-1	0.62461	0.000215	0.000346
l1-10-airline-1	0.03317	0.000660	0.013733
l1-10-opc-1*	0.44850	0.003257	0.003257
l1-50-airline-1	0.24634	0.000501	0.002756
l1-50-opc-1*	0.68751	0.000277	0.000277

Table 9: ACPH(2) models ($\alpha_2 = 1 - \alpha_1$). Data sets where the third moment was approximated are marked by an asterisk ('*').

for every phase-type distribution

$$c^2 \geq \frac{1}{n}$$

holds [1], we need a phase-type distribution of at least order $n = 10$ to approximate this distribution. Using the G-FIT tool we found that the Erlang distribution with $k = 10$ and rate $\lambda = 0.025943$ (i.e. a HErD with only one Erlang branch) fits this distribution best.

4.3 Evaluation

Table 9 shows that for three data sets the third moment could only be approximated by an ACPH(2). Furthermore, one data set required a model of at least order $n = 10$. For these four phase-type distributions where there was no exact fit we compute the relative error in the first three moments,

$$e_i = \frac{|\hat{c}_i - c_i|}{c_i} \text{ for } i = 1, 2, 3,$$

as proposed in [2, 6], where c_1, c_2 and c_3 are the mean, variance and centred third moment of the data, and $\hat{c}_i$ denotes the respective moment of the phase-type distribution. The results are given in Table 10. We observe that, with the exception of the Erlang distribution for OPC response times in the 10 users, no loss scenario (nl-10-opc-1), the relative error in the first two moments is negligible. Furthermore, in all four cases there is a non-negligible error in the third moment.

Data Set	e_1	e_2	e_3
nl-10-airline-1	3.6941e-09	9.5226e-09	0.8668
nl-10-opc-1	1.7106e-14	0.0124	0.6627
l1-10-opc-1	3.4790e-08	8.2298e-08	0.8326
l1-50-opc-1	1.2283e-08	2.1674e-08	0.2817

Table 10: Relative error in the first three moments for the ACPH(2) approximations from Table 9.

5. CONCLUSION AND FUTURE WORK

In this paper we presented a preliminary analysis of the response times in a SOA testbed based on Sun's Java Adventure Builder. We found that packet loss and load increase both expected response times and variance. We presented phase-type models that capture the first three moments of the data sufficiently well.

We are currently in the process of conducting a more in-depth analysis of the data that will result in more complex phase-type models that allow us to better describe the shape of the response-time distributions. Furthermore, we will study the temporal behaviour of response times and derive appropriate models.

In the future we plan to conduct more experiments with scenarios where we vary the workload and the workflows within the system, as well as the type and frequency of injected faults.

6. ACKNOWLEDGEMENT

This work has been supported by DFG grant Wo 898/2-1.

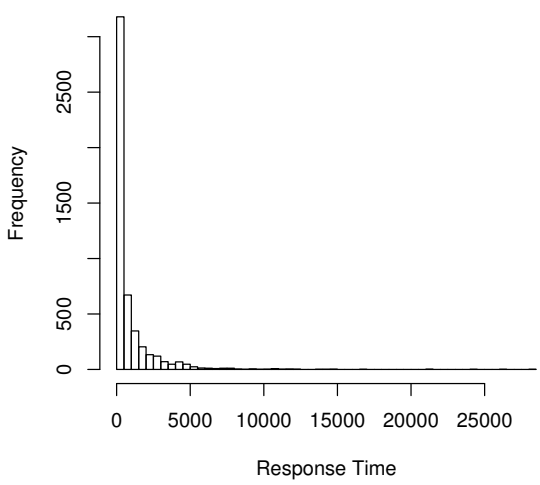

Figure 10: Distribution of response times for the Airline (packet loss, 50 users)

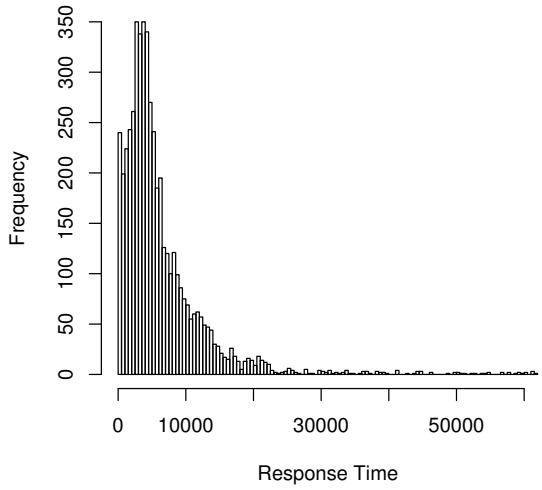

Figure 11: Distribution of response times for the OPC (packet loss, 50 users)

7. REFERENCES

[1] D. Aldous and L. Shepp. The least variable phase-type distribution is erlang. *Stochastic Models*, 3:467–473, 1987.

[2] A. Bobbio and M. Telek. A Benchmark for PH Estimation Algorithm: Results for Acyclic-PH, 1994.

[3] V. Datla and K. Goseva-Popstojanova. Measurement-based performance analysis of e-commerce applications with web services components. In *Proc. IEEE International Conference on e-Business Engineering ICEBE 2005*, pages 305–314, 2005.

[4] A. Gorbenko, V. Kharchenko, O. Tarasyuk, Y. Chen, and A. Romanovsky. The threat of uncertainty in service-oriented architecture. Technical Report 1122, Newcastle University, School of Computing Science, Oct 2008.

[5] A. Gorbenko, A. Mikhaylichenko, V. Kharchenko, and A. Romanovsky. Experimenting with exception handling mechanisms of web services implemented using different development kits. Number 1010, School of Computing Science, March 2007.

[6] A. Horváth and M. Telek. Approximating heavy tailed behaviour with Phase type distributions. In *3rd International Conference on Matrix-Analytic Methods in Stochastic models (MAM03)*, 2000.

[7] A. Horváth and M. Telek. PhFit: A General Phase-Type Fitting Tool. In *TOOLS '02: Proceedings of the 12th International Conference on Computer Performance Evaluation, Modelling Techniques and Tools*, pages 82–91, London, UK, 2002. Springer-Verlag.

[8] K. S. Juse, S. Kounev, and A. Buchmann. PetStore-WS: Measuring the Performance Implications of Web Services. In *Proceedings of the 29th International Conference of the Computer Measurement Group on Resource Management and Performance Evaluation of Enterprise Computing Systems (CMG 2003), Dallas, Texas, USA, December 7-12, 2003*, pages 113–123. Computer Measurement Group (CMG), Dec. 2003.

[9] S. M. Kim and M. C. Rosu. A survey of public web services. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 312–313, New York, NY, USA, 2004. ACM.

[10] S. Kounev and A. Buchmann. Performance Modeling and Evaluation of Large-Scale J2EE Applications. In *Proceedings of the 29th International Conference of the Computer Measurement Group on Resource Management and Performance Evaluation of Enterprise Computing Systems (CMG 2003), Dallas, Texas, USA, December 7-12, 2003*, pages 273–283. Computer Measurement Group (CMG), Dec. 2003. Best-Paper-Award at CMG-2003.

[11] M. F. Neuts. *Matrix-Geometric Solutions in Stochastic Models. An Algorithmic Approach*. Dover Publications, Inc., New York, 1981.

[12] P. Reinecke, A. P. A. van Moorsel, and K. Wolter. A measurement study of the interplay between application level restart and transport protocol. In M. Malek, M. Reitenspieß, and J. Kaiser, editors, *Service Availability. Proceedings of the First International Service Availability Symposium*, volume 3335 of *LNCS*, pages 86–100. Springer, May 2004.

[13] P. Reinecke, A. P. A. van Moorsel, and K. Wolter. The Fast and the Fair: A Fault-Injection-Driven Comparison of Restart Oracles for Reliable Web Services. In *QEST '06: Proceedings of the 3rd International Conference on the Quantitative Evaluation of Systems*, pages 375–384, Washington, DC, USA, 2006. IEEE Computer Society.

[14] P. Reinecke and K. Wolter. ACPH models for SOA response times. http://www.informatik.hu-berlin.de/∼preineck/acphmodels/.

[15] P. Reinecke and K. Wolter. Adaptivity metric and performance for restart strategies in web services reliable messaging. In *WOSP '08: Proceedings of the 7th International Workshop on Software and Performance*, pages 201–212, New York, NY, USA, 2008. ACM.

[16] P. Reinecke and K. Wolter. Phase-type approximations for message transmission times in web services reliable messaging. In S. Kounev, I. Gorton, and K. Sachs, editors, *Performance Evaluation – Metrics, Models and Benchmarks*, volume 5119 of *Lecture Notes in Computer Science*, pages 191–207. Springer, June 2008.

[17] P. Reinecke and K. Wolter. Towards a multi-level fault-injection test-bed for service-oriented architectures: Requirements for parameterisation. In *SRDS Workshop on Sharing Field Data and Experiment Measurements on Resilience of Distributed Computing Systems*, Naples, Italy, 2008. AMBER.

[18] Sun Microsystems. GlassFish Application Server. https://glassfish.dev.java.net/. Last visited 23 June 2009.

[19] Sun Microsystems. Java adventure builder reference application. https://adventurebuilder.dev.java.net/, 2006. Last seen April 28th, 2009.

[20] M. Telek and A. Heindl. Matching Moments for Acyclic Discrete and Continous Phase-Type Distributions of second order. *International Journal of Simulation Systems, Science & Technology*, 3(3–4):47–57, Dec. 2002.

[21] The Apache Software Foundation. Apache JMeter. http://jakarta.apache.org/jmeter, 2008.

[22] A. Thümmler, P. Buchholz, and M. Telek. A Novel Approach for Phase-Type Fitting with the EM Algorithm. *IEEE Trans. Dependable Secur. Comput.*, 3(3):245–258, 2006.

[23] M. Tian, T. Voigt, T. Naumowicz, H. Ritter, and J. Schiller. Performance impact of web services on internet servers. In *Proceedings of IASTED International Conference on Parallel and Distributed Computing and Systems*, Marina Del Rey, California, USA, 2003.

[24] Various authors. NetEm – LinuxNet. http://linux-net.osdl.org/index.php/Netem. Last visited October 8th, 2007.