A Petri Net based approach for modelling multi-layered distributed protocols

Nicole Sergent
Département d'Informatique
Ecole Polytechnique Fédérale de Lausanne
1015 Lausanne (Switzerland)
sergent@lse.epfl.ch

Abstract

This paper presents an original modelling approach which aims to provide probabilistic guarantees on time in asynchronous distributed systems for several important algorithms, like consensus and atomic broadcast. The modelling framework is based on Hierarchical Coloured Timed Petri Nets (HCTPN). Simulation is used not only to find statistical upper bounds on the termination time of these algorithms but also to study the influence of the Fault Detectors parameters and to obtain optimal values for them.

Keywords: Petri nets, simulation, UDP/IP, consensus, atomic broadcast.

1. Introduction

In asynchronous distributed systems message delays and relative processor speeds are unbounded. Since there is no shared memory nor global clock, the communication and the synchronisation between processes on different sites are done by message passing. The distributed system is composed of a finite set of processes completely connected through a set of reliable communication channels. To avoid blocking in case of failures, the asynchronous environment was extended by including the failure detectors concept [1].

Reaching agreement between several computational nodes is an important problem since data are distributed. This implies solving consensus. To achieve fault-tolerance in systems subject to failures, services are replicated on different sites and specific communication protocols are used to maintain consistency between replicas. One of these is atomic broadcast, which provide the guarantee that all the correct processes deliver the same messages in the same order.

Usually, the performance of distributed algorithms is measured in terms of the number of messages exchanged during the execution. We assess the termination time of the algorithms is also important and in this paper we present a modelling approach which gives statistically significant results for multi-layered distributed protocols.

The performance measures approach is difficult and not reliable in distributed systems, since computation begins and ends on different processors and processes don't have access to a global time. Another major impediment is that results are usually biased by unpredictable network and station loads or by scheduling delays, not related to the studied algorithms. For these reasons the only reliable way to study the termination time for distributed algorithms is by modelling and simulation. Modelling distributed algorithms using the analytical technique may be effective only for algorithms with deterministic number of exchanged messages [2]. This is not the case for the Chandra-Toueg consensus algorithm and for atomic broadcast [1], where the number of generated messages depends on the communication delays. Inversely, the traffic produced by the algorithms has a direct influence on the end-to-end delay experienced by a message in the underlying network. A previous attempt to model the consensus algorithm was made in [3], but the solution requires the assumption there is a maximum message delay [4]. The model proposed in this paper doesn't have to change the asynchronous system model.

2. The model and simulation results

This paper is concerned with a Petri net model for a three-layers distributed protocol (UDP/IP, consensus, atomic broadcast) and the model simulation results, which give probabilistic guarantees on the termination time of the consensus and atomic broadcast algorithms and allow optimal choices for the Fault Detectors parameters.

Our approach is to use a formal description technique with sound mathematical basis, like Petri nets. A Petri net model has the advantages of graphical representation, simple specification and the ability to model synchronisation and parallelism. Classical Petri nets describing real systems tend to be complex and extremely large. This leads to extensions of the initial Petri nets, like introducing "colour", "time" and "hierarchy" [5]. We built our Petri nets using EXSPECT, a software tool developed by Eindhoven University of Technology [6], which implements HCTPN.

Distributed algorithms may be seen as upper layer applications based on a lower layer, the network. We focused on a very common network protocol, UDP/IP, built on top of an Ethernet CSMA/CD bus. First, a simple model for UDP/IP in a private LAN was built. The idea is the end-to-end time experienced by a message in the underlying network can be decomposed into the time spent on the sending station's CPU (t_S), the network delay (t_n) and the time spent on the receiving station's CPU (t_R). If the network or the needed CPUs are busy, a message has to wait, which adds some waiting times to the end-to-end delay. We assume point-to-point communications. The network controller has its own processor which is responsible for implementing the Ethernet protocol. This network model is suitable for a large range of computer architectures and is independent of specific implementations of the UDP/IP protocol. Our UDP/IP model was validated against the performance results of a real system and translated into a Petri net.

In our model the consensus algorithm [1] uses the UDP/IP layer for exchanging messages. In the consensus each participant process p_i, i=1,..., n, proposes an initial value v_i. At the end of consensus each process agrees to adopt one value v chosen from the set of the proposed values v_i.

We denote by f the maximum number of faulty processes. The algorithm [1] proceeds in asynchronous rounds. Each round is divided into 4 asynchronous phases. In Phase 1 each participant sends its current estimate to the current coordinator c, timestamped with the round number when this estimate was adopted. In Phase 2 the coordinator c waits until it receives n-f estimates from the participants, chooses the one with the largest timestamp and sends it to all the processes as the new estimate proposed by c, $estimate_c$. In Phase 3 a process p waits for $estimate_c$ or for a suspicion from his Failure Detector. If p receives $estimate_c$, it replies with an acknowledgement *ack* indicating he adopted $estimate_c$ as its own estimate. If p suspects c, it replies with a negative acknowledgement *nack*. In Phase 4 c gathers n-f votes (*acks* or *nacks*). If all the votes are *ack*, then c broadcasts a request to decide $estimate_c$. Any time a process delivers a decision request, it decides accordingly.

We define the termination time of the consensus as the time elapsed since the first process sends a message in Phase 1 of the first round until all the correct processes have decided. The protocol was modelled using Petri nets and a unified hierarchical model was obtained by integrating the UDP/IP Petri net model in the consensus model. The dynamic behaviour of the system under study is represented in the form of token movements within the model. For a detailed description of the model, see [7].

Simulating the consensus model we can obtain probabilistic results on the termination time of this algorithm. Simulation is also a method to study the consensus behaviour under different input conditions (without and with crashes) and to choose optimal values for the Fault Detectors (FD) parameters, Δ_i and Δ_{to}. The implementation considered is an approximation of an Eventually Strong Failure Detector: FD_i sends periodically, with an interrogation delay Δ_i, an interrogation message to all FD_j. Any time FD_i receives an interrogation message from FD_j, it replies with an answer message to FD_j. If after a time-out delay Δ_{to} after sending an interrogation message to FD_j, the processor p_i hasn't received any message from p_j, FD_i adds p_j to the suspected processes set. A system designer may choose between a range of possible Δ_i and Δ_{to} values, according to his time-response requirements, but he must avoid other values which are degrading the terminating time of consensus by increasing the traffic on the network or by generating too many erroneous suspicions. The optimum values depend on the number of participant processes n, as the network traffic may considerably vary for different n values. We can thus provide the statistical guarantee that consensus will finish during a fixed time period with a pre-defined probability. A small risk percentage has to be accepted as inevitable in an asynchronous systems, where there is no upper limit for the communication delays. Using the Central Limit Theorem from statistics and the independent replications method [8], we build 95% confidence interval around the termination time values plotted in figures 1 and 2.

The simulations were made for n=7, t_S=584 µs, t_R=634 µs, t_n=232 µs. The termination time plotted in Figure 1 is in the best case, when no crash failures occur. Figure 2 presents the worst case, when the coordinator of the first round crashes. From Figure 1 we may see the termination time decreases as Δ_i increases, since the interrogation messages contribute at the network traffic and thus increase the end-to-end delay. For the four Δ_i values considered the recommended time-out value Δ_{to} is 20 ms. Smaller values increase the termination time by producing erroneous suspicions and thus increasing the number of rounds. From Figure 2 we may see that Δ_i=25 ms or Δ_i=30 ms would be a bad choice, since they generate too much traffic on the network. Indeed, as these two values are smaller than Δ_i=35 ms, they would normally provide a smaller termination time when crashes occur (Figure 2), since a crash would be more quickly detected. This is not the case in Figure 2, where the Δ_i=25 ms and Δ_i=30 ms curves are above the Δ_i=35 ms and Δ_i=38 ms curves. Acceptable Fault Detectors parameters are Δ_i=35 ms and Δ_{to}=20 ms or Δ_i=38 ms and Δ_{to}=20 ms. If we choose Δ_i=35 ms and Δ_{to}=20 ms the termination time of consensus will be smaller than 76 ms with a probability of 95%. For Δ_i=38 ms and Δ_{to}=20 ms the termination time of the consensus will be smaller than 80 ms with 95% probability. A system designer may choose any Δ_i value greater than 35 ms according to his time-response requirements. The 5% of risk may be decreased by increasing the guaranteed termination time value.

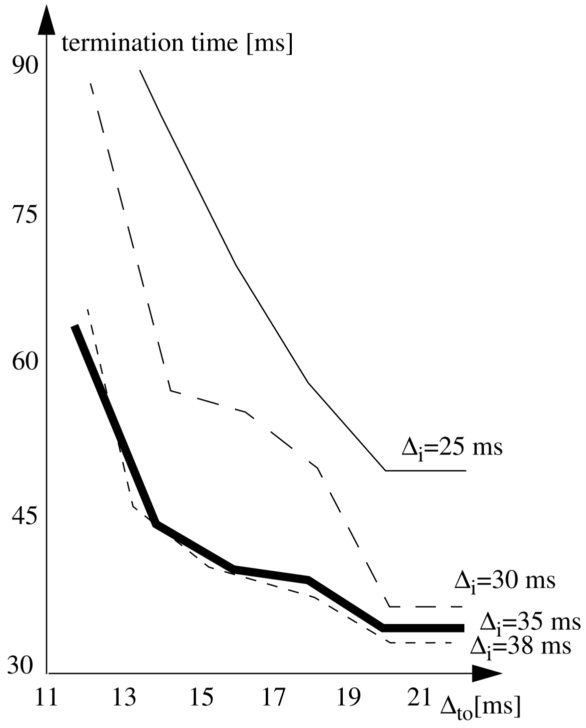

Figure 1

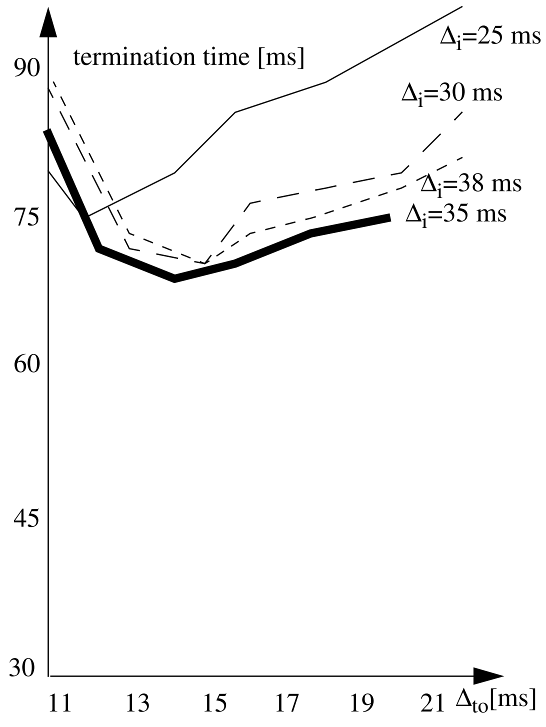

Figure 2

The Atomic Broadcast algorithm is an upper layer protocol which uses the consensus and the Reliable Broadcast algorithms [1]. The final unified Petri net model for the Atomic Broadcast will contain the UDP/IP and consensus sub-nets. Simulating this model we can predict what is the maximum average rate for generating broadcasts without degrading the system performances (an excessive rate may lead to a network overload and increased end-to-end delays). Similarly to consensus, we define the termination time for the Atomic Broadcast algorithm as the time elapsed since a process A-broadcast a message until all the correct processes have delivered that message. Further work will provide probabilistic guarantees on this time.

3. Conclusion

The Petri nets framework offers ample opportunity for graphical modelling of the various causal dependencies between the steps taken by the nodes in a distributed system. The advantage of using Hierarchical Coloured Timed Petri nets and the EXSPECT tool is the possibility to make a good trade-off between the size of the net and the length of the code, represented by arc inscriptions and preconditions. Our net size was kept reasonable by writing over 1000 lines of code lines in Z, a functional language. The Petri net model, built in a hierarchical style, is very flexible, it allows for instance to modify the network model without having to modify the hole net.

The results obtained cannot be used for critical real-time applications since a not null risk percentage has to be accepted as inevitable. There is no absolute limit for the termination time of the consensus or for other distributed algorithms in an asynchronous environment. Instead, the paper presents an original approach for designing distributed algorithms and for evaluating the probability that the algorithm termination time will be smaller than a certain pre-defined value.

References:

[1] Chandra T. D., Toueg S. , «Unreliable failure detectors for reliable distributed systems» , Journal of ACM, 34(1), pp. 225-267, 1996.

[2] Menascé D., Yesha Y. , «On a Unified Framework for the Evaluation of Distributed Quorum Attainment Protocols» , IEEE Trans. on Soft. eng., vol.20, no. 11, nov. 1994.

[3] Vachon J. , «Analyse temporelle d'un algorithme de consensus dans un système asynchrone» , Proc. RenPar'8, Bordeaux, mai 1996, pp. 65-69.

[4] Ponzio S. , «Consensus in the presence of timing uncertainty: omission and byzantine failures», Proc. of the 10th Annual ACM SPDC, August 1991, pp. 125-138.

[5] Jensen K. , Coloured Petri nets : basic concepts, analysis, methods and practical use, Springer Verlag, 1994.

[6] ExSpect Reference Manual - Release 5.0 , Bakkenist ExSpect, Netherlands, november 1994.

[7] Sergent N. , «L'evaluation probabiliste du temps de terminaison d'un algorithme de consensus», Rapport technique, Department of Computer Science, EPFL, 1996, Suisse. To appear.

[8] Computer Performance Modeling Handbook, Peter D. Welch- «The Statistical Analysis of Simulation results», pp. 267-327, Addison-Wesley, 1983.