Passage-End Analysis

Allan Clark, Adam Duguid, and Stephen Gilmore

The University of Edinburgh, Scotland

Abstract. Passage-end calculations are a new style of passage measurement for eXtended Stochastic Probes (XSP) [1] which add the ability to split the analysis into several cases depending on conditions which hold at the end of a passage. This makes it possible to separate successful responses to a request from negative responses, timeouts or other failures. This allows the expression of service level agreements such as: "At least 90 percent of all requests receive a response within 10 seconds and at least 60 percent of all such requests are successful."

1 Introduction

Many systems which are analysed for response-time profiles have more than one way in which the passage in question may terminate. Commonly a request from a client may end in a successful completion such that the client received the desired service but may also end in failure due to a timeout or rejection. There may also be two or more ways in which the request can be satisfied, for example a data retrieval request may be serviced by accessing the cache or the disk. In these situations we wish to analyse the passage-time profiles for the separate cases of success or failure, and cache or disk retrieval. It may be that the general response-time analysis indicates acceptable performance but that successful requests are serviced too slowly.

A stochastic probe [2,3] is a component added to a model in order to make reasoning about the model more convenient. In the context of PEPA [4] a stochastic probe is a single sequential component which observes the activities of a PEPA model.

For example we may analyse the response-time of a service as observed by a single client. Once this is known, in order to provide more information on how this may be improved, we then analyse the response-time of all requests which are made when the service is definitely not broken. We would expect this to be better than the more general case of all requests. Conversely we may also analyse the response-time for all requests made when the service certainly is broken and expect this to be worse than for the general case. From this we may determine whether, in order to improve response-time, it is better to repair the server more quickly or make the server more reliable such that it breaks less often.

Although with eXtended Stochastic Probes splitting the measurement of a passage with respect to the starting conditions is convenient it is not clear how one may split a passage based on how the passage completes — or based on some event during the passage. The novelty in this paper is the use of several

J.T. Bradley (Ed.): EPEW 2009, LNCS 5652, pp. 110–115, 2009.

absorbing states, one for each target event. For a passage-end calculation the user must specify a list of target actions. These may be actions performed by the model itself or communication signals sent by user defined probes. During transient analysis each target state is modified by adding a transition to a distinct absorbing state based on the causal event.

As an example, consider analysing the response-time of a service which begins with a request but may be concluded with either a cached response or a networked response. It is not possible to simply measure these two passages with separate runs using a probe such as: *request*:start,*cached*:stop. The reason is that this will compute the probability of completing the passage at (or within) time t via a cached response plus the probability of completing the passage via a networked response and then restarting and completing the passage via the cached response all at (or within) time t. Indeed you may complete the passage twice, three times or any number of times via the networked response before finally completing via the cached response.

Using the same algorithm we can calculate the raw pdf and cdf of the passage from the request to the cached response. In this case the cdf will not tend to one but to the fraction of requests which are ultimately serviced by the cache. Similarly for the area under the pdf and for both functions of the request to networked passage. We can normalise these graphs based on the probability of completing at or within the given time t via any target event.

We can instead normalise the raw pdf and cdf by dividing through the probability of completing at (or within) time t by the percentage of all requests which are ultimately serviced by the target event in question. We can know this percentage by calculating enough hops such that sufficiently close to all of the probability mass at π^N is in one of the absorbing states. We may then take the probability of being in the appropriate absorbing state at π^N.

Hence using a passage-end calculation it is possible to calculate:

- The probability of completing a passage by a cached response at or within a given time.
- The probability that, assuming the passage completes at or within a given time in some way, that it does so via the cached response. This answers such questions as: "What percentage of responses received within 5 seconds are received via the cache/network?"
- The cdf and pdf profiles for all requests which are serviced by the cache.

In the above "cached" may be replaced by "networked" or any other target event of the given passage, including probe communication signals. The second kind of question is helpful in evaluating some SLAs (Service-Level Agreements), particularly for services which may end in success, failure or cancellation. For example the SLA may say that ninety percent of requests receive a response within 10 seconds. We may analyse the model and find that this is indeed the case but a passage-end analysis reveals that eighty-nine percent of such requests are rejection/failure responses. Hence we may wish to modify our SLA to state that ninety percent of all successful requests receive a response within 10 seconds.

2 Example of Passage-End Analysis

We consider an example of passage-end analysis applied to emergency response service quality [5]. The scenario centres on the function of an automatic crash response subscription service such as OnStar [6] from General Motors. These utilise multiple built-in sensors which capture critical details in the event of a car crash. The built-in communications module automatically contacts the OnStar service and relays the information obtained from the sensors, including location information from the on-board GPS. The service attempts to contact the driver to ask if they need help. If the service cannot contact the driver then they send medical assistance to the car's location.

The PEPA model in Figure 1 represents the scenario where the protocol for attempting to call the driver is to try three times. After three unsuccessful attempts to contact the driver the system assumes that the driver is unavailable and at this point must decide on the basis of the car telemetry whether to dispatch an ambulance. We are interested then in the time between the airbag deploying and either a successful or failed attempt to contact the driver. The start event of our passage is then the *airbag* activity and the two ways in which a passage may terminate is with an *answer* or a $timeout_3$ activity. Note that the timeout activities are distinguished such that only the third timeout will end the passage.

$$
\begin{aligned}
Car_1 &\stackrel{def}{=} (airbag, r_1).Car_2 \\
Car_2 &\stackrel{def}{=} (reportToService, r_2).Car_3 \\
Car_3 &\stackrel{def}{=} (processReport, r_3).Car_4 \\
Car_4 &\stackrel{def}{=} (callDriver, r_4).Car_5 \\
Car_5 &\stackrel{def}{=} (timeout_1, p * r_5).Car_6 \\
&\quad + (answer, (1-p) * r_5).OK \\
Car_6 &\stackrel{def}{=} (timeout_2, p * r_5).Car_7 \\
&\quad + (answer, (1-p) * r_5).OK \\
Car_7 &\stackrel{def}{=} (timeout_3, p * r_5).Rescue \\
&\quad + (answer, (1-p) * r_5).OK \\
OK &\stackrel{def}{=} (finish, r_1).Car_1 \\
Rescue &\stackrel{def}{=} (rescue, r_6).Car_1
\end{aligned}
$$

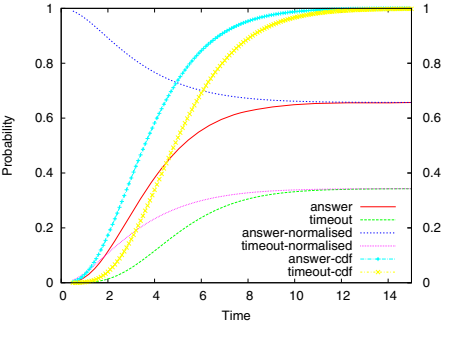

Fig. 1. The model of the driver contact protocol within the airbag crash assistance case scenario. The initial state of this process is Car_1. The graph shows results of passage-end analysis of driver contact retries.

The results from our passage-end analysis of the model are shown in the graph in Figure 1. The first two lines to consider are the 'answer' and 'timeout' lines, these plot the unmodified cumulative distribution functions of completing the passage via the driver answering the phone or a third timeout occurring respectively. Note that neither of these two lines tend to one as is usual for a cdf of a passage. This is because the passage may never end in the prescribed

ways. By looking at the long-term probabilities, or the limits of these two lines (which sum together to one) we can say what percentage of airbag initiations end in the driver answering the phone (and conversely a third timeout occurring). From these two lines we can also answer the question: "What is the probability that the driver answers the phone a given time after the airbag was deployed, regardless of whether the driver is capable of answering at any time".

The next two lines to consider are the above cdf functions normalised by dividing the probability (in each case) at each time by the probability of completing the passage within the given time in any way. These lines tend to the same value in the limit as before because eventually there is a probability of very close to one that we will have completed the passage in some way within that time. These lines allow us to answer the question: "If the passage is completed within a given time t, what is the probability that the passage was completed via the driver answering the phone" (or conversely by a third timeout occurring).

The final two lines to consider are the 'answer-cdf' and 'timeout-cdf'. Here we have normalised the original two lines by dividing the value at each time by the percentage of all requests which are eventually serviced in the given way. By doing this we achieve the cumulative distribution function of all requests initiated by an airbag deployment which then result in the driver answering the phone or the call timing out for the third time respectively. This can then answer our question: "What percentage of airbag deployments whose driver cannot be contacted are serviced within a given amount of time". This figure is often more interesting than just the question "What percentage of airbag deployments are resolved within a given time in some way" since if the driver is unhurt the response-time is of less importance.

3 Implementation

Passage-end analysis for XSP is fully implemented in the International PEPA Compiler (IPC), a formal analysis tool for steady-state and transient evaluation of PEPA models. The IPC compiler is part of the `ipclib` [7] suite, a collection of tools for the specification and evaluation of complex performance measures over Markovian process algebra models. The `ipclib` suite is an extension of the IPC tool previously used for computing response-time quantiles from PEPA models [8,9]. The IPC tool has been applied to a number of modelling problems such as performance of personal-area networks [10] and compiler optimisations [11]. Although we have concentrated here mostly on passage-time computation, IPC also supports the computation of steady-state, transient and counting measures as described in [12].

4 Conclusions

We have modified passage-time analysis to allow for distinct passage results. We have done so within the framework of eXtended Stochastic Probes to ensure that our passage-end queries remain robust over model modifications and do not

require that the modeller modify their original model. The set of queries which can be specified with XSP is therefore extended.

Another bonus which we obtain almost for free is the ability to analyse passages which may never complete at all. This only works for passages in which there is only one source state, because if the model deadlocks we cannot analyse the embedded Markov chain to obtain the distribution of probability to the source states at the beginning of the passage. This allows the modeller to provide a concrete answer to the question: "How long should I wait for my response?" because we can now say that a given percentage of all requests which ultimately are successfully serviced are serviced within 10 seconds, hence if you have waited longer than 10 seconds it is likely you will never receive a response and hence can cancel the request yourself.

Acknowledgements

The authors are supported by the EU FET-IST Global Computing 2 project SENSORIA ("Software Engineering for Service-Oriented Overlay Computers" (IST-3-016004-IP-09)).

References

1. Clark, A., Gilmore, S.: State-aware performance analysis with eXtended Stochastic Probes. In: Thomas, N., Juiz, C. (eds.) EPEW 2008. LNCS, vol. 5261, pp. 125–140. Springer, Heidelberg (2008)
2. Argent-Katwala, A., Bradley, J., Dingle, N.: Expressing performance requirements using regular expressions to specify stochastic probes over process algebra models. In: Proceedings of the Fourth International Workshop on Software and Performance, Redwood Shores, California, USA, pp. 49–58. ACM Press, New York (2004)
3. Argent-Katwala, A., Bradley, J., Clark, A., Gilmore, S.: Location-aware quality of service measurements for service-level agreements. In: Barthe, G., Fournet, C. (eds.) TGC 2007 and FODO 2008. LNCS, vol. 4912, pp. 222–239. Springer, Heidelberg (2008)
4. Hillston, J.: A Compositional Approach to Performance Modelling. Cambridge University Press, Cambridge (1996)
5. Clark, A., Gilmore, S.: Evaluating quality of service for service level agreements. In: Brim, L., Leucker, M. (eds.) Proceedings of the 11th International Workshop on Formal Methods for Industrial Critical Systems, Bonn, Germany, August 2006, pp. 172–185 (2006)
6. OnStar: OnStar by General Motors, car safety device and vehicle security system (April 2009), `http://www.onstar.com/`
7. Clark, A.: The ipclib PEPA Library. In: Harchol-Balter, M., Kwiatkowska, M., Telek, M. (eds.) Proceedings of the 4th International Conference on the Quantitative Evaluation of SysTems (QEST), pp. 55–56. IEEE, Los Alamitos (2007)
8. Bradley, J., Dingle, N., Gilmore, S., Knottenbelt, W.: Extracting passage times from PEPA models with the HYDRA tool: A case study. In: Jarvis, S. (ed.) Proceedings of the Nineteenth annual UK Performance Engineering Workshop, University of Warwick, July 2003, pp. 79–90 (2003)

9. Bradley, J., Dingle, N., Gilmore, S., Knottenbelt, W.: Derivation of passage-time densities in PEPA models using IPC: The Imperial PEPA Compiler. In: Kotsis, G. (ed.) Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems, University of Central Florida, pp. 344–351. IEEE Computer Society Press, Los Alamitos (2003)
10. Ding, J., Hillston, J., Laurenson, D.: Performance modelling of content adaptation for a personal distributed environment. Personal Wireless Communication (November 2007)
11. Djoudi, L., Kloul, L.: Assembly code analysis using stochastic process algebra. In: Thomas, N., Juiz, C. (eds.) EPEW 2008. LNCS, vol. 5261, pp. 95–109. Springer, Heidelberg (2008)
12. Argent-Katwala, A., Bradley, J.T.: Functional performance specification with stochastic probes. In: Horváth, A., Telek, M. (eds.) EPEW 2006. LNCS, vol. 4054, pp. 31–46. Springer, Heidelberg (2006)