

TTM-9 Traffic and Dependability, Tools and Method Laboratory.

**Verktøy og metodikk for studium av trafikk og pålitelighet,
laboratorium.**

First version prepared by
Peder J. Emstad, Poul E. Heegaard and Bjarne E. Helvik 2001-08-28

Revised (and Lyxified) by
Bjarne E. Helvik and Anders Mykkeltveit
September 2007
[Latest bug and ambiguity fix BEH; 2008-12-18]

1 Introduction

This document defines the in depth study subject *TTM9 Traffic and Dependability; Tools and Method Laboratory* given as an element in the specialisation in dependability and performance at the Department of Telematics, the Norwegian University of Science and Technology.

1.1 Objectives

The objective is to make the student familiar with the common tools and techniques for investigating the dependability and the traffic handling properties of telematic systems. Emphasis is put on the carrying out of systematic and well documented investigations, as well as extraction and presentation of the findings in a style suited for publication.

1.2 Outline of tasks

The lab consists of three tasks outlined below. The first task is somewhat larger than the last two tasks which have an approximately equal workload. It is required that all tasks are done satisfactorily in order to pass the subject. All tasks are based on the simplified network server cluster presented in Section 2.

1. *Simulator*. Based on the description of the network server cluster, its environment and operation policies, a DEMOS based simulator shall be developed. With this simulator, a series of experiments shall be carried out. The results will be used in the succeeding tasks.
2. *Data analysis*. The objective of this task is to perform statistical analysis on a log of events which may stem from the operation of a system, a simulation, measurement equipment etc. Simulation traces are used.
3. *Evaluation*. The results obtained from simulation are compared to results obtained from other models of the system.

Note that the output from the simulation study (trace files) will be used for data analysis in Task 2, and simulation results shall be used for comparison with analytical results in task 3. Hence, it is advised to study all tasks before starting the definition of the simulation and the simulation experiments. Furthermore, the experiments are not specified in detail, and it is a part of the “lab work” to decide which parametrization to choose, which simulation experiments to run etc. The suggested default value for each parameter should be regarded as a reference point, not *the* value.

It is strongly advised that the students establish a proper book-keeping, encompassing an (up-dated) plan for the work and of work done. (what is investigated, i.e system configuration and parametrization; where are the results; what is observed, etc.).

1.3 Dependability and performance study

The lab is divided into two parts - dependability and performance study. The students may choose to study one of these aspects of the system. In the rest of this document, the prefix [Performance] means that the following text is only relevant for the performance study, while the prefix [Dependability] means the same with respect to dependability.

1.4 Tools

The recommended tools for the work are identified below. The students may use other tools with similar capabilities. However, in this case the students must validate that the tools applied are able to solve the problems at hand and no support can be expected with regards to “technical problems” with other tools than those listed below.

1.4.1 Simula/DEMOS

The Simula/DEMOS simulation language/tool should be familiar to the student. A very brief introduction, documentation and how to access the tools is found on the web page <http://www.item.ntnu.no/fag/ttm9> under the link Simula/DEMOS.

1.4.2 Mathematica

The students should have gained some familiarity with Mathematica from previous courses. Mathematica is a universal tool e.g. symbolic and numerical mathematics, plotting. For a brief introduction, documentation and how to access the tools is found on the web page <http://www.item.ntnu.no/fag/ttm9/> under the link Mathematica.

[Dependability] For dependability analysis in task 3, special packages and notebooks demonstrating the application are available. They are available at <http://www.item.ntnu.no/fag/ttm4120/current/mathematica.php> at the bottom of the page. If there are problems in obtaining the packages contact Bjarne.E.Helvik@item.ntnu.no.

2 System description

The system description contains information relevant to both the [Dependability] and the [Performance] of the system. It is therefore up to the student to find out what is related to performance and what is related to dependability.

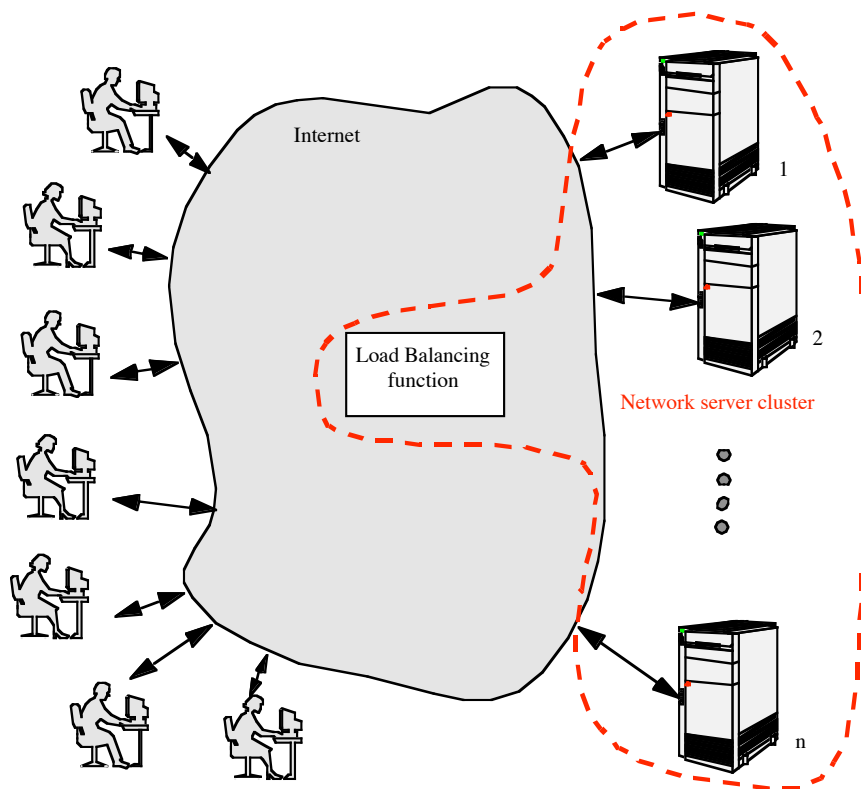


Figure 1: Sketch of system studied

2.1 System structure

A network server cluster is regarded. Each server in the cluster is identical. The servers operate in load sharing, where several load sharing strategies may be applied. The implementation of the various load sharing strategies are taken care of by a load balancing function within the network. This load balancing requires system information and processing which have a cost. The load balancing function is considered as a part of the system, but the equipment and software performing it is not included.

The system has a sufficient performance and is said to be working when at least m out of the n servers are in operation.

Parameters

The most important parameters for the system structure are listed below, each with a suggested variable name for the simulator and a typical value.

- Number of servers, n ; INTEGER numServers; 3

- Number of servers required for sufficient performance, m ; INTEGER oprServers; 2
- Maximum queue lengths in each server; INTEGER maxQ; 10

2.2 Workload and load balancing

The number of users in the system is so large that it may be regarded as infinite. Each user represents a small and equal workload. Handling of tasks from the same user is such that all tasks offered to the server cluster may be regarded as independent of each other and identically distributed. A task may be served by any working server, and only one visit to a server is required to complete the task. The distribution of the workload per task is unknown. The load balancing function in the network, cf. Figure 1, may assign tasks to servers according to candidate policies:

RR (Round Robin). Tasks are assigned to the n servers in a sequential circular order, i.e. $1, 2, \dots, n - 1, n, 1, 2, \dots$. Tasks are only assigned to working servers. Each server maintains a local FIFO queue of the tasks assigned to it. This queue has a fixed size and tasks arriving when the queue is full are lost. Queued tasks are lost when the server fails.

VQ (Virtual queue). The load balancing function maintains one virtual central queue for all servers. The queue ordering is FIFO and tasks arriving when the virtual queue is full are lost. Tasks are assigned, without delay, to servers in the order servers complete their task. Server failures do not affect queued tasks. This is a reference policy and requires usually too much system information to be applicable in a practical system.

RO (Random Order). The load balancing function passes tasks to working servers in a random order. Each server maintains a local FIFO queue of the tasks assigned to it. This queue has a fixed size and tasks arriving when the queue is full are lost. Queued tasks are lost when the server fails.

Parameters

The most important parameters for the system structure are listed below, each with a suggested variable name for the simulator and a typical value.

- Expected task inter-arrival time (inverse of task arrival intensity); REAL timeNextTask; 25 ms
- Expected task processing time; REAL timeProcessTask; 50 ms
- Distributions of task processing time (processing times are i.i.d.)
 - Negative exponentially distributed;; Default
 - Deterministic;
- Load Balancing, i.e. RR, VQ, RO; INTEGER lbPolicy (1 => RR, 2 => VQ, 3 =>RO); 1

2.3 Failure modes and fault handling

A server is either intact or defect, i.e. it never works with reduced capacity. Servers are not physically co-located and has no tight interaction. Hence, it is assumed that the servers fail independently of each other. The failure process is assumed to be Poissonian. Failures are caused

by two kinds of faults, permanent and temporary. The permanent faults include solid hardware faults and environment failures. The temporary ones include logical faults in SW and HW, transient HW faults and intermittents. With a certain probability, a failure is immediately recognized as caused by a permanent fault. Otherwise it is sought handled as caused by temporary fault.

2.3.1 Handling of temporary faults

A fault, not immediately recognized as permanent, is handled as transient according to the following procedure:

1. The failed server is attempted restarted after some tasks are dropped and a default state is set for the system (without a full reload of the software). This takes a randomly distributed time (unknown at the time of the simulator development) and has a certain probability of success.
2. If the restart fails, a full reload of the server is attempted. This takes a randomly distributed time (unknown at the time of the simulator development) and has a certain probability of success.
3. If also the reload fails, the fault fault is considered as permanent.

2.3.2 Handling of permanent faults

A repair of the permanent fault(s) of the cluster is (are) initiated when a number of servers have failed. We denote the number of failed servers before repair is initiated as k . k should be in the range $0 < k \leq n - m$. All failed serveres are repaired during the same repair action and are put simultaneously into operation. The time from initiation of repair to all servers are repaired is random. The distribution of the repair time may be assumed to be independent of the number of failed servers. The distrubtion is not known at the time of the simulator development.

Parameters

The most important parameters for the system structure is listed below with a suggested variable name for the simulator and a typical value.

- Expected time between failure of a server for all kinds of faults (inverse of failure intensity). Inter-failure times are identically and independently negatively exponentially distributed; REAL timeNextFailure; 17 h
- Probability that a fault is immediately diagnosed as a permanent fault; REAL probbPermFailure; 0.1
- Probability that a restart is successful; REAL probbRestarted; 0.8
- Probability that a reload is successful; REAL probbReloaded; 0.8
- Expected duration of a restart; REAL timeRestart; 15 s
- Expected duration of a reload; REAL timeReload; 150 s

- Expected duration of a manual repair; REAL timeRepair; 1.5 h
- Distributions of the duration of **a)** restarts, **b)** reloads and **c)** manual repairs are (for the sake of simplicity) assumed to be either
 - Negative exponentially distributed;; Default
 - Deterministic;

The different kinds of failur/fault handling have their individual distribution. All times of the same kind are i.i.d.

3 Simulation (Task 1)

3.1 Simulator development

A parametrized simulator of the system described in Section 2 shall be developed.

3.1.1 Input

By parametrized is meant that the simulator shall be able to run various simulation scenarios without alteration and recompilation of the simulator code. Key parameters and their name is and default values with respect to:

- System structure,
- Workload and load balancing,
- Failures and fault handling,

are given in Section 2. Note that the list does not claim to be exhaustive and it is left to the student to define the complete list of parameters. In addition

- Simulation parameters:
 - Total simulation time and or the number of events (of which types?) that shall be observed; e.g. REAL timeProcessJob
 - Simulation transient period
 - Random seed for the simulation experiment; REAL simseed
 - Eventual replications for interval estimation/error control.

All system parameters shall be read from file.

3.1.2 Output

Trace files necessary for the detailed analysis in Task 2 shall be generated as well as data necessary for the analysis in task 3

In addition to this, directly generated statistics from DEMOS which at least presents:

- [Performance] Carried traffic in the cluster and on each server.
- [Performance] Relative and absolute number of lost tasks in the cluster and on each server.
- [Performance] Task turn around time.
- [Dependability] The availability of the system and each server.
- [Dependability] Up times and down times of the system and each server.

3.2 Validation study

Most of the use of the simulator will take place in connections with Tasks 2, and 3. In this task, the students should validate that the simulator is correct, i.e. make it plausible that the simulator is a correct representation of the system described in Section 2. The student may choose a few simple parametrization/configurations of the simulator that may be verified by theoretical results, or validation may be done by comparison with other independently obtained simulation results .

3.3 Documentation

A concise documentation including the following items should be prepared as a hand-in:

1. A user guide to the simulator and use of the simulation including the its environment, cf. Section 4.1.
2. Design documentation of the simulator.
3. Format of the trace files.
4. Results from the introductory investigations/validation, cf. Section 3.2.

Hints

If bins are used, variables must be used to store the intermediate values and the final values one wants to write to file (ex., delays and task turn around time).

If waitqueues are used, the simulations will be slower and `cim -m<tail> filename` must be used to compile with increased memory. For instance: `cim -m10 test.sim` will allow the simulator to use 10 Mbytes of memory.

To save time needed on task 2, make sure the output files generated by the simulator are convenient to use for the data analysis. Also, keep in mind that Mathematica is not optimized to handle huge amounts of data.

4 Data analysis (Task 2)

4.1 Planning production runs

As a start, the students should get familiar with the computation requirements for the simulator, so he/she can plan production runs. Execution of series of simulation experiments may very re-

source demanding in terms of computer time/resources. Although the system studied in this lab is not very complicated and the simulation time will therefore not be extremely long. However, it is advantageous to roughly estimate the simulation time needed to obtain all the results that are needed. The following tests are suggested:

- [Dependability] Try different PDF for the fault handling procedures, i.e. the timeReload, timeRestart, and timeRepair.
- [Performance] Try increasing the timeProcessJob by 50%, 100%, and 150%. Comment the “accuracy” of the observations.

Make rules of thumb with respect to required execution times for generating the results needed to produce the graphs, as well as size of the trace files used. Based on these results, answer the following questions

- Is it worth the extra effort by creating scripts to automate the dissemination and start of simulation runs, as well as collection of results?
- Are the simulations so long that they need to be run in parallel on a different computers, or can the experiments be executed on a standard PC? (The Department of Telematics has a MOSIX-based computation cluster called *Pride* which may be used if necessary)

4.2 Quantities to be extracted from the data

The trace files of the simulation shall be analysed with respect to the following quantities.

4.2.1 Dependability

1. Down time distribution of a server and of the entire cluster. The distribution shall be presented as smoothed graphs representing the pdf of the down times,
2. The distribution of the interval availability of a server and of the entire cluster. These availabilities shall be presented for various interval lengths and compared in the same graph, where the objective is to compare the shape of the distributions. Indicate also the asymptotic unavailabilities with 60% and 90% confidence intervals.

4.2.2 Performance

1. The service time distribution, i.e. the the distribution of the task turn around times for the three load balancing policies.
2. The expected waiting time and the loss probability for the three load balancing policies when
 - the offered traffic varies from 0.1 to 0.9 Erlang per server,
 - the queue-size varies. NB different for the three policies.

4.3 Documentation

The results from the analysis shall be presented as “publication quality” graphs in a brief report. The graphs should be properly titled and annotated with respect, legends, axis, parametrization, etc.

5 Evaluation (Task 3)

The objective of this task is to compare the results from the simulator with results obtained from other models of the system. In addition, the question in Section 5.3 is mandatory for the dependability study.

5.1 [Performance] Queueing policies to be studied

Assume the service time is negative exponentially distributed.

1. Model the virtual queue (**VQ**) policy with the $M/M/n/\infty$ queueing model and approximate the loss probability with the probability that the system state is at or above the maximum queue-size.
2. Model the virtual queue (**VQ**) policy with the $M/M/n/n+s$ queueing model where s is the queue-size. Draw the state transition diagram, set up the linear set of state probabilities and find expressions for conditional expected waiting time and loss.

Make plots of expected waiting time and loss against offered traffic and compare with simulation results.

Optionally, the following subtasks may also be carried out for an improved insight into the effect of the various queueing policies:

3. Model the Round Robin (**RR**) policy with the $G/M/1/s+1$ model where s is the queue-size (formulas given in Appendix).
4. For the virtual queue policy plot against buffer size a family of curves for $n=2, 3$ and 4 showing how much traffic can be offered and still meet the stated loss (say 5%).

5.2 [Dependability] Comparisons of the results from various approximate techniques

1. *Approximate analysis by reliability block schemes.* Make the independence assumptions necessary to establish reliability block schemes of a server and entire network server cluster. Determine the expected down times and availabilities. Compare to items 1 and 2 of Section on the preceding page obtained with n.e.d. service restoration times.
2. *Analysis by Markov model.* Establish a Markov model (state diagram) of a server. Determine the expected down time and availability and compare with the results from a single server obtained in the item above as well as items 1 and 2 of Section on page 9.

3. *Service restoration time distribution.* Compare the quantities obtained in item on this page above with simulations of deterministic service restoration times with the same expectation. Optional: Extend the model of a server to Erlang- k distributed service restoration times and introduce the results in the comparison for $k = 2, \dots$
4. *Larger Markov model.* This item is optional. Establish a Markov model (state diagram) of a system with more than one server, e.g. $n = 2$. Consider to use the techniques based on Krönecker sums from the ‘StateDiagrams.m’ mathematica package or developing a model based on the symmetries between servers. Compare with the results from a cluster obtained in item on the previous page above as well as items 1 and 2 of Section on page 9.

5.3 [Dependability] Transient behaviour

5. *Relaxation time.* Make a plot of the instantaneous availability of a server when it is “started” in all its states at $t = 0$. Note that it is asked for instantaneous availability, not the interval availability. The plot may be obtained by simulation¹, by finding the transient solution of the model in item 2 above, or preferably both. The relaxation time of the system commented upon with respect to parameters and initial state.

5.4 Documentation

The results from the comparisons shall be presented as “publication quality” graphs and briefly commented upon in a short report. The graphs should be properly titled and annotated with respect to scale, legends, axis, parametrization, etc. The comment should also discuss whether discrepancies between simulation and analytical results are due to approximations in the analytical methods or stochastic variations in the simulations.

¹In this case, consider how the transient information most efficiently may be obtained.

A The G/M/m queue with finite waiting room

Results for the G/M/m/s+1 queueing model is found in [Hokstad 75]. In the case of one server and with finite waiting room of s queueing places the arrival congestion under stationary conditions is found as

$$B = \frac{A^*(\beta)}{G(s)} \quad (1)$$

where $A^*(\beta)$ is the Laplace-Stieltje transform (LST) of the interarrival distribution, $A^{*(i)}(\beta)$ its i th derivative and β is the service intensity. Further

$$G(0) = 1$$

$$G(k) = \sum_{j=1}^k \gamma_j G(k-j) \quad k = 1, 2, \dots \quad (2)$$

and

$$\gamma_k = \begin{cases} \frac{1 + \beta A^{*(1)}(\beta)}{A^*(\beta)} & \text{for } k = 1 \\ -\frac{(-\beta)^k A^{*(k)}(\beta)}{k! A^*(\beta)} & \text{for } k = 2, 3, \dots \end{cases} \quad (3)$$

The mean waiting time given the customer has to wait is

$$E(W|W > 0) = \left(\sum_{j=0}^{s-1} G(j) - sA^*(\beta) \right) (\beta(G(s-1) - A^*(\beta)))^{-1} \quad (4)$$

Reference

Per Hokstad. *The G/M/m queue with finite waiting room*. Journal of Applied Probability, Vol. 12, p 779-792, 1975