Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Analytical Queuing Model for Telesto

Report Group 32

Dominic Langenegger

dominicl@student.ethz.ch

Advanced Systems Lab 2013
Systems Group
ETH Zürich

Supervisors:
Markus Pilman
Prof. Dr. Gustavo Alonso

December 20, 2013

Abstract

This report summarizes the work of analytically modeling and evaluating *Telesto*, the Distributed Message Passing System built by Simon Marti and Dominic Langenegger during the first milestone of the course project of the *Advanced Systems Lab* course 2013 at ETH Zurich. All relevant information about the first milestone can be found in [1].

Used concepts, mathematical formulas and theoretical aspects are heavily based on [2] and the lecture slides.

Contents

Chapter 1

Analytical Queuing Model

This chapter introduces the analytical queuing model built to model the characteristics of *Telesto*. It also includes explanations about simplifications and assumptions that were made.

We recall Table 5.3 in [1], showing in which parts of the system a request spends how much time:

Phase	Time	Relative
Waiting for client	79 µs	0.82%
Parsing request	14 µs	0.15%
Waiting for database	9.574 µs	98.80%
Responding	22 µs	0.23%

Table 1.1: Time spent on various tasks by middleware workers

Based on this data we can safely say, that the time a request is handled by the middleware is minimal in comparison to the time spent to actually handle it on the database tier. Therefore, we can simplify the queuing model for the middleware by reducing it to the database tier as seen in figure 1.1.

Notice, that this leads to a very simplified model with one queue per middleware and one service per database connection, since we can remove the service time of the middleware interaction before and after the database interaction. The database queues then become redundant and we can directly link the queues of the middleware to all database connection services.

Due to the architecture of *Telesto*, the database connection pool and the worker thread pool are two entirely separated parts and therefore each request handled by any worker thread, can be handled by any database connection out of the pool.

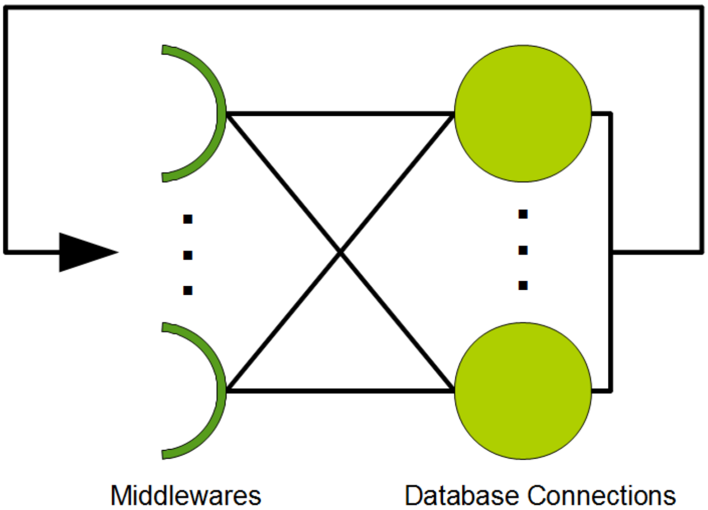

Figure 1.1: Simplified model of the closed queuing system for *Telesto*

1.1 Workload & Model

As seen in figure 1.1, we use a closed queuing network to model *Telesto*. Since all conducted tests during milestone 1 (see [1] for more details) operate with clients that only send new requests as soon as a previous on is completed, the number of jobs in the system at any time is directly corresponding to the number of clients and therefore fixed.

It is important to note, that some clients may query the middleware to retrieve a message when actually no message is waiting for them. Based on the experiments from milestone 1, we know that this is the case for roughly one sixth of all message requests.

1.1.1 Clients

Our queuing network does not include any modeling of the clients. This is because the time a job actually spends at the client is minimal since a client has only the following very simple tasks (that take almost no time):

(a) Send a message to a middleware

(b) Wait for a message

(c) Receive the message

(d) Start from (a); possibly using some information of the received message

The actual time a job is being processed at the client is in steps (c), (d) and (a), which is comparable in workload to the tasks of the middleware for message processing as seen in **??**, excluding the part where the middleware waits for the database. Since this takes very little time compared to the time spent processing a job at the database, it is save to simplify the model by omitting the clients entirely.

However the think time of the network can be determined by including the client retry delay of 100 ms into the calculation. If we assume that (as stated above) roughly one sixth of all client queries end up with no message as a result (and therefore the clients waits those 100 ms), we get a think time T of about $\frac{100ms}{6} \approx 17ms$.

1.1.2 Network

In *Telesto*each client request actually passes the network in four different phases:

1. Client → Middleware
2. Middleware → Database
3. Database → Middleware
4. Middleware → Client

Since we cannot measure the time spent in the network for the connections to the database individually, we don't model the network in between the middleware and the database separately.

For the connection from the clients to the middlewares we can assume that all messages have about the same size since all conducted tests use messages that are never longer than about 10 characters. This means, that we can assume the network delay is constant for every phase and not load dependent. Therefore it can simply be modeled as a delay center in the network model.

1.1.3 Middleware

The middleware consists of a single dispatcher thread that handles all incoming connections and puts incoming data into a queue. This can be modeled as a simple fixed-capacity server.

The many worker threads in a *Telesto* middleware instance, are modeled as load-dependent servers that then pass the job into a queue to the database connection pool. Since the thread pool scales up to the number of threads it is trivial to see, that this is load-dependent because multiple jobs can be processed in parallel.

The response from the database is then again packed into a *Telesto* network packet to be sent to the client. This again happens by the same worker thread that handled the database call. Therefore this can also be modeled load-dependent with the same properties as the parsing part of the workers.

1.1.4 Database

Every worker thread in the middleware retrieves a database connection from a theoretically independent database connection pool and uses it to execute a query on the database. Each of these connections is modeled as a separate load dependent server.

Notice that the database itself may not scale very well with increasing number of connections since its parallelism is very limited. However we think that it is very hard to model this by directly relating e.g. the number of cores of the machine to the number of service centers since the database may very well scale to many more connections than cores for several reasons like idle time and partial occupancy or internal parallelization optimization.

1.2 Notation

This section provides a short overview of the used notation as seen in [2].

Arrival Process
: τ_i is the **interarrival time** between two arrivals at t_{i-1} and t_i.

Service Time Distribution
: The distribution of time spent by a job in a specific service.

Number of Servers
: The number of identical and independent servers for one particular service

System Capacity
: The maximum number of jobs allowed in the system. This is usually always ∞.

Population Size
: The maximum number of potential jobs that can ever enter the system. This is usually always ∞.

Service Discipline
The order jobs are served from a queue like *First In, First Out* (FIFO), *Round Robin* (RR) or *Last In, First Out* (LIFO).

1.3 Parameters

Every device and the whole system modeled in the analytical queuing network for *Telesto* is specified by some parameters. These are (based on [2]):

Service Type t
The type of the device. I.e. fixed capacity, delay center or load dependent

Service Time s
The time needed to complete one job by a server

Visits v
The number of visits to a device

Service Rate μ
The service rate of a device given the number of jobs in it. This measure is used for load dependent servers only.

Number of Clients n
The number of clients in the system sending requests.

Think Time z
The additional delay between two requests. In *Telesto* a client only waits for a certain *client retry delay* of usually $100ms$ if a query for a message didn't return any message, which is the case in roughly 25.8% of all queries as seen in table 1.2.

To determine these values for all components of the network model, we used the data from our benchmarks during milestone 1 as shown in table 1.2.

Measure	Value
Mean Packet Throughput	$2498\ s^{-1}$
Mean Message Throughput	$927\ s^{-1}$
Mean Response Time	10.16 ms
Mean Packets With Wait Time	$2498 - 2 \cdot 927 = 644s^{-1} \approx 25.8\%$
Mean Client Wait Time (Think time)	$100ms \cdot 25.8\% = 25.8ms$

Table 1.2: Important measurement data of *Telesto*

Table 1.3 shows the values for each of these parameters.

	Service Type t	**Service Time** s	**Visits** v	**Think Time** z
Client				25.8ms
Network	Delay Center	1.30ms	2	
Dispatcher	Fixed Capacity	0.01ms	1	
Worker In	Load-Dependent	0.01ms	1	
Worker Out	Load-Dependent	0.02ms	1	
Database	Load-Dependent	8.01ms	1	

Table 1.3: Parameter values for devices in the *Telesto* network model

1.3.1 Service Rate μ

The service rate has to be treated specially for the load dependent centers at the database and the middleware.

TODO state clearly how they are modeled **TODO** equation like:

$$\mu(n) = \begin{cases} n/S & n < m \\ m/S & n \geq m \end{cases}$$

Chapter 2

Analytical Analysis

Based on the parameters and model introduced in the previous chapter this chapter gives details about the executed performance analysis.

2.1 Algorithm

To perform the analysis of the whole closed queuing network of *Telesto* we use Mean-Value Analysis (MVA) as introduced in section 34.2 of [2]. Since we have multiple load-dependent centers within our model, we in particular use the *MVA Including Load-dependent Centers* algorithm from Box 36.1 in [2].

2.2 Queuing Network

The following subsections show the results of our simulations in comparison to some of the measurements and benchmarks we did on *Telesto* during milestone 1.

2.2.1 Combined Database Connection and Worker Thread count

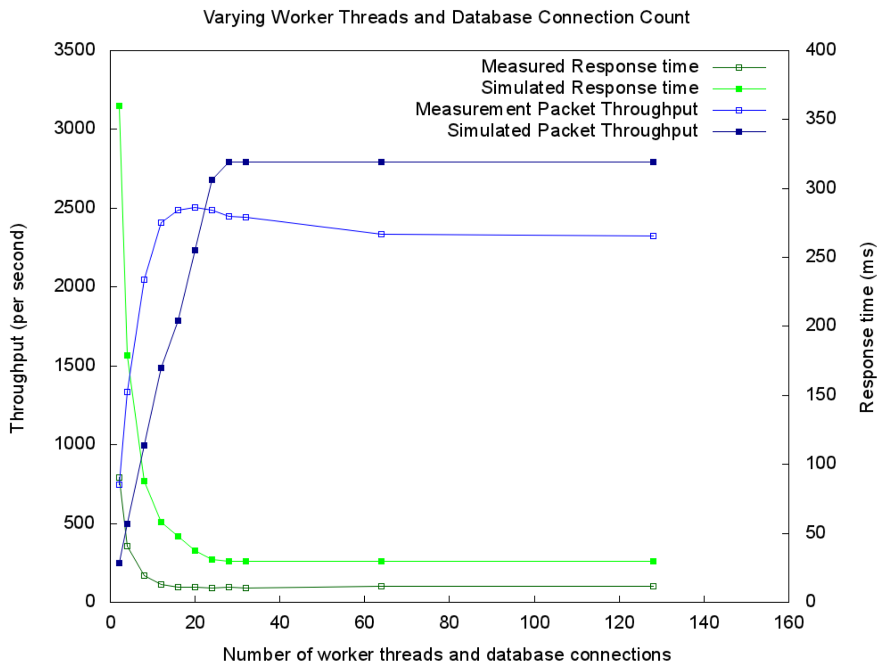

Figure 2.1: Varying number of worker threads and database connections

Figure 2.1 shows the comparison of both the response time and the throughput of the simulation and experiments varying the number of worker threads and database connections simultaneously.

2.2.2 Client Count

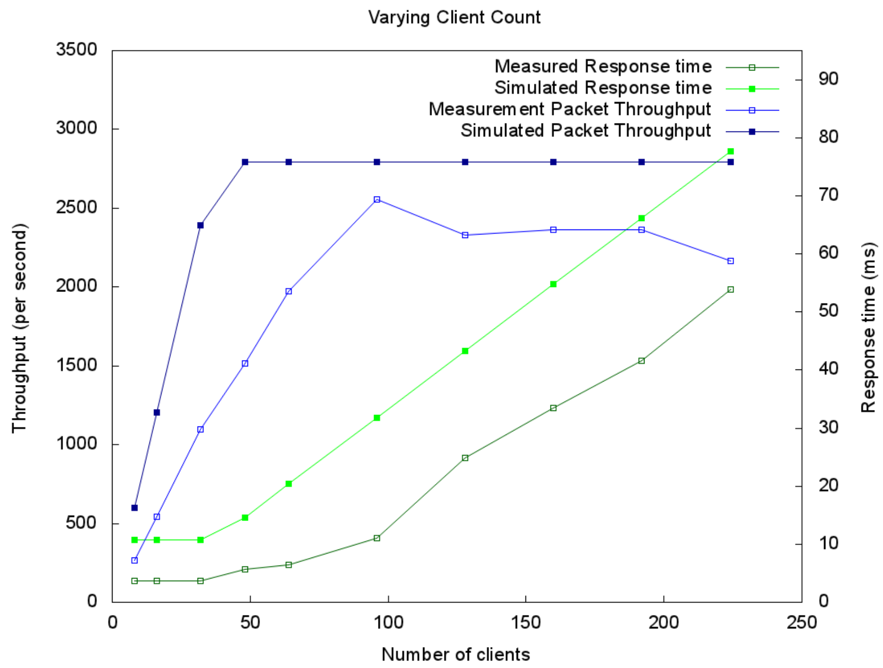

Figure 2.2: Varying client count

In figure 2.2 we plotted the response time and throughput for *Telesto* when varying the number of clients and therefore jobs in the network.

2.2.3 Client Retry Delay

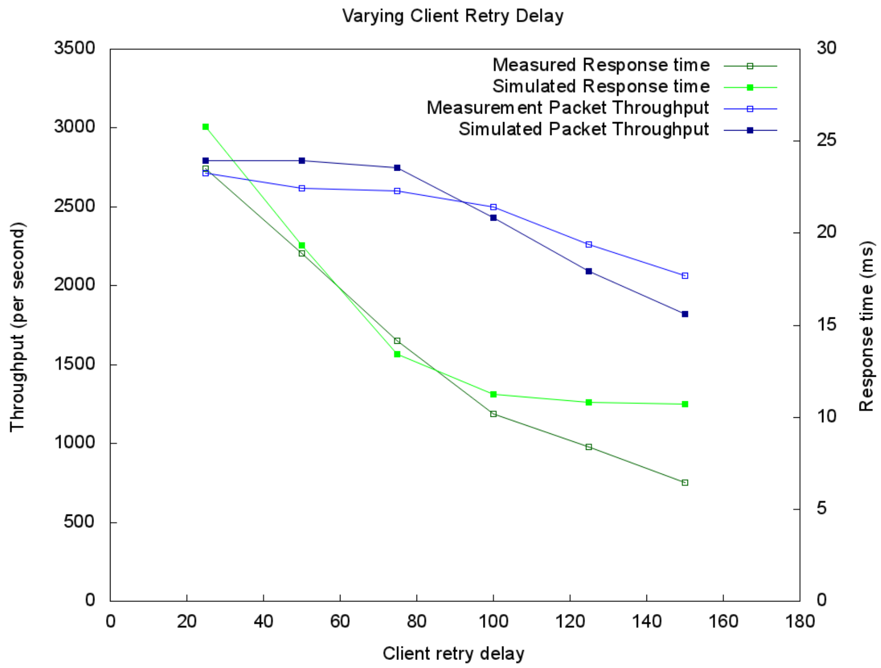

Figure 2.3: Varying client retry delay (i.e. think time)

Since the network think time is directly dependent on the client retry delay in *Telesto*, we also simulated this experiment with our MVA model. The results are shown in figure 2.3.

2.3 Database

2.4 Bottleneck

Chapter 3

Conclusion

3.1 Multiple Middlewares

TODO some text here to justify why this wasn't modeled and tested

Bibliography

[1] Langenegger, D., Marti, S.: Telesto - A Distributed Message Passing System. Advanced Systems Lab, ETH Zurich (November 2013)

[2] Jain, R.: The Art Of Computer Systems Performance Analysis:. Wiley India Pvt. Limited (2008)