ON OPTIMIZING SERVICE AVAILABILITY OF AN INTERNET BASED ARCHITECTURE FOR INFRASTRUCTURE PROTECTION

Novella Bartolini and Enrico Tronci

University of Rome "La Sapienza"[1]

Abstract

Modern Infrastructure Protection Systems (IPSs) tend to be geographically distributed as well as Internet based. Geographic distribution has the goal of increasing fault tolerance as well survivability (e.g. against attacks of various kinds). Usage of a public communication infrastructure like Internet has the goal of decreasing costs as well as easying system deployment.

During an emergency, computational as well as communication resources tend to be scarce. When there are no resources available a request is not admitted into the systems. As a result emergency requests may get blocked. More specifically, two service classes must be served, a low priority class for maintenance and monitoring and a high priority class for emergency related interventions. A suitable service policy is necessary to guarantee system responsiveness when emergency events occur while keeping the Quality of Service (QoS) of low priority requests at an acceptable level. Unfortunately the above are conflicting requirements since to guarantee system responsiveness upon an emergency (high priority) event we should always have free resources to use in such a situation.

In this paper we show how the above admission control policy problem can be modeled as a Semi Markov Decision Process (SMDP). By solving such SMDP we compute a policy that finds a tradeoff solution between minimizing the probability that a critical service request is blocked due to resource unavailability and reserving too much resources to critical events resulting in unacceptable performance of low priority requests. For computational and robustness reasons we also propose some heuristics. Finally, we compare the performance of the computed optimal policy and heuristics by means of simulation and probabilistic model checking.

Keywords: Critical Infrastructure, Internet, QoS, Access Policy, Markov Decision Process

1. System architecture

The reference scenario of our work is one with distributed and replicated servers.
A discussion on the possible architectural choices to perform server selection and request redirection is out of the scope of this paper. We refer to [2] for a short survey on this topic. As for Infrastructure Protection Systems (IPSs), we consider an anycast based Content Delivery Network (CDN) architecture [2] in which admission control operations are performed by access routers (see Figure 1) or by application level dispatchers.

[1] Computer Science Department, Via Salaria 113, 00198 Roma, Italy. Email: {novella,tronci}@di.uniroma1.it. Corresponding Author: Enrico Tronci. Tel: +39 06 4991 8361 Fax: +39 06 8541 842

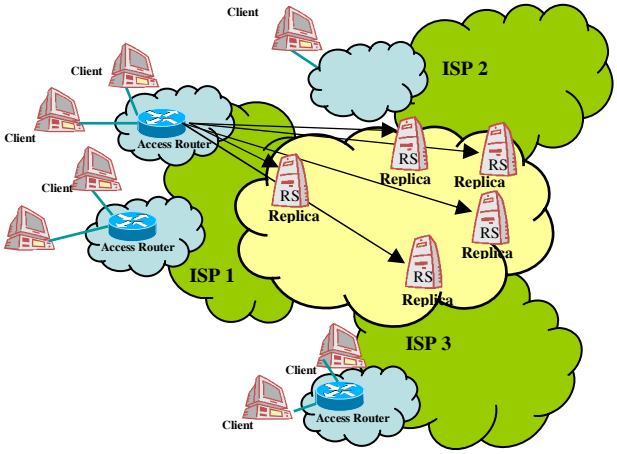

Figure 1. An Internet based architecture for infrastructure protection.

This CDN architecture was inspired by [1] and similar schemes are discussed in [4, 6, 9], just to give some examples. The access points (routers or dispatchers) may collect statistics about replica servers by means of active and passive measurements, or receive status update messages from the available servers, create user and session profiles, perform replica server selection mechanisms and access control to guarantee the required performance to all types of services, with prioritization of sessions and of requests belonging to critical session phases. Through their measurement activity the access points become dynamically aware of the resource availability and traffic behaviour. In this paper we focus on the access control capabilities of these systems.

2. Analytical model

In [5, 12] it is suggested that Markov Modulated Poisson Processes (MMPPs) can be used to approximate or predict the burstiness of the input of an Internet based service. Following [3] we assume exponential arrivals of service requests, while the session duration is modelled by means of an alternation of idle and active phases, following a Markov modulated process of the lifetime of a service session such as in Fig. 2.

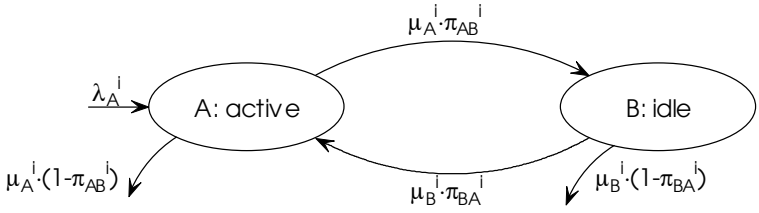

Figure 2. Service session model.

We differentiate between two classes of clients: high priority (e.g. emergency) and low priority (e.g. monitoring), such that the high priority clients, representing critical requests, receive a better service than basic clients in case of overload, (the index i in Figure 2 is introduced to differentiate these two classes).

External traffic on the non-dedicated links between access routers and replica servers has an impact on the server availability. This is modelled by means of the random variable x_{cong}, ($0 <= x_{cong} <= C$) that follows a Markov modulated vacation (or ON/OFF) process.

The controlled dynamic of the described system constitutes a Semi Markov Decision Process (SMDP). We model the state of the process by using a random variable for each phase of each type of service, plus another random variable x_{cong} to represent the congestion level. The state space of the process is

$$\Lambda = \left\{ \mathbf{x} = \left(x_1, x_2, x_3, x_4, x_{cong}\right) : \sum_{i=1}^{4} b_i x_i \le C; \sum_{i=1}^{4} x_i \le C^{ID}; x_{cong} \le C; x_i \ge 0 \right\}, \quad (1)$$

where b_i is the resource consumption of phase i, and there is a limit C^{ID} on the number of sessions that can be tracked at the same time, while the number of available servers is limited to C. The decision space is $A_x = \{a : a_i = 0 \text{ if } x + e_i \notin \Lambda, i = 1, \ldots 4\}$, where e_i is an identity vector, that is a vector of zeroes, except for a one in the i-th position. The indicators a_i denote the admission, with value 1, or the denial of service, with value 0, of class-i new session requests.

Let p^a_{xy} denote the uniformized transition probability from state x to state y if the decision a is taken and $(x, a) \in S$ where S is the set of all feasible pairs of vectors (*state*, *decision*).

To complete the description of the decision process and to be capable of formulating an objective function, we introduce a cost and a profit function: $r_{cost}(\mathbf{s}, \mathbf{a})$ and $r_{profit}(\mathbf{s}, \mathbf{a})$ for each pair $(x, a) \in S$.

If a new session request is rejected, a rejection penalty will be paid, P_L for low priority requests, and $P_H > P_L$, for high priority requests. Phase transitions are not subject to the admission control, and all subsequent phases of an admitted session are also admitted provided that enough non-congested servers are available. If there are no available servers to complete a session the system will incur a disruption penalty D_H, in case of high priority session disruption, and $D_L < D_H$, in case of low priority session disruption.
Disruption penalties are usually higher than rejection penalties.
Profits for successful service completion are also introduced and denoted by V_L and V_H for low and high priority requests respectively.

The objective function we formulated is the average expected reward per unit of time $\sum_{(\mathbf{s},\mathbf{a}) \in S} \left[r_{profit}(\mathbf{s}, \mathbf{a}) - r_{cost}(\mathbf{s}, \mathbf{a}) \right] \cdot x_{\mathbf{sa}}$, where x_{sa} is the probability for the system to be in state s and at the same time to take decision a

The Linear Programming (LP) formulation associated with our decision process for the maximization of the average reward is:

$$\max \sum_{(\mathbf{s},\mathbf{a})\in S} \left[\; r_{profit}(\mathbf{s},\mathbf{a}) - r_{cost}(\mathbf{s},\mathbf{a}) \; \right] \cdot x_{\mathbf{sa}}$$
$$x_{\mathbf{sa}} \geq 0 \qquad (\mathbf{s},\mathbf{a}) \in S$$
$$\sum_{(\mathbf{s},\mathbf{a})\in S} x_{\mathbf{sa}} = 1 \qquad .$$
$$\sum_{\mathbf{a} \in A_{\mathbf{j}}} x_{\mathbf{ja}} = \sum_{(\mathbf{s},\mathbf{a})\in S} \tilde{p}^{\mathbf{a}}_{\mathbf{sj}} x_{\mathbf{sa}} \qquad \mathbf{j} \in \Lambda$$

(2)

3. Comparisons among the optimal admission policy and heuristics

The purpose of this optimization analysis is to obtain clues for the formulation of possible heuristics to be adopted in realistic scenarios where the analytic methodology cannot scale. By iteratively solving problem (2) we saw that in the most typical cases the optimal policy shows a double threshold behavior when deciding which request to accept: a first threshold is in terms of available servers, and the second one is in terms of process identifiers available at the dispatcher level.
For this reason we consider the following heuristic (HEU) that mimics the behaviour of the optimal policy (OPT). The HEU policy reserves $K_{reserved\_servers}$ units of server capacity and $K_{reserved\_ID}$ session identifiers to the high priority stream of requests.
We refer to x_{think} as to the number of ongoing session in the idle phase, therefore $x_{think} = x_1 + x_3$, and to x_{busy} as to the number of ongoing session in the active phase, that is $x_{busy} = x_2 + x_4$.

We define the following threshold values: $T^{servers} = C - K_{reserved\_servers}$ and $T^{ID} = C - K_{reserved\_ID}$.

The HEU policy can be formulated as follows:

- If $x_{busy} < \min\{ T^{ID} - x_{think} ; T^{servers} - x_{cong} \}$ take decision $(a_1,a_3)=(1,1)$, i.e. give service to both streams of requests.
- If $\min\{ T^{ID} - x_{think} ; T^{servers} - x_{cong} \} \leq x_{busy} < \min\{ C^{ID} - x_{think} ; C - x_{cong} \}$ take decision $(a_1,a_3)=(0,1)$, i.e. give service only to high priority requests.
- If $x_{busy} \geq \min\{ C^{ID} - x_{think} ; C - x_{cong} \}$ take decision $(a_1,a_3)=(0,0)$, i.e. no new session can be admitted, neither from the low priority, nor from the high priority stream, due to lack of resources.

The implementation of such a regular policy (HEU) is inexpensive and can be easily implemented on access controllers.

By means of simulations we analyzed the effects of the optimal policy and of our heuristics with different choices of the threshold parameters, showing that a simple and regular policy such as HEU can have a performance close to the optimal in most scenarios.

4. Model checking

By using uniformization techniques and by resorting to discrete time we can study the SMDP process described in Section 2 through its embedded Markov Chain (MC). In particular it is possible to use a simulator to estimate the probability of reaching an undesired state. In the following discussion s_{bad} denotes an undesired state. For example the one in which an emergency request has been blocked.

Unfortunately, the smaller the probability of reaching a given undesired state s, the less reliable is the estimate obtained by simulations, since there will be very few computation traces leading to s. This makes life quite hard for a simulator. Such low probability computation traces can be analyzed with a probabilistic model checker e.g. as PRISM [7, 8] or FHP-Murphi [10, 11] In our case we use FHP-Murphi, a probabilistic model checker that can handle real numbers.

We proceed as follows. First of all we choose a sampling time T. From this and the SMDP parameters in Section 3, we can compute the transition probability for the MC in Figure 3. Running FHP-Murphi on the MC of Figure 3 confirms the effectiveness of the heuristic devised in Section 3 for the admission control policy.

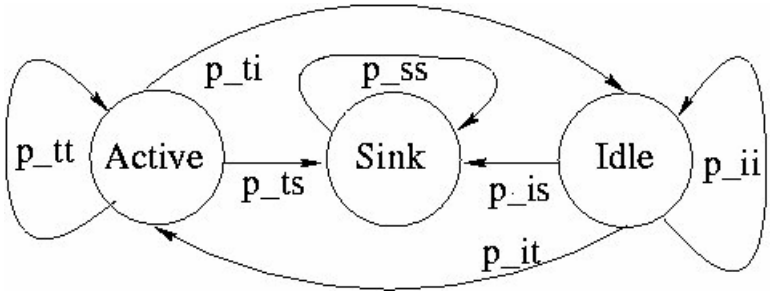

Figure 3. Discrete Time Markov Chain for the SMDP in Section 2

Note that, since FHP-Murphi works with closed systems, in Figure 3 we have added a Sink with respect to the model in Figure 2. Such sink state models users leaving the system (i.e. releasing system resources).

5. Experimental Results

In this section we show the experimental results we obtained by using the simulator and the model checker. Our idea is to exploit the simulator efficiency in order to estimate typical system behaviour on fairly long time horizons (Section 5.1). This would not be possible with a probabilistic model checker because of state explosion. On the other hand a simulator cannot give us much information about (undesired) rare events, namely dropping of a high probability request. On a short time horizon we compute such probabilities by using a probabilistic model checker (Section 5.2).

5.1. Simulation results

In this section we describe the experimental results obtained by means of a simulator based on OPNET [13] using synthetic traffic generators that follow the session model introduced in section 2, with arrival rates of different orders of magnitude to keep into account the rarity of emergency requests.
By means of simulations we analyze the effects of the heuristic (**HEU**) introduced in the previous section 3. We provide performance comparisons among the optimal policy (**OPT**) and the heuristic with different choices of the threshold parameters. Experimentations showed that the best threshold choice depends on many factors, among which the most important are costs and rewards, session arrival rates and average lifetime of successfully completed sessions.
A trivial policy, consisting in doing nothing to improve performance, will be named **noAC** and will be used with OPT as a benchmark for comparisons. With the noAC policy both streams of session activation requests are treated alike and no discrimination is done between service classes.

We consider a scenario with C=8 servers, C_{ID}=10 available identifiers, and where the traffic parameters are: low priority arrival rate λ_L=2 sec^{-1}, high priority arrival rate λ_H=0.0001 sec^{-1}, active phase rate for both high priority and low priority $\mu_L^A=\mu_H^A$ =10 sec^{-1}, idle phase rate for both high priority and low priority $\mu_L^B=\mu_H^B$ =0.05 sec^{-1}, phase transition probabilities $\pi_L^{AB}= \pi_H^{AB}$ =0.9, $\pi_L^{BA}= \pi_H^{BA}$ =0.4. In the considered scenario, the congestion arrival rate per single server is μ_{AP}=0.001 sec^{-1} and congestion departure rate λ_{AP}=0.01 sec^{-1}.
The cost setting is: dwell cost in a busy and congested state P_{BC}=100 sec^{-1}, low priority service denial P_L=10, low priority service disruption D_L=30, high priority service denial $P_H=2\times10^5$, high priority service disruption $D_H=2.1\times10^5$, while rewards are V_L=30, $V_H=10^5$.

Figure 4 shows how in the described scenario, the introduction of an admission control policy brings an improvement in terms of average reward. The heuristic threshold setting that better approximates the optimal policy is to reserve T^{ID}=9 identifiers and $T^{servers}$=6 units of server capacity.

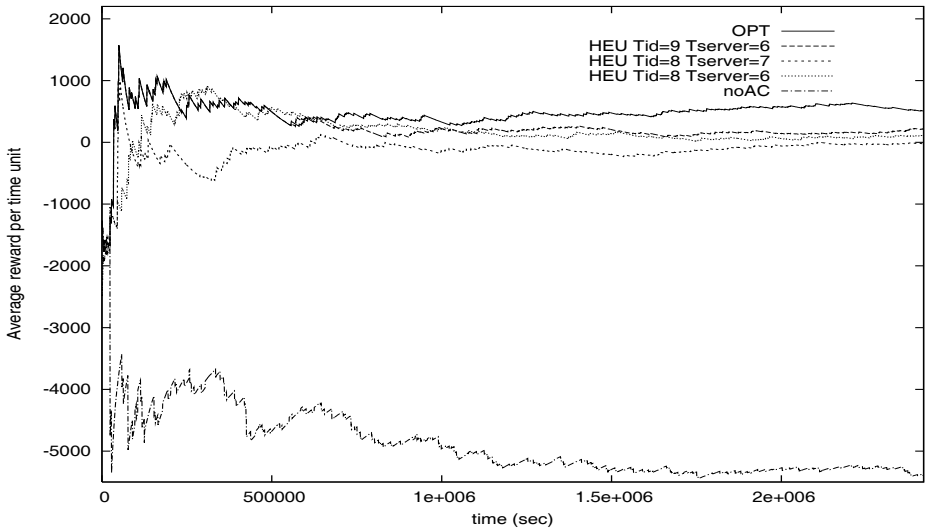

Figure 4 – Average Reward Function

The improvement in terms of rewards comes from a lower blocking probability of high priority requests, as can be seen in Fig. 5.

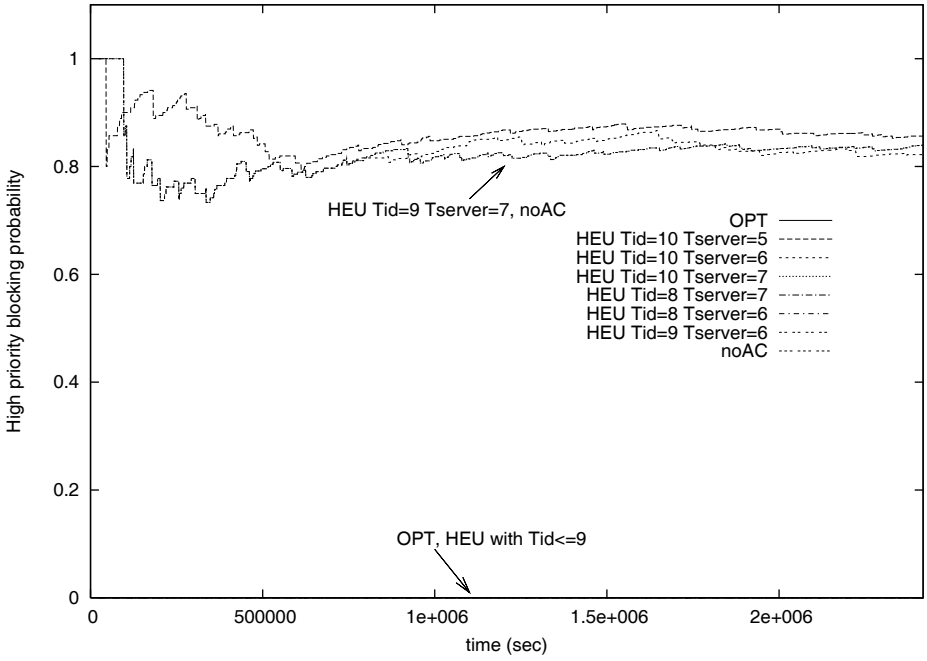

Figure 5 – Critical event (high priority) blocking probability

This improvement comes at the expense of an increase in the blocking probability of low priority requests as can be seen in figure 6. This figure shows that the OPT policy has the worst performance in terms of blocking probability of low priority requests. The smoothness of the graph of figure 6 is due to the more stable and frequent arrival process of low priority requests.

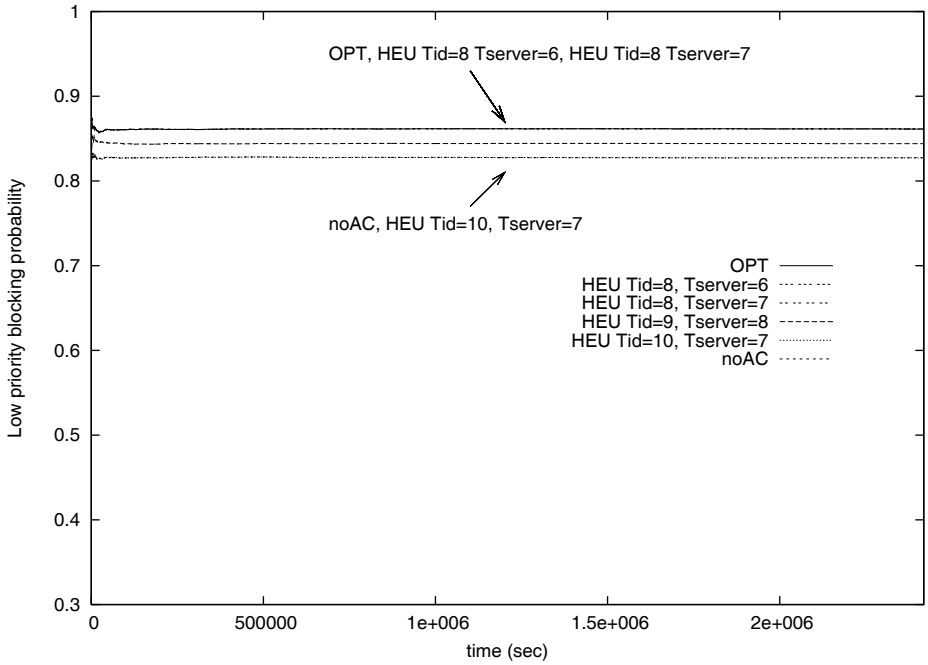

Figure 6 – Low priority blocking probability

bbb

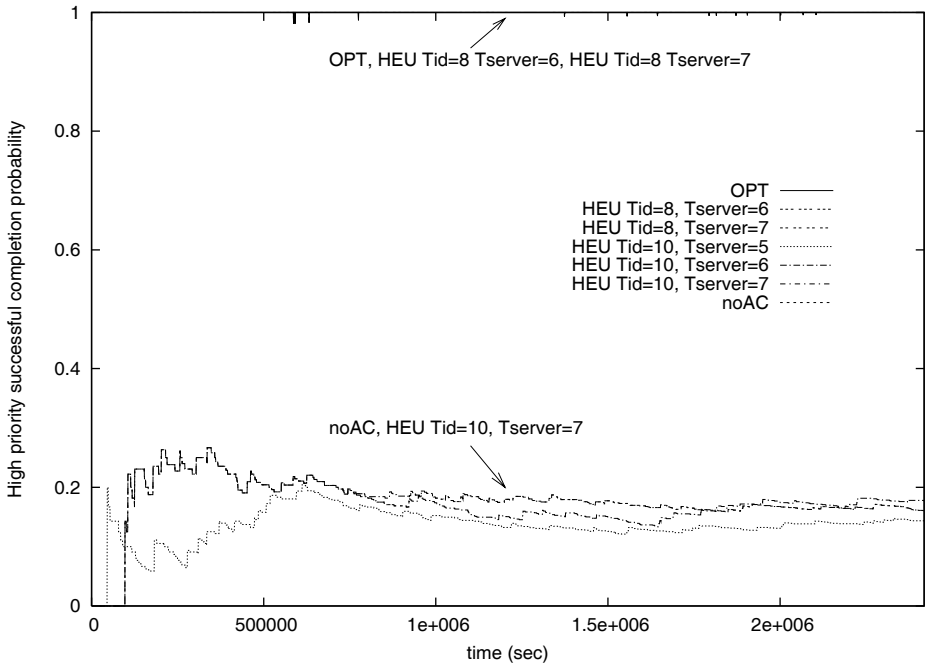

Figura 7 - Critical requests successful completion probability

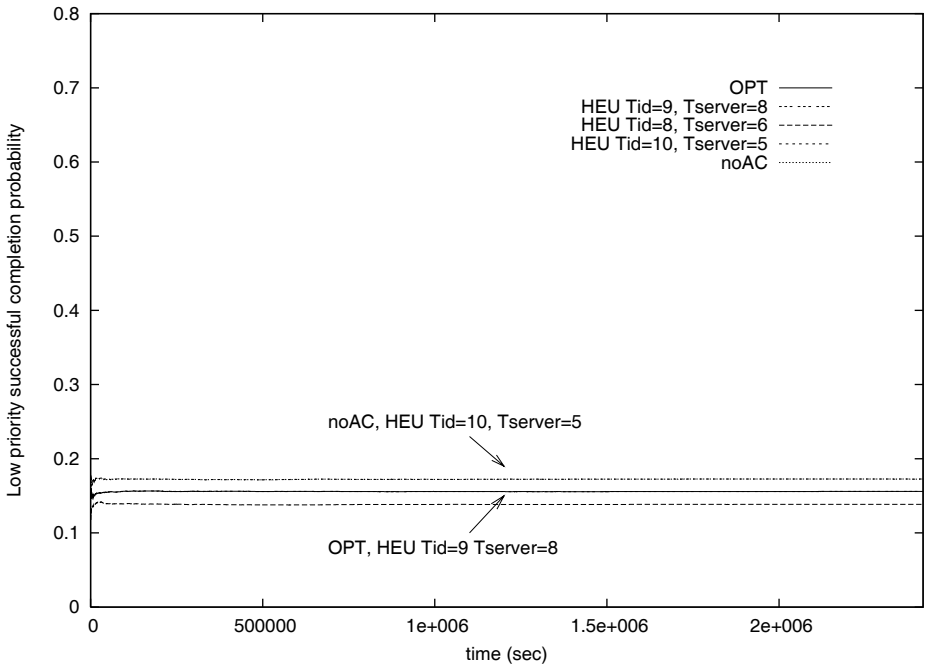

Figure 8 – Low priority successful completion probability

5.2. Model checking results

Given a Markov chain, a probabilistic model checker can compute in an exact way the probability of reaching an undesired state. In the following our *undesired state* is the one in which a high priority request is dropped (Section 5.1). Of course we would like the probability of reaching such a state (i.e. of dropping a high priority request) to be low. This is exactly what we are going to compute with the probabilistic model checker FHP-Murphi.

More specifically we compute the probability that in at most 25 events at least a high request is dropped. An event for us is any system transition that changes the state of the systems, thuis a *nop* is not an event.

Our system parameters are, of course, exactly the same one used by the simulator and the model checker system model (a Markov chain) has been obtained from the simulator model as described in Section 4. As an admission policy we use the heuristics HEU described in Section 3. Let

- AL be the number of low priority processes in the active state;
- AH be the number of high low priority processes in the active state
- IL be the number of low priority processes in the idle state;
- IH be the number of high priority processes in the idle state;
- X_cong be the number of congested servers.

AL	AH	IL	IH	X_cong	Drop Prob	CPU Time (s)
0	0	1	0	0	0	18.29
0	0	6	4	6	5.442186651e-07	7543.78
0	0	8	0	1	3.782883752e-09	1448.2
0	0	8	0	2	3.782883752e-09	1941.54
0	1	0	0	0	1.027936619e-10	143.09
1	0	6	0	1	3.885609142e-10	1464.49
1	0	7	0	1	3.545190093e-09	1765.86
1	0	7	0	2	3.545190094e-09	2176.02
2	0	6	0	1	3.431086205e-09	1969.58
2	0	6	0	2	3.431086444e-09	2498.04

Table 1. Experiments run on a Linux PC with a 3GHZ Pentium 4 with 512MB RAM.

Table 1 shows the results we obtained using FHP-Murphi to compute the dropping probabilities for the most frequent system states, i.e. the states the occur more often in a *long enough* time window. Column “Drop Prob” gives the dropping probability for the system states defined in columns AL, AH, IL, IH, X_cong. Column “CPU Time” gives the model checke computation time in seconds.

Note that such probabilities are so small that it is not possible to estimate them by using a simulator. On the other hand the curves obtained in Section 5.1 by using the simulator cannot be obtained by using a probabilistic model checker because of state explosion.

5. Conclusions

During an emergency a suitable service policy is necessary to guarantee system responsiveness when emergency events occur while keeping the Quality of Service (QoS) of low priority requests at an acceptable level. Unfortunately the above are conflicting requirements since to guarantee system responsiveness upon an emergency (high priority) event we should always have free resources to use in such a situation.

In this paper we have shown how the above admission control policy problem can be modeled as a Semi Markov Decision Process (SMDP). Solving such SMDP we can compute a policy that minimizes the probability that a service request is blocked due to resource unavailability. Moreover for computational and robustness reasons we have also proposed some heuristics (Section 3).

We have compared the performance of the computed optimal policy and heuristics by means of simulation and probabilistic model checking. This twofold approach has allowed us exploit simulation computational efficiency to study typical system behaviour on long time horizons (Section 5.1) as well as the model checking accuracy to compute probabilities for rare undesired events, namely dropping of a high probability request (Section 5.2).

References

1. G. Agarwal, R. Shah, and J. Walrand. Content distribution architecture using network layer anycast. Proc. of IEEE Workshop on Internet Applications, 2001.
2. N. Bartolini, E. Casalicchio. A walk through content delivery networks. Lecture Notes on Computer Science, 2965, 2004.
3. N. Bartolini, E. Casalicchio, and I. Chlamtac. Session based access control in content delivery networks in presence of congestion. Proc. of IEEE QShine 2004, Dallas, TX, 2004.
4. M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. Scalable application-level anycast for highly dynamic groups. Networked Group Communications, 2003.
5. A. Iyengar, M. Squillante, and L. Zhang. Analysis and characterization of largescale web server access patterns and performance. Proceedings of World Wide Web, 1999.
6. D. Katabi and J. Wroclawski. A framework for scalable global ip-anycast (gia). In Proceedings of SigCom, 2000.
7. M. Kwiatkowska, G. Norman, and D. Parker. Prism: Probabilistic symbolic model checker. In Proc. TOOLS 2002, volume 2324. LNCS, Springer Verlag, April 2002.
8. M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with prism: A hybrid approach. In Proc. TACAS'02, volume 2280. LNCS, Springer Verlag, April 2002.
9. C. Partridge, T. Mendez, and W. Milliken. Host anycasting service. http://rfc.sunsite.dk/rfc/rfc1546.html.
10. G. Della Penna, B. Intrigila, I. Melatti, E. Tronci, and M. V. Zilli. Finite horizon analysis of markov chains with the murphi verifier. In IFIP WG 10.5 Advanced Research Working Conference on: Correct Hardware Design and Verification Methods (CHARME), L'Aquila, Italy, Oct 2003. LNCS, Springer.
11. G. Della Penna, B. Intrigila, E. Tronci, and M. Venturini Zilli. Bounded probabilistic model checking with the mur' verifier. In Proc. of 5th International Conference on:

Formal Methods in Computer Aided Veri_cation" (FMCAD), Nov. 2004, LNCS, Springer
12. T. Yoshihara, S. Kasahara, and Y. Takahashi. Practical time-scale fitting of selfsimilar traffic with markov-modulated poisson process. Proc. of the 6th Int. Conf. on Telecomm. Systems, 2001.

Author Biographies

Novella Bartolini graduated with honors in 1997 and received her PhD in computer engineering in 2001 from the University of Rome "Tor Vergata". She currently is assistant professor at the Computer Science department of the University of Rome "La Sapienza". In 1999-2000 she has been visiting scientist at the Erik Johnsson School of Telecommunications at the University of Texas at Dallas. In 1997 she has been a researcher of the Fondazione Ugo Bordoni. She has been a Program Committee and Organizing committee member and Program Chair of several international conferences. She is member of the editorial board of the international journal Elsevier Computer Networks. Her current research interests include Content Delivery Networks, Web Performance and Wireless Networks.

Enrico Tronci received his Master degree in 1987 in Electrical Engineering from the University of Rome "La Sapienza". In 1991 Enrico Tronci received his Ph.D degree from Carnegie Mellon University, Pittsburgh, USA. From 1992 to 1993 he has been a Post-Doct at LIP (Laboratoire pour l'Informatique du Parallelisme) at the ENS (Ecole Normal Superior) of Lyon (France). Presently he is associate professor at the Computer Science department of the University of Rome "La Sapienza".
His research interests comprise: Formal Methods, Automatic Verification Algorithms, Model Checking, Hybrid Systems. Tronci authored many scientific papers both on international journals and on international conferences. He has been Conference Chair for the CHARME (Correct Hardware Design and Verication methods) conference in 2003 and in the program committee of the conferences: FMCAD (Formal Methods in Computer Aided Design) 2004, CHARME 2005. Enrico Tronci participated to many research projects sponsored from the European Community, CNR, ENEA and MURST.