Dependability Analysis and Verification for Connected Systems*

Felicita Di Giandomenico[1], Marta Kwiatkowska[2], Marco Martinucci[1], Paolo Masci[1,3], and Hongyang Qu[2]

[1] Information Science and Technologies Institute, CNR, Pisa, Italy
[2] Oxford University Computing Laboratory, Oxford, UK
[3] Dept. of Information Engineering, University of Pisa, Italy

Abstract. The Connect project aims to enable the seamless composition of heterogeneous networked systems. In this context, Verification and Validation (V&V) techniques are sought to ensure that the Connected system satisfies dependability requirements. Stochastic model checking and state-based stochastic methods are two appealing V&V approaches to accomplish this task. In this paper, we report on the application of the two approaches in a typical Connect scenario. Specifically, we make clear (i) how the two approaches can be employed to enhance the confidence in the correctness of the analysis, and (ii) how the complementarity of these approaches can be fruitfully exploited to extend the analysis.

1 Introduction

The Connect project [14] aims at dropping the barriers that prevent heterogeneous networked systems from being Connected, by enabling their seamless composition in spite of technological evolution. To achieve this aim, Connect intends to dynamically synthesise the Connectors that allow the networked systems to communicate. The resulting emergent Connectors compose and adapt interaction protocols run by the Connected systems.

In addition to functional properties, Connectors generally need to satisfy non-functional properties as well. Therefore, an evaluation to assess whether the Connector specification is adequate to satisfy the dependability requirements is highly desirable. Indeed, Verification and Validation (V&V) techniques are sought in Connect to ensure that networked systems, as well as the generated bridging Connectors, satisfy specified levels of accomplishment for dependability requirements, according to pertinent dependability metrics. Note that, in Connect, dependability is used as a term inclusive of several non-functional properties [4], e.g., including also performance aspects.

In Connect we are interested in quantitative, or probabilistic, dependability evaluation. To this purpose, as indicated in [1], the two main approaches are modelling and (evaluation) testing. Since evaluation testing assumes that a test

* This work is supported by the European FP 7 project CONNECT (IST 231167).

suite is run on the system under test, which in the Connect vision is not a priori available, for the analysis we focus on modelling.

Modelling is composed of two phases: (i) building a model for the system from the elementary stochastic processes that represent the behaviour of the components of the system and their interactions (these elementary stochastic processes relate to failures, to repair, service restoration and possibly to system duty cycle or phases of activity); (ii) processing the model to obtain the expressions and the values of the dependability measures of the system.

Research in dependability analysis has developed a variety of modelling techniques, each of which focuses on particular levels of abstraction and/or system characteristics. As reported in [18], important classes of model representation include combinatorial methods (such as Reliability Block Diagrams), model checking, and state-based stochastic methods. We are not interested in combinatorial methods, which are simpler approaches and do not easily capture certain features, such as stochastic dependence and imperfect fault coverage. Therefore, in the context of Connect, we consider as evaluation techniques:

– Stochastic model checking, which is a formal verification technique for the analysis of stochastic systems. It is based on the construction of a probabilistic model from a precise, high-level description of a system's behaviour.
– State-based stochastic methods, which use state-space mathematical models expressed with probabilistic assumptions about time durations and transition behaviours. They allow explicit modelling of complex relationships (e.g., concerning failure and repair processes), and their transition structure encodes important sequencing information.

In this paper, these two approaches are applied to a Connect scenario to perform various dependability analyses, thus showing their complementarity in assessing dependability properties. First, both approaches are used to validate two basic dependability properties. Next, extra properties are checked by the appropriate approach, selected according with its ability to cope with the specific type of analysis. Indeed, the different formalisms and tools implied by the two methods allow: (i) on the one hand, to complement the analysis from the point of view of a number of aspects, such as level of abstraction/scalability/accuracy, for which the two approaches may show different abilities to cope with; and (ii) on the other hand, through the inner diversity, provide cross-validation to enhance confidence in the correctness of the analysis itself. The rest of the paper is structured as follows. Section 2 introduces the tools implementing the approaches, and properties supported by the tools. A Connect scenario, namely, a model of a distributed market scenario, is presented in Section 3, and analysed in Section 4. We conclude the paper in Section 5.

2 Analysis and verification tools

In this section, we present a brief description of PRISM and Möbius, which implement stochastic model checking and state-based stochastic methods respectively.

2.1 PRISM

PRISM [12] is a popular probabilistic model checker, which can handle discrete and continuous time Markov chain models (DTMCs and CTMCs), as well as Markov decision processes (MDPs). DTMC and MDP models may be verified against probabilistic temporal logic formulae given in terms of PCTL (Probabilistic Computational Tree Logic) [11, 6], as well as cost/reward-based properties, and LTL (Linear Temporal Logic) formulae [19]. CTMCs may be verified against CSL (Continuous Stochastic Logic) [2, 3] formulae. Both states and transitions in a system can be associated with rewards, which allow for the checking of both instantaneous and cumulative properties.

So far PRISM has been applied to numerous probabilistic models, such as network protocols, security protocols, randomised distributed algorithms, biological processes, etc.

Modelling formalism. As CONNECTed systems evolve in a manner where a system stays in a state for a certain period of time and then moves to a successor state, we model such systems using CTMCs because they preserve the memoryless property. For CTMCs, the memoryless property not only requires that the probability of firing a transition totally depends on the current state, but also asks the probability to be independent of the elapsed time so far. The only continuous probability distribution exhibiting this property is the exponential distribution, which associates a *rate* to each transition in CTMCs. The rate can be understood as the average number of times we can execute the transition per unit of time. The probability of executing a transition from the current state within t time units is $1 - e^{-\lambda \cdot t}$. The rates associated with all transitions in a CTMC can be stored in a *transition rate matrix* $\mathbf{R}$, where each entry represents a rate between a pair of states. A transition can only occur from state s to state s' if $\mathbf{R}(s, s') > 0$. If more than one transition can be executed in state s, the successor state is determined by the first transition being taken. Let S be the set of states in a CTMC. The amount of time for which the system stays in s before any transition occurs is governed by an exponential distribution with rate $E(s)$ such that $E(s) \overset{def}{=} \sum_{s' \in S} \mathbf{R}(s, s')$. The probability of going to successor state s' from state s is calculated as follows.

$$\mathbf{P}(s, s') = \begin{cases} \mathbf{R}(s, s')/E(s) & \text{if } E(s) \neq 0, \\ 1 & \text{if } E(s) = 0 \text{ and } s = s', \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Properties of interest. Using formula (1), we can compute the probability of reaching a set of target states through all paths. *Steady-state* behaviour is another interesting property for CTMC models. The steady-state probability for a state s is the probability of being in s in the long run, which can be used to infer the percentage of time that the model spends in s in the long run.

In addition to path and steady-state probabilities, we consider two additional types of reward for instantaneous and cumulative rewards separately. Every transition is associated with an instantaneous reward and every state has a cumulative reward. The former is the actual reward obtained when the system executes a transition, and the latter is the coefficient, at which the reward is computed in a state, for the amount of time spent in that state. We can define the expected reward of reaching a set of target states F through paths. The reward for a path that does not pass any target state is set to ∞. Thus, the expected reward of reaching a state in F from state s is finite if all non-zero probability paths starting from s pass a state in F.

2.2 Möbius

Möbius [8] is a popular software tool that provides a comprehensive framework for model-based dependability and performance evaluation of systems. The main features of the tool include: (i) multiple high-level modelling formalisms, including, among others, Stochastic Activity Networks (SANs) [20] and PEPA fault trees [10]; (ii) a hierarchical modelling paradigm, allowing one to build complex models by first specifying the behaviour of individual components and then by combining the components to create a model of the complete system; (iii) customised measures of system properties; (iv) distributed discrete-event simulation, to evaluate measures using efficient simulation algorithms to repeatedly execute the system and gather statistical results of the measures; (v) numerical solution techniques, to obtain exact solutions for Markov models.

Modelling formalism. We model the system with Stochastic Activity Networks (SANs). SANs are stochastic extensions of Petri Nets; they have a graphical representation and consist of four primitive objects: *places*, *activities*, *input gates* and *output gates*. Places in SANs have the same interpretation as in Petri Nets, i.e., they hold tokens. The number of tokens in a place is referred to as the marking of that place, and the marking of the SAN is the set of all place markings. There are two types of activities: instantaneous and timed. Timed activities represent actions that have a duration that impacts the performance of the modelled system, e.g., message transmission time, recovery time, time to fail. The duration of each timed activity is expressed via a time distribution function. Both instantaneous and timed activities may have *case probabilities*. Each case probability stands for a possible outcome of the activity, and can be used to model probabilistic aspects of the system, e.g., probability for a component to fail. Input gates control the enabling of activities, and output gates define the state change that will occur when an activity completes.

SAN models can be composed with *Join* and *Rep* operators. Join is used to compose two or more SANs. Rep is a special case of Join, and is used to construct a model consisting of a number of replicas of a SAN. Models in a composed system interact via *Place Sharing*. Place Sharing is a composition formalism based on the notion of sharing places via an equivalence relation.

Properties of interest. Properties of interest are specified with *reward functions*. Each reward function is a C++ function that specifies how to measure a property on the basis of the marking of the SAN. There are two kinds of reward functions: *rate reward* and *impulse reward*. Rate rewards can be evaluated at any time instant. Impulse rewards are associated with specific activities and they can be evaluated only when the associated activity completes. Measurements can be conducted at specific time instants, over periods of time, or when the system reaches the steady state.

3 The distributed market scenario

We consider a case study based on a distributed market, where consumers execute a discovery protocol to gather information on the products sold by merchants. The discovery phase is performed in two steps. In the first step, consumers interoperate with all merchants to gather a list of all available products. In the second step, consumers select a product type and continue to interoperate with a subset of merchants (those that sell the selected product type) to gather additional information on the product.

Since consumers and merchants have heterogeneous devices that execute different protocols, interoperability among them is obtained via a CONNECTor that bridges the functional mismatches between the protocols. Without loss of generality, we assume that all merchants have the same protocol $P1$, and that all consumers have the same protocol $P2$ ($P2 \neq P1$).

Figure 1 illustrates the LTSs (Labelled Transition Systems) for consumer, merchant, CONNECTor and CONNECTed system. For simplicity, in Figure 1(c), we assume that there are two merchants in the market, one of which sells the product requested by a consumer. A larger number of merchants is handled by the CONNECTor in the same way. In the first discovery step, the CONNECTor receives the consumer's request `rdgBrowse` and sends a message `mSearch` to all merchants. Each merchant responds with a message `resp`. Note that the CONNECTor can handle any order of responses from the merchants. The CONNECTor then sends back to the consumer a message `tupleListBrowse`. In the second step, the CONNECTor converts the consumer's request `rdgGetInfo` into two requests `httpGet` and `httpGetResp` to the merchant selling the product, and obtains the responses `soapReqGetInfo` and `soapRespInfo` respectively. In the end, the CONNECTor returns a response `tupleListInfo` to the consumer. The LTS for the composed system is illustrated in Figure 1(d). The complete description of the scenario for a CONNECTed system with one consumer and one merchant is presented in [13].

Basic dependability properties. In this paper, we consider two basic properties that will be analysed using both PRISM and Möbius. In Section 4, further properties will be analysed in order to extend the analysis in accordance with the capabilities of the approaches.

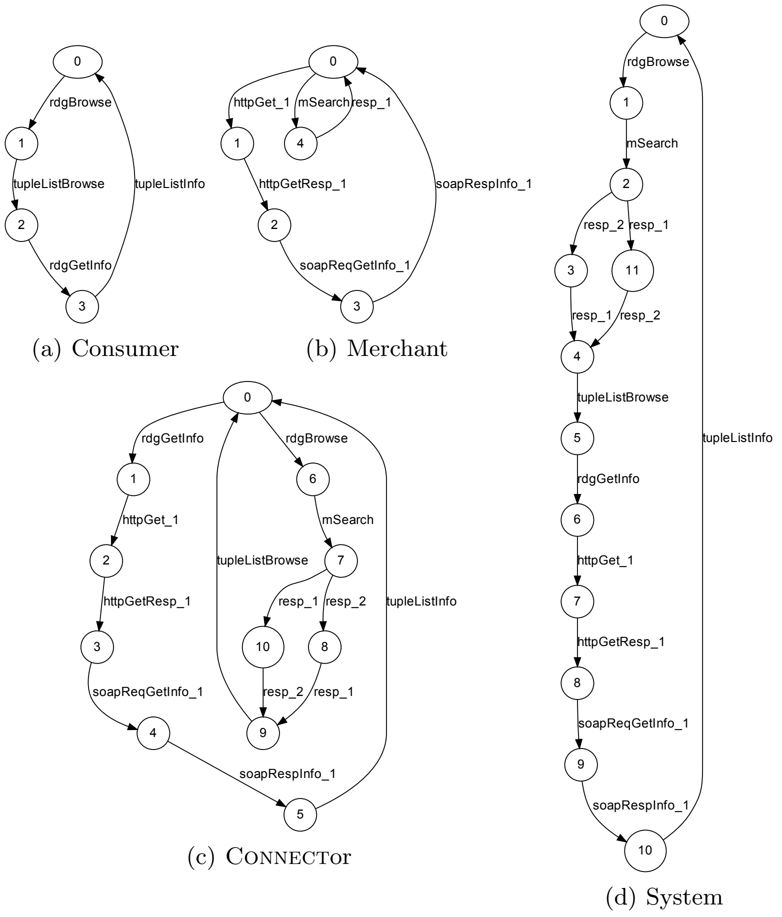


Fig. 1. LTS models for distributed market scenario

- *Coverage in the first step.* We are interested in the percentage of merchants that can give a response to the CONNECTor in a given time interval during the first step. This property is affected by the network characteristics in the market, e.g., the number of consumers, the number of merchants, and the reliability of the channel. In the rest of the paper, we analyse the result for the case in which there is one consumer and three merchants. The transmission speed is modelled by different values of the rate associated with the transitions among consumer, CONNECTor, merchants.
- *Latency in the second step.* Latency is more interesting in the second step, as the communication between the CONNECTor and the merchants involves more message exchanges. Similarly to coverage, we assume there is a single consumer in addition to three merchants, all of which are selling the requested product. We measure the time spent by the consumer from when it starts to send `rdgGetInfo` to the arrival of `tupleListInfo`.

4 Dependability analysis and verification

In this section, we model the CONNECT scenario using PRISM and Möbius respectively, and perform dependability analysis on the models.

4.1 PRISM models

The LTSs for the scenario in Figure 1 can be translated into the PRISM CTMC model in a straightforward manner. In detail, each component LTS in Figure 1(a)-1(b) is translated into a PRISM module in the following way. We define a variable in each module, whose domain is the set of states in the corresponding LTS and whose initial value is the initial state of the LTS. Each transition in the module has the same label as the corresponding one in the LTS. Since the LTSs do not contain information for rates, we deliberately assign rate $R1 = 1$ to all transitions between the consumer and the connector, and assign $R2$ (which may vary) to those between the connector and the merchants.

In order to check the basic properties specified in Section 3, we need to add the timeout mechanism to the model. Two timeouts T1 and T2 are introduced to model the maximum time for the first step and second step respectively. However, the deterministic delay in timeout breaks the basic rule of CTMCs: all delays in a CTMC model respect exponential distributions, and this makes the model difficult to verify. In this paper, we use an Erlang distribution to approximate a deterministic delay T by a sequence of transitions, each of which has an exponential distribution of rate k/T, where k is the number of transitions in the sequence. The accuracy of the approximation, as well as the verification time, increases as k increases. In the experiments, we choose k to be $T \times 10$, i.e., the rate in the Erlang distribution is 10, which is a reasonable trade-off between speed and accuracy.

4.2 Stochastic verification

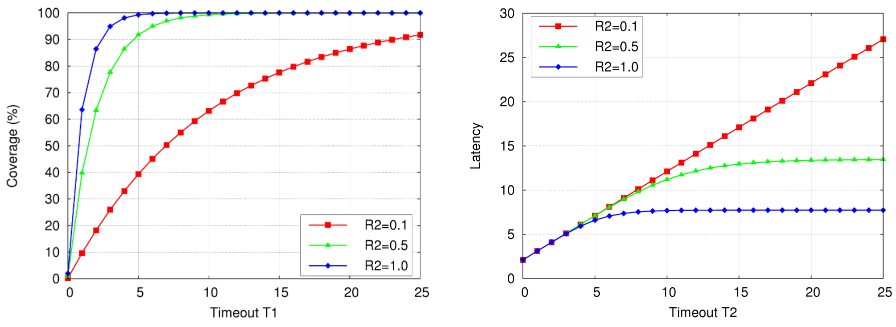
In this section, we first show the verification results for the basic properties, and then discuss additional properties that can be verified using CSL. For each property, we construct a set of experiments by choosing different values for timeouts $T1$ and $T2$, and letting $R2$ range over values 0.1, 0.5 and 1.0 respectively. This way, we can illustrate the trend as $T1$ (resp. $T2$) increases.

Coverage. This property is specified by the following CSL reward formula on the reward structure *Coverage*:

$$\mathcal{R}_{=?}[C^{\leq T}], \tag{2}$$

where $C^{\leq T}$ represents the cumulative reward up to time bound T. The structure *Coverage* associates the real value m/n to states where, among the total number n of merchants, m merchants send back their response to the connector within

$T1$ time units after they receive the request `mSearch`. We choose T to be $T1 + 10$ to take into account the time for transmitting message `rdgBrowse`. This formula calculates the expected cumulative reward within T time units, and the verification results are shown in Figure 2(a).



(a) Coverage (b) Latency

Fig. 2. Verification results

Latency. In the second step of the discovery, the connector triggers a timeout after $T2$ units of time when it receives message `rdgGetInfo` from the consumer. When the timeout occurs, the connector does not wait for pending responses from merchants, and returns `tupleListInfo` to the consumer. To verify this property, we use formula (2) on the reward structure *Latency*, which assigns $1/(T2/k) = 10$ to each transition used to approximate the timeout. The results are depicted in Figure 2(b).

Probability of receiving replies from all merchants in the second step. In addition to the basic properties, we are also interested in the probability of receiving responses from all merchants contacted within deadline $T2$ in the second discovery step. This property is formulated as follows:

$$\mathcal{P}_{=?}[F^{\leq T}(m = n)], \tag{3}$$

where $T = T2 + 2$, n is the total number of merchants contacted and m is the number of merchants that reply before the timeout. This formula computes the probability of all paths that can reach a state satisfying $m = n$ within T units.

In formula (3), $T = T2 + 2$ is a reasonable bound to take into account the transmission time for `rdgGetInfo` and `tupleListInfo`, given $R1 = 1$ for these two messages. However, it is still possible that it takes longer than 2 units of time to transmit these messages, which generates a small error in the experimental results. Steady state probability can be used to overcome this problem. If we ignore truncation errors, formula (4) gives an accurate value for the required probability at a cost of longer verification time.

$$\mathcal{S}_{=?}[m = n] \tag{4}$$

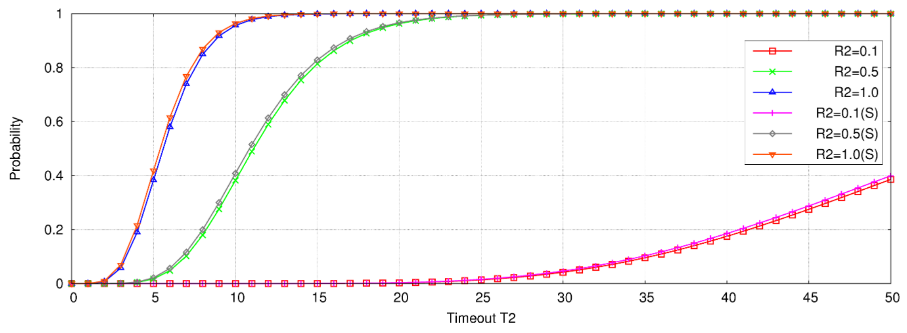


Fig. 3. Verification results for probability and steady-state probability

In Figure 3, the curves labelled $R2 = 0.1$, $R2 = 0.5$ and $R2 = 1.0$ are probabilities computed using formula (3), and the others are computed using formula (4) for the same rates, respectively. Note that the accurate result for the coverage property can be computed by the steady-state reward formula:

$$\mathcal{R}_{=?}[\mathcal{S}] \tag{5}$$

on the corresponding reward structure. However, the difference between the values computed by formulae (2) and (5) in this example is negligible.

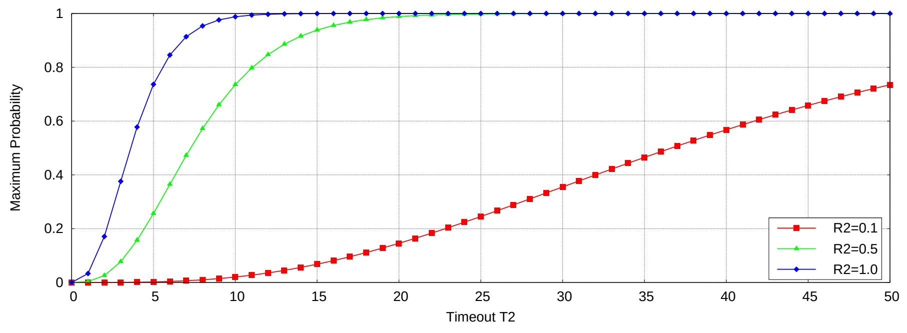


Fig. 4. Verification results for maximum probability

Maximum probability of receiving replies from all merchants in the second step. In the second discovery step, a certain number $n1$ of merchants, instead of the total number n of merchants, are contacted for the information of the request product. This number $n1$ may vary depending on many factors, such as the traffic in the network. The property considers the maximum probability of receiving responses from all $n1$ merchants in the following situation. At the beginning of the second step, when the consumer is in state 2 in Figure 1(a), (formulated as `Consumer` $= 2$), the number of merchants that will be contacted is beyond a certain bound, i.e., $n1 \geq n \cdot a$ ($a \in (0, 1]$). Let $m1$ be the number of responses received before timeout occurs at $T2$. The property can be checked by formula (6):

$$\mathcal{P}_{=?}[F(m1 = n1)\{n1 \geq n \cdot a \wedge \texttt{Consumer} = 2\}\{\texttt{max}\}]. \tag{6}$$

The results for $a = \frac{1}{3}$ and $n = 3$ are illustrated in Figure 4.

4.3 SAN models

The SAN models of merchant, consumer and CONNECTor are shown in Figure 5. The model of the CONNECTed system is obtained by composing, via place sharing, the SAN models of consumer, CONNECTor and merchants (the SAN model of the merchants is obtained by replicating a merchant with the Rep operator). There is a shared place for each pair of activities that represent send/receive actions: send activities add tokens in the shared place, while receive activities remove tokens from the shared place and use the marking of the shared place as enabling condition. Note that, in general, a send activity may control $n > 1$ receive activities (e.g., in the case of a message with multicast/broadcast addresses); in this case, the send activity will add n tokens to the shared place to allow the simultaneous enabling of the receive activity of n receivers.

Timing aspects for send/receive actions are taken into account in the SAN models as follows: when n receive activities complete simultaneously after a send action completes, the receive activities are instantaneous and the send activity is timed; when n receive activities complete independently after a send action completes, the receive activities are timed and the send activity is instantaneous. Timeouts are modelled with timed activities that force the enabling of other activities.

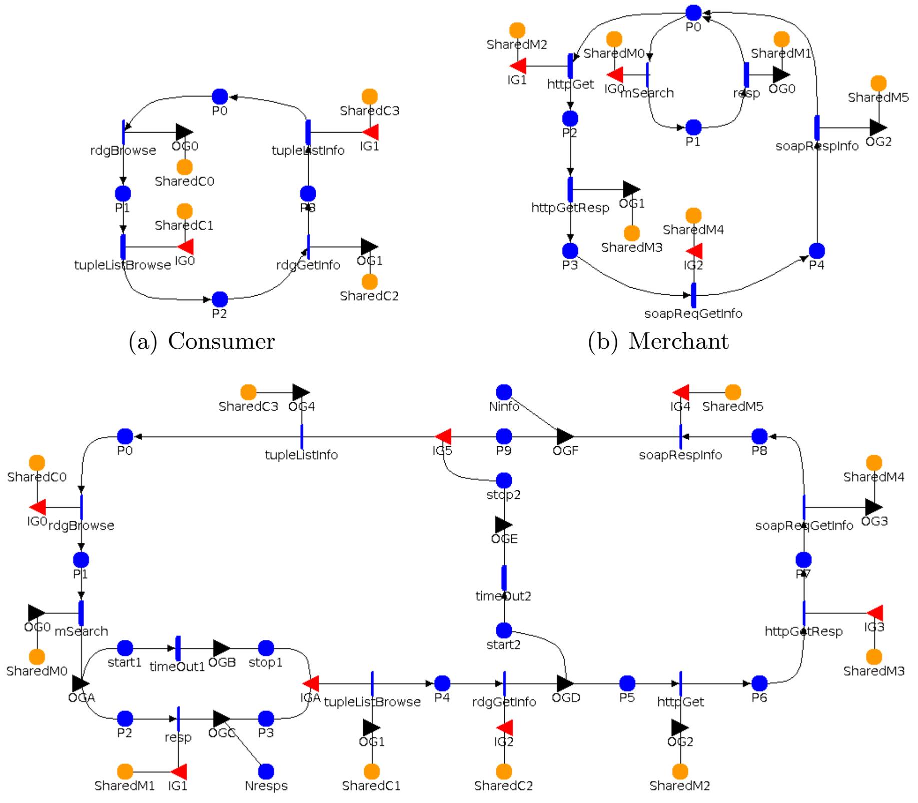
In the following we describe in detail the behaviour of the model of CONNECTed system during the first step. In the description, we will use the prefixes `C`, `M[i]`, and `CON` to disambiguate the names of local places, activities and gates of consumer, merchants, and CONNECTor.

Initially, all places in the models have zero tokens, except `P0`, which contains one token in all models. The consumer starts the communication, because `C.rdgBrowse` is the only enabled activity. When `C.rdgBrowse` completes, one token is placed in `C.P1` and one token in `SharedC0`. At this point, `CON.rdgBrowse` is enabled. When `CON.rdgBrowse` completes, one token is moved from `SharedC0` to `CON.P1`, and `CON.mSearch` becomes enabled.

When `CON.mSearch` completes, the marking changes as follows: n tokens are placed in `SharedM0`, because n merchants must be involved in the communication; n tokens are placed in `CON.P2`, because the CONNECTor must wait for one `resp` from each merchant; one token is placed in `CON.start1`, because the CONNECTor has a timeout on the maximum waiting time.

Each token in `SharedM0` enables the instantaneous activity `M[i].mSearch` of a merchant. All such activities complete immediately[1] and enable `M[i].resp`.

[1] When instantaneous activities and timed activities are enabled at the same time, all instantaneous activities complete first.



(a) Consumer (b) Merchant

(c) CONNECTor

Fig. 5. SAN models

Whenever a `M[i].resp` completes, a token is placed in `SharedM1` to enable the instantaneous activity `CON.resp` of the CONNECTor, which places a token in `CON.P3` and `CON.Nresps`. The number of tokens in `CON.Nresps` represents the number of merchants that will participate to the interactions during the second step. This behaviour continues until `CON.tupleListBrowse` becomes enabled, i.e., either when `CON.timeOut1` completes (one token is placed in `CON.stop1`), or n responses are received from the merchants (n tokens are present in `CON.P3` and `CON.Nresps`).

4.4 State-based stochastic analysis

In this section, we present the analysis performed with Möbius: first, we cross-validate the results obtained by PRISM for coverage and latency; second, we scale up to large systems with hundreds of merchants; third, we refine the SAN models to take into account some real-world aspects that have an impact on coverage and latency, such as traffic patterns and communication failures.

Cross validation. The reward functions are expressed as follows.

Coverage. This property is specified by accumulating over time the following impulse reward on `CON.resp` (`NMerchants` is a parameter of the composed model, and holds the number of merchants):

```
double coverage() {
   if ( CON->start1->Mark() > 0 ) { return 1.0/NMerchants; }
   return 0;
}
```

Latency. This property is specified by accumulating over time the following rate reward function:

```
double latency() {
   if ( CON->P4->Mark() > 0 || CON->start2->Mark() > 0
        || CON->stop2->Mark() > 0) { return 1; }
   return 0;
}
```

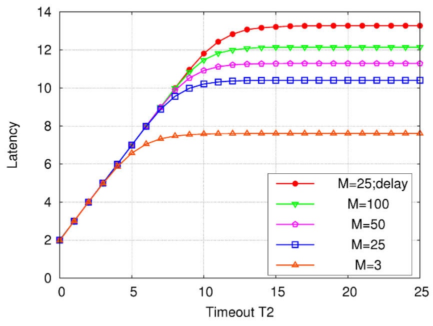
We were able to successfully reproduce with Möbius the verification results of PRISM. We used simulation, and the relative difference between the average results was always below 2%.

Scalability of the models. CONNECTed systems may include an arbitrary large number of networked systems. Therefore, we investigated the scalability of the SAN model of the CONNECTed system by analysing large networks. The developed SAN model of the CONNECTed system is parametric with respect to the number of merchants: networks with different number of nodes can be modelled by changing only one model parameter.

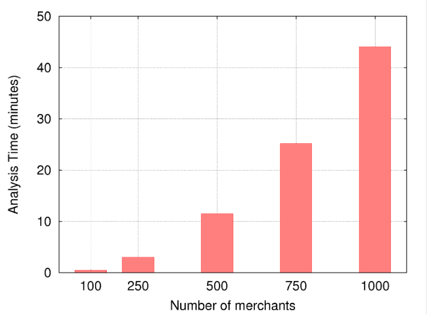
We successfully assessed coverage and latency for scenarios with hundreds of merchants. Figure 6(a) shows the analysis results for latency in scenarios with at most 100 merchants. The number of batches needed to reach a confidence level of 95% and a confidence interval of 10% for the considered models was always below $10K$, because the models are relatively simple. Figure 6(b) reports the average time to complete $10K$ simulation batches for different number of merchants on a system with a 2.8GHz Intel Core2 Quad processor.

Latency for different traffic patterns. CONNECTed systems are expected to be a mix of heterogeneous user applications, each of which may have different characteristics and requirements. Currently, there is no single traffic distribution that can efficiently capture the traffic characteristics of all types of networks under every possible situation. A large number of empirical studies have shown that network traffic is self-similar and that it generally exhibits multiple time-scale behaviour [16]. These aspects can be modelled with subexponential distributions, such as Weibull and Lognormal.

We investigated the effect of different subexponential distributions on latency by changing the probability distribution function of the timed activities. For a

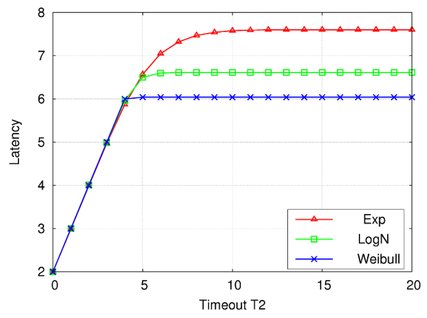


(a) Latency

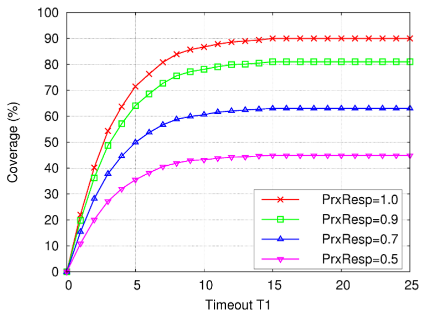


(b) Analysis Time

Fig. 6. Latency and time required for the analysis for different system size



(a) Latency for different traffic patterns



(b) Coverage for different P_{rxResp}

Fig. 7. Latency and coverage in different settings

fair comparison, we have chosen distribution parameters that allow the same mean value in all cases. The analysis results are shown in Figure 7(a). We can notice that different traffic patterns lead to different latency profiles.

Coverage in the case of failures. Communication in the real-world can be subject to failures. Therefore, failure modes need to be accounted for when setting up the system model. Failure modes can pertain the value domain (e.g., wrong output), and/or the time domain (e.g., omission). In this section, we assess coverage in the case of omission failure of the multicast search received by the merchants (`M[i].mSearch`) and omission failure of the responses sent by the merchants (`M[i].resp`). Figure 7(b) shows the coverage profiles for different probability P_{rxResp} of failures of `resp`; in the figure, the probability of failure of `M[i].mSearch` depends on the timeout value and is derived from the analysis results reported in [17]; e.g., for $T1 = 10$, the probability of failure of `M[i].mSearch` is 0.87.

5 Conclusions

We have shown two approaches to analyse dependability properties for a Connect scenario. As can be seen in the previous section, the experimental results produced by one approach match those by the other approach for a significant range of properties. Each approach has its own advantages regarding modelling capability, specification of properties, scalability, etc. For example, bounded until formula $\phi_1 \mathcal{U}^I \phi_2$, steady state formula $\mathcal{S}_{\bowtie p}[\phi]$ and more complex CSL formulae can be verified in PRISM without manually augmenting models, while Möbius can deal with larger sized models and can mix exponential distributions with other distributions.

During the case study, we also found that the cross validation was particularly useful to improve the confidence in the correctness of the models. For example, by analysing the mismatches between results produced by PRISM and Möbius on the common properties, we were able to remove subtle non-deterministic behaviours that were erroneously present in the models, and eliminated the mismatches.

In the future, we are planning to apply both approaches to a more complex case study, and explore further the cross-fertilisation capabilities of PRISM and Möbius. In particular, we would like to exploit the assume-guarantee reasoning method [15, 9] implemented in PRISM to analyse large models. We were unable to apply this here because the assume-guarantee approach has only been developed for Markov decision processes and safety properties at present. In addition, we are also interested in speeding up our approaches via incremental verification and analysis, and developing online techniques to provide support for on-the-fly CONNECTor synthesis [5], such as those based on [7].

References

1. A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.
2. A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Verifying continuous time Markov chains. In R. Alur and T. Henzinger, editors, *Proc. 8th International Conference on Computer Aided Verification (CAV'96)*, volume 1102 of *LNCS*, pages 269–276. Springer, 1996.
3. C. Baier, J.-P. Katoen, and H. Hermanns. Approximate symbolic model checking of continuous-time Markov chains. In J. Baeten and S. Mauw, editors, *Proc. 10th International Conference on Concurrency Theory (CONCUR'99)*, volume 1664 of *LNCS*, pages 146–161. Springer, 1999.
4. A. Bertolino, F. Di Giandomenico, A. Di Marco, V. Issarny, F. Martinelli, P. Masci, I. Matteucci, R. Saadi, and A. Sabetta. Dependability in dynamic, evolving and heterogeneous systems: the connect approach. In *Proc. 2nd International Workshop on Software Engineering for Resilient Systems (SERENE2010)*, 2010.
5. A. Bertolino, P. Inverardi, V. Issarny, A. Sabetta, and R. Spalazzese. On-the-fly interoperability though automated mediator synthesis and monitoring. In *Proc. 4th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISOLA'10)*, LNCS. Springer, 2010. To appear.

6. A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Proc. FSTTCS'95*, volume 1026 of *LNCS*. Springer, 1995.
7. R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola, and G. Tamburrelli. Dynamic QoS management and optimisation in service-based systems. *IEEE Transaction on Software Engineering*. To appear.
8. G. Clark, T. Courtney, D. Daly, D. D. Deavours, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. G. Webster. The Möbius modeling tool. In *9th Int. Workshop on Petri Nets and Performance Models*, pages 241–250, Aachen, Germany, September 2001. IEEE Computer Society Press.
9. L. Feng, M. Kwiatkowska, and D. Parker. Compositional verification of probabilistic systems using learning. In *Proc. 7th International Conference on Quantitative Evaluation of Systems (QEST'10)*. IEEE CS Press, 2010. To appear.
10. R. Gulati and J. B. Dugan. A modular approach for analyzing static and dynamic fault trees. In *Annual Reliability and Maintainability Symposium*, pages 57–63. IEEE Computer Society Press, 1997.
11. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.
12. A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *Proc. TACAS'06*, volume 3920 of *LNCS*. Springer, 2006.
13. P. Inverardi, V. Issarny, and R. Spalazzese. A theory of mediators for eternal CONNECTors. In *Proc. 4th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISOLA'10)*, LNCS. Springer, 2010. To appear.
14. V. Issarny, B. Steffen, B. Jonsson, G. Blair, P. Grace, M. Kwiatkowska, R. Calinescu, P. Inverardi, M. Tivoli, A. Bertolino, and A. Sabetta. CONNECT Challenges: Towards Emergent CONNECTors for Eternal Networked Systems. In *14th IEEE International Conference on Engineering of Complex Computer Systems*, pages 154–161. IEEE Computer Society, 2009.
15. M. Kwiatkowska, G. Norman, D. Parker, and H. Qu. Assume-guarantee verification for probabilistic systems. In *Proc. of 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'10)*, volume 6015 of *LNCS*. Springer, 2009.
16. W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.
17. P. Masci, S. Chiaradonna, and F. Di Giandomenico. Dependability analysis of diffusion protocols in wireless networks with heterogeneous node capabilities. In *8th European Dependable Computing Conference (EDCC2010)*, pages 145–154. IEEE Computer Society, 2010.
18. D. M. Nicol, W. H. Sanders, and K. S. Trivedi. Model-based evaluation: from dependability to security. *IEEE Transactions on Dependable and Secure Computing*, 1:48–65, January-March 2004.
19. A. Pnueli. The temporal logic of programs. In *Proc. 18th Annual Symposium on Foundations of Computer Science (FOCS'77)*, pages 46–57. IEEE Computer Society Press, 1977.
20. W. H. Sanders and J. F. Meyer. Stochastic Activity Networks: formal definitions and concepts. pages 315–343, 2002.