Performability Modeling for Multi-layered Service Systems

C. M. Woodside
Dept. of Systems and Computer Engineering
Carleton University
email:cmw@sce.carleton.ca

Multi-layered Service Systems

Distributed software is usually structured in layers, with some kind of operational control or user interface tasks as the topmost layer, making requests to various layers of servers. Client Server systems and ODP (Open Distributed Processing) systems such as DCE, ANSA and CORBA are structured this way, as are distributed transaction processing systems. Performance and dependability are the two strong motivators for introducing distributed processing. A special form of layered performance model called Layered Queueing or Rendezvous Nets has been developed to model such systems, to provide a model formulation which is close to the software architecture (e.g. [1, 2, 3]). This note describes how dependability analysis could be added to the layered model, in a way which would make the dependability-related parameters particularly easy to specify. The result would be a straightforward and powerful performability analysis for distributed software systems. Performability concepts and tools are described in many papers, for example a recent survey by Haverkort and Niemegeers [4].

Use of MSS Models for Analyzing Performability

A typical MSS model is shown in Figure 1, with User tasks at the top and three layers of server tasks below. An arrow represents a request-reply interaction, such as an RPC (remote procedure call). The bottom layer might represent data services, and the intermediate layers provide different calculations and format changes (called "business applications" in three-tier client-server systems). System nodes (processors) are represented by ovals. Figure 1 describes both the architecture of the software, and its configuration. The performance model parameters attached to such a model are the mean total demand for execution on the node, and the mean number of requests for each interaction. The model also may have additional detail not shown in Figure 1, such as multi-threaded server tasks, multiple classes of service by a task, and asynchronous messages (with no reply).

For performability analysis this model allows us to identify failure parameters directly where they affect tasks (a crashed server), nodes (a failed computer or disk) and (with a little extension) links which convey interactions. Thus there is a minimum of translation needed by the user, to attach failure and repair rates to the relevant items.

Expressing the Operational Transitions and Rewards

A dependable distributed system employs alternative servers and routing of requests to mask failures. Indeed the widespread interest in CORBA is partly due to its capability to reconfigure a system by re-directing the service requests, either for better performance or to counter a failure. The MSS model has to be modified to express the strategy to be used in case of failure, and to generate the modified configuration of a system that is partly failed but is still running at a reduced functionality. In the modified configuration the target of some interactions has been altered to use stand-by servers.

A conventional approach is to specify back-up or alternative targets for requests, to be used when the primary target fails. There may be multiple alternatives with an order of preference. Figure 2 shows three alternatives for one interaction, labelled by their preference-order. Only one of these interactions is active at one time, and a full description of a system with its selected alternatives will be called an "Operational Model." A failure or repair changes the "structure state" which may cause the system to switch to a new Operational Model. In an optimized scheme, the choice of Operational Model is optimal for each structure state.

In a partly-failed state there are three ways performance is affected:

- the performance of different kinds of responses is degraded (e.g. response type R_i has response time T_i, which might be increased);
- the degraded system is not stable for some response types (it cannot keep up with the arrival rate, so $T_i \rightarrow \infty$);
- some interaction which is required to complete the response is unavailable (there is no remaining backup), so the response may fail.

The MSS model expresses situations when a given interaction is required in a response with a certain *probability*, so in the third case we can evaluate the probability F_i that response R_i fails.

The reward structure for an MSS may blend the different measures into a single performability figure, or keep them separate.

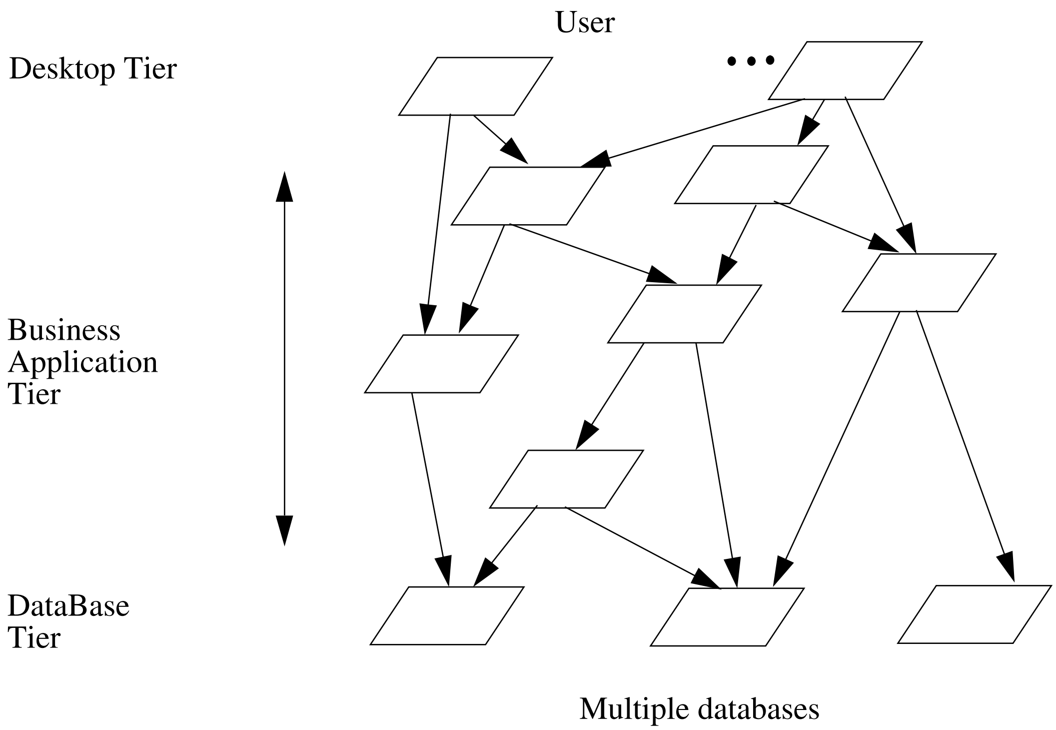

Figure 1. A Multilayered Service System (MSS) for Three-Tier Client-Server Processing

Generating the Structure State Analysis

In an MSS model there is a set of entities C_i which may fail: nodes, links, and tasks. Each entity C_i has its own structure state S_i, and the overall system state is $S = (S_1, S_2, ..., S_N)$. For each state S there is an operational model M(S), as just described.

In many cases the failures will be independent and we will make that assumption here, although it is straightforward to consider dependent failures. In the independent case each S_i is governed by a separate Markov chain with an operational state (S_i = OK), a failed state (S_i = F), and rates of failure and repair, which give its probability p (S_i = F) of being in a failed state. Dependencies in the distributed system still exist through the chains of interactions, such that a failure of a server may cause an entire distributed response to fail, but they are expressed in the configuration rather than in the structure state analysis. This makes the performability calculations simpler (and re-uses the configuration analysis which will be done for system operation in any case).

In some cases there are physical dependencies, for instance a point-to-point link is controlled by its two nodes and if one of the nodes goes down so does the link. Then the dependent node's states should be modelled in a single Markov Chain, in the usual way.

Solution Method

The analysis for a given reward function now has three steps:

1. Find the Markov Chain solutions that give the reachable set of failure states S, and the probability $p(S)$ of each,

2. For each S generate the operational model M(S) of the MSS, and the set of response failure probabilities F_i for M(S),

3. Apply layered queueing analysis or another performance model to obtain the performance reward rates for

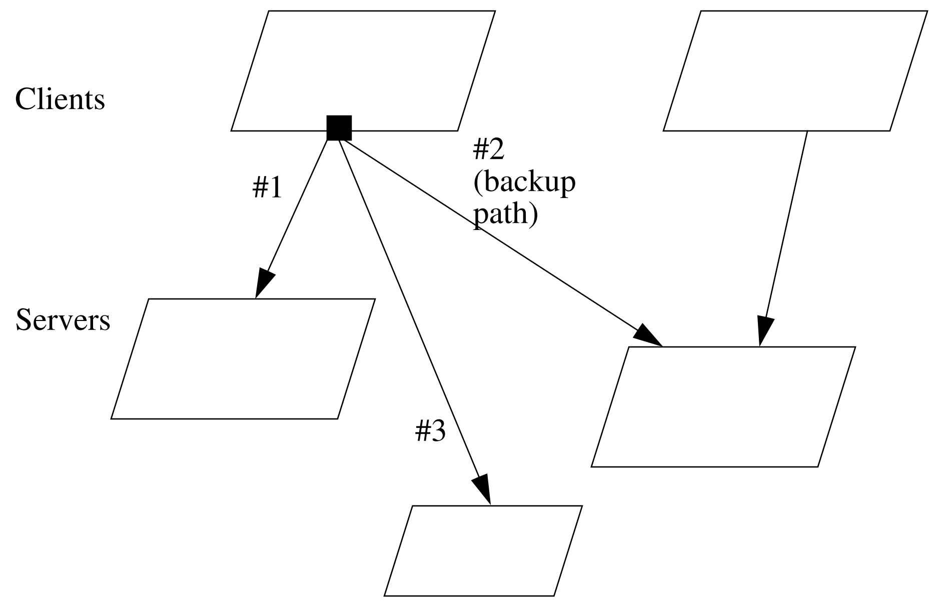

Figure 2. Alternative Targets for a Service. (If #1 fails then #2 is used.

each operational configuration,

4. Combine the reward rates for response failures and performance in the usual way, to construct the desired measures. For example the average response time for R_i, in the absence of response failures, would be

$$T_i = \sum_S p(S) T_i(S)$$

Bibliography

[1] C.M. Woodside}, "Throughput Calculation for Basic Stochastic Rendezvous Networks", *Performance Evaluation*, vol. 9 no. 2, pp 143--160, April 1989.

[2] C.M. Woodside, J.E. Neilson, D.C. Petriu and S. Majumdar, “The Stochastic Rendezvous Network Model for Performance of Synchronous Client-Server-like Distributed Software”, *IEEE Transactions on Computers*, vol. 44, No. 1, pp. 20-34, January 1995.

[3] J.A. Rolia and K.C. Sevcik, “The Method of Layers”, *IEEE Trans. on Software Engineering*, vol. 21, No. 8, pp. 689-700, August 1995.

[4] B.R. Haverkort and I.G. Niemegeers, “Performability Modelling Tools and Techniques”, *Performance Evaluation*, vol. 25, pp. 17-40, 1996.