Applying Formal Methods to Gossiping Networks with mCRL and Groove

Pepijn Crouzen
Department of Computer Science
Saarland University
crouzen@alan.cs.uni-sb.de

Jaco van de Pol Arend Rensink
Department of Computer Science
University of Twente
{vdpol,rensink}@cs.utwente.nl

ABSTRACT

In this paper we explore the practical possibilities of using formal methods to analyze gossiping networks. In particular, we use μCRL and Groove to model the peer sampling service, and analyze it through a series of model transformations to CTMCs and finally MRMs. Our tools compute the expected value of various network quality indicators, such as average path lengths, over *all* possible system runs. Both transient and steady state analysis are supported. We compare our results with the simulation and emulation results found in [10].

1. INTRODUCTION

Gossiping networks provide a novel way of constructing distributed systems. A gossiping network consists of a large number of simple nodes, which have a limited view of the network. The idea is that information is dissipated in a *gossiping* style, i.e. every node communicates its information to a small number of other nodes in the same way people spread gossip through a community. This style of communication is also called *epidemic* for its similarity to a disease spreading through a population. Gossiping networks have been used successfully in a number of applications (for an overview see [6]).

In [2] the use of formal methods is proposed to analyze the behavior of gossiping networks. The advantage is that formal methods are precise and the results are traceable (i.e. performance problems can be traced back to specific design decisions). The disadvantage of formal methods is that they rarely scale. As the size of the system under analysis is increased, the models grow exponentially. Another problem is that a system may be too complex to model using a particular formalism. First, a gossiping network is inherently dynamic, because nodes may enter or leave the system, and their connections vary over time. Furthermore, gossiping network models combine concurrency and probabilistic behavior in a timed setting, which leads to modeling and analysis complications.

In this paper, we will use formal methods (in the form of explicit state model-checking) to analyze gossiping networks. Our main goal is to experiment with precise, explicit-state, formal models and to investigate the potential and the limitations of this approach. In particular, we want to answer the following questions:

- Is it possible to model the complex nature of gossiping networks using formal methods?
- How well does explicit state model-checking scale?
- Are the – possibly small scale – results useful in making design decisions?

To investigate the first question, we model the peer sampling service of [10] using the μCRL [4] tool-set, which supports the use of complex data-types. The central challenge is to model a dynamically changing network using static data-types. The μCRL specification is then transformed to a labeled continuous-time Markov chain, by combining concurrent, probabilistic and stochastic behavior along the lines of the MLotos process algebra [9]. We then perform analysis (on a normal modern workstation) to see for which size system we can still generate explicit-state models. Finally, we compare our results with the simulation and emulation results from [10] to see if we can detect the same interesting phenomena using formal methods as are observed when employing simulation and emulation.

Obviously, any complete explicit state method can only handle relatively small networks. *Symmetry reduction* is particularly interesting in the setting of gossiping networks, as it abstracts individual node identities and instead looks at the overall structure of the network (in terms of the connections between the nodes). We explore symmetry reduction for gossiping networks by using the Groove tool [15]. This tool utilizes graph transformations and is therefore ideal for the description of the behavior of gossiping and other dynamic networks. Furthermore, since Groove handles graphs modulo isomorphism, it automatically abstracts individual node identities. The results obtained in this way are still complete and precise. However, it is clearly desirable in the future to also use some form of abstraction to counter the state-space explosion problem even more drastically [2].

The paper is organized as follows. Section 2 describes gossiping networks. Section 3 gives an overview of the different formalisms used in this paper. Section 4 describes how we used these formalisms to model gossiping networks. The analysis of the gossiping network models is then explained in Section 5. Then the results of the analysis are given in Section 6. Finally, we discuss the possible avenues for future work in Section 7 before concluding the paper in Section 8.

2. GOSSIPING NETWORKS

One of the primary uses of networks is the distribution of information from and to the constituent nodes. Traditionally, special network nodes, known as servers, are designed to be responsible for this distribution; other nodes are then called clients. The drawback of the client-server approach is that the servers alone are responsible for the proper functioning of the whole network. Therefore, this approach does

not scale well and is unsuitable for very large or dynamic networks with high performability requirements [17].

An elegant alternative was found by abandoning the idea of a central server coordinating the proper functioning [5]. All nodes then behave according to some simple algorithm and, hopefully, the proper network behavior emerges spontaneously without any one node being responsible for the correctness of the entire network. This approach mimics the way a group of people spread gossip. No single person takes it upon him or herself to collect all gossip and distribute it to everyone, yet because people naturally share the gossip they know, it can be expected that in the long run everyone knows everything about everybody. Because of this similarity, these networks are referred to as gossiping networks (or epidemic networks, because the way information is spread throughout the network also mimics the way a disease spreads throughout a population during an epidemic) [5].

In the absence of a central server, the nodes of a network must themselves acquire and maintain knowledge of the structure of the network. This is the responsibility of the so-called peer sampling service. The idea is that the nodes continuously exchange information about the nodes they know about. The goal of this behavior is to maintain a well-balanced network as this greatly improves the reliability and efficiency of the network. In [10] it is assumed that each node knows only a small number of its peers (the set of peers known to a node is known as its view, which has a maximum size). The active behavior of a node is then as follows:

1. It selects a peer from its view;
2. It selects what part of its view it will send;
3. It sends this subview and receives a subview in return;
4. It merges the received subview with the original view;
5. It prunes excess peers from its view, if necessary.

There are several parameters in this protocol:

- The communication policy (step 3): *push*, *pull* or *both* (push-pull). This refers to cases where, respectively, only the active node sends its view, only the passive node sends its view, or both nodes send their views. In this paper we study the differences between these policies.
- The selection of peers to communicate, view to send and peers to prune (steps 1, 2 and 5), which can be based on the age of the links in the network (being the time since the last communication between the two nodes). In this paper we ignore all age parameters: peer selection and pruning are done at random (with an equal probability for each possible choice), and always the entire view is sent.

Gossiping networks are difficult to analyze due to their size and the many different parameters. Furthermore, we cannot analyze the nodes in isolation (a technique which is useful in analyzing client-server systems) as we are specifically interested in behavior that emerges in (large) networks of nodes. So far, mostly simulation and emulation have been used [10], but this has a number of drawbacks. Simulation relies heavily on the accuracy of the simulation models used and can only give results in the form of confidence intervals. Emulation on the other hand is very costly and the precise interpretation of the results is often obscure, i.e. when something interesting happens it is difficult to find out what caused this event. Finally, both simulation and emulation struggle to find so-called rare events, i.e. events that have a very low probability to happen (such that they rarely happen in simulation/emulation), but are still common enough to cause great problems during the operation of the network.

As a first start we study a simple version of the gossiping protocol as described in [10] where peer selection and view selection are always random. Methods to implement other peer selection and view selection strategies are discussed in Section 7.

3. FORMALISMS

In this section, we describe the formalisms used in the modeling and analysis of gossiping networks. For the sake of brevity we keep the descriptions short and refer to other sources for more detailed information about the formalisms. Figure 1 shows how these formalisms have been chained together for the purpose of this paper.

3.1 mCRL

μCRL [4] combines process algebra (in the style of the algebra of communicating processes, ACP [3]) with abstract data types. From process algebra, it inherits operators like $+$ (alternative choice), $\cdot$ (sequential composition) and $||$ (parallel composition). Normally, parallel processes interleave their actions in an asynchronous way. When specified explicitly, parallel processes can synchronize on specific actions.

The data part is used to model the state of a recursive process ($X(s) = p[X(s')]$), conditional branching ($p \lhd b \rhd q$) and to describe the data communicated by synchronized actions ($send(m)$). The possibly infinite summation ($\sum_{x:N} read(x)$) is used to model the input of an arbitrary $x : N$, where N is a possibly infinite set of values.

3.2 Groove

Groove [15] is a tool for the verification of graph transformation systems. A Groove specification is a set of graph transformation rules, each of which consists of a left hand side (LHS) and a right hand side (RHS). The effect of a rule is given by the "difference" between LHS and RHS; in particular, nodes and edges can be added or removed. A rule is applicable to a graph wherever the graph contains an image of the LHS; applying the rule essentially means replacing the LHS image by a copy of the RHS.

Given a rule system and an initial graph, a model of the behavior is obtained by exploring all rule applications recursively to the initial graph and all resulting new graphs. This gives rise to a transition system in which the states are graphs and the transitions are rule applications. Hence, to model the behavior of a given system, all relevant information, including the data structures, should be encoded into the initial graph, by means of nodes and edges, and all dynamic steps should be encoded as graph transformation rules.

A special feature is that states are collapsed modulo graph isomorphism; in other words, Groove performs automatic symmetry reduction (see [16]). This turns out to be of great advantage in for the gossip protocol, since this contains a very large degree of symmetry.

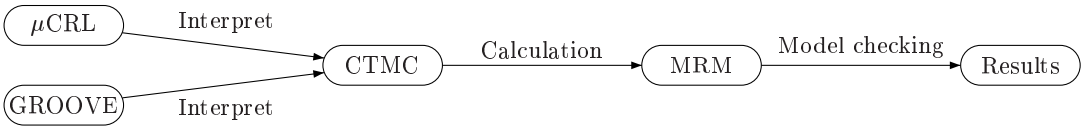


Figure 1: The analysis trajectory.

3.3 Continuous-time Markov chains

Continuous-time Markov chains are a class of stochastic processes with a discrete state space, where state transitions occur after time-delays governed by negative exponential distributions (for an overview of CTMCs and other Markovian models see [8]). A CTMC can be embellished with a labeling function which labels each state with a set of logical propositions. We call the resulting model a labeled CTMC. In our case, the states of the CTMC are labeled with directed graphs representing the state of the gossiping networks, but it is obvious that a directed graph of bounded size can be encoded as a set of propositions.

3.4 Markov reward models

A Markov reward model is a CTMC augmented with a *reward structure* assigning a real-valued reward to each state in the model [1]. We use this reward structure to measure several quality indicators of the gossiping networks: the variance of the indegree of the nodes, the average length of the shortest path between every possible combination of nodes and the clustering coefficient (see [10] and Section 5).

We are interested in calculating the expected value of these measures at certain time-points as well as the expected value of the measures in the long run. We can calculate this by implementing the *possible extension to CSRL* first mentioned in [1] and implemented in [11]. The *instantaneous reward* corresponds to the expected value of a measure at a certain time point. The instantaneous reward at time point t is calculated by summing up, for all states s, the product of the probability of being in s at time-point t (transient probability) and the reward of s. The *expected reward rate* corresponds to the long-run expected value of a measure. The expected reward rate can be calculated by summing up, for all states s, the product of the long-run average probability (steady-state probability) of being in state s and the reward of s.

4. MODELING

In this section, we describe how we modeled gossiping networks. First, an abstract overview of the behavior of a node in a gossiping network is provided. Next, the associated μCRL specification is given. Finally, we describe how we modeled the gossiping networks using the graph transformation tool Groove.

4.1 Abstract model

The state of one node in our gossiping network is described by its view, i.e. the other nodes it knows about, and its internal state. Such a view is modeled simply as a set of nodes. The behavior of a node is divided into an active and a passive "thread", following [10]. A schematic representation of the different internal states of a node using the *push* policy can be seen in Figure 2.

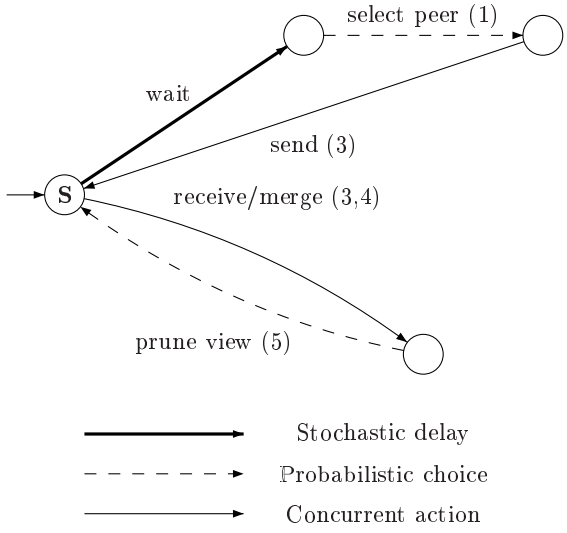


Figure 2: Schematic of the behavior of a gossiping network node using the *push* policy.

Initially, a node is in its stable state (marked **S** in Figure 2). After a stochastic delay (the *wait* transition in Figure 2) the node may move to its active thread. At this point the protocol described in Section 2 starts: in its active thread the node randomly selects a peer (with equal probability, step 1), sends its view (augmented with its own identity) to the selected peer (step 3) and returns to its stable state. The selected peer receives this view in its passive thread, provided it is in a stable state, and merges it with its own view (steps 3 and 4); it then prunes the merged view randomly to a correctly sized subset (with equal probability, step 5). After this view selection the node returns again to its stable state.

The *pull* policy is similar, except that here the active thread, after selecting a peer, requests the view of that peer, merges it with its own view, and truncates it randomly. Finally, in the *push-pull* behavior, views are exchanged in both directions.

A full network consists of N such nodes, working in parallel. It is important to understand that if all nodes are in a stable state, any node could start the active thread, and select potentially any other node. So for an N node network there are $N(N-1)$ potential continuations (limited only by the actual contents of the views).

A major issue in any concurrent setting is how the events of different nodes are ordered. In [10] a round-robin schedule is assumed: in every round, every node acts exactly once. However, such an ordering would require a central authority (at least a global clock), which makes sure that each node acts at the appropriate time. But the lack of a cen-

tral authority is one of the principal properties of gossiping networks so we find this assumption too restrictive. In this paper we assume that all nodes act after a stochastically distributed delay. The delay distributions of the nodes are identical, but independent. This means that the nodes are all *expected* to act at the same rate, but the independence means that there is no need for a central authority. In this model rare occurrences, such as a single node acting much faster than the other nodes for a period of time, are possible even though they will have an extremely small probability. Such rare occurrences are generally difficult to detect using standard simulation or emulation techniques.

There could be concern that a model composed of several nodes might deadlock. Specifically, this would happen if two nodes would simultaneously enter their active threads and attempt to communicate with each other. Both nodes would then be stuck waiting for the other node. To avoid such situations, the active and passive threads must somehow run atomically. This can be modeled by the maximal progress assumption [14], i.e. all internal behavior occurs immediately. In practice, this means that all communication and view-updating actions have priority over the stochastic delay. This can also be explained stochastically: Since the *Wait* delays are drawn from continuous distributions the probability that two timers expire at the same time is zero. If internal computation times are neglected, the probability that another timer expires during internal computation is also zero. Hence we may safely assume that the passive threads are always ready to receive information.

The stochastic delay *Wait* is assumed to be governed by a negative exponential distribution and is thus modeled as a continuous-time Markovian transition. In reality, however, the delay could be implemented as a deterministic delay. This can be approximated using an Erlang distribution. Such an Erlang distribution would consist in our model of a chain of identically distributed exponential distributions, i.e. a chain of Markovian transitions. To improve the accuracy of the approximation we need to increase the number of phases in the Erlang distributions, i.e. we must make the chain longer. This, however, exponentially increases the size of the network model. We have not experimented with this in our analysis.

4.2 mCRL

Using the μCRL language, we modeled each node as a separate process. The state parameters of each node denote its identity and its current view. Nodes are composed in parallel, and communicate by sending/receiving views. For this, we introduce explicit `send` and `receive` actions, which synchronize atomically (handshaking). Complex operations, like merging views and selecting subviews, are specified by equations in the abstract data part.

In order to model one exchange (including pushing and pulling views) in the protocol atomically, we specify synchronized `send`- and `recv`-actions with four arguments as follows:

`send`(i, j, v, w) denotes that (the active thread of) node i pushes view v to (the passive thread of) node j, and pulls view w from it.

`recv`(i, j, v, w) denotes that (the passive thread of) node j receives view v from (the active thread of) node i, and sends view w to it.

In order to model non-deterministic strategies for peer selection and view selection, we include two predicates:

`peerselect`(v, p) : given current view v, it is possible to select p from it for the next communication

`viewselect`(v, u) : given a view v, it is possible to select the subview u from it.

Given all these ingredients, a node with identity i and current view v, and having two threads, can essentially be modeled as follows:

$$
\begin{aligned}
&Node(i : Id, v : View) = \\
&\quad \textstyle\sum_{j:Id} \sum_{w:View} \sum_{u:View} send(i, j, v, w) \cdot Node(i, u) \\
&\qquad \triangleleft\ peerselect(v, j) \wedge viewselect(merge(v, w), u)\ \triangleright \delta \\
&+ \quad \textstyle\sum_{j:Id} \sum_{w:View} \sum_{u:View} recv(j, i, w, v) \cdot Node(i, u) \\
&\qquad \triangleleft\ viewselect(merge(v, w), u)\ \triangleright \delta
\end{aligned}
$$

A network with three nodes and node 2 in the center is then modeled as:

$$Node(1, \{2\}) \parallel Node(2, \{1, 3\}) \parallel Node(3, \{2\})$$

In fact, we used a slightly more complicated model: a delay action is added; the peer select and view select transitions are explicitly modeled as internal transitions; node i is properly added to v and deleted from w; all datatypes, including the selection predicates, must be specified in full detail. The actual model that we used is parameterized over the pull/push policy, the sizes of the network and the view, and over the initial configuration. We were also able to specify peer and view selection strategies based on hop counters, but these models have not been analyzed in detail.

Note that we relied on the strong data specification capabilities of μCRL. However, μCRL has no notion of probabilistic choice, or stochastic time. So, as one can see above the choice of peer selection and view selection are modeled as non-deterministic choice in μCRL. In order to model the delays, the send-action is preceded by an action "delay". Only after generating the state space, the other tools in the tool chain interpret "delay" as stochastic delay. Also, they interpret non-deterministic as equiprobable choice.

The behavior of the gossiping network is now defined as the parallel composition of the behaviors of its constituent nodes. The maximal progress assumption is implemented by giving all other transitions priority over the delay action. The state space of this network basically consists of the views of all nodes. If we interpret the peers in the view of a node as its neighbors in a directed graph, then each state in the behavior of the network is labeled by a directed graph. In Section 5, we will see how we transform this behavior to a Markov reward model and how we then analyze it to compute interesting measures for the network.

4.3 Groove

The Groove model of the gossiping network directly encodes the structure of the network as a graph, with network nodes as graph vertices and their view as a set of outgoing edges. In addition, the model includes some auxiliary vertices and edges to control the behavior. An example initial graph, for a network of size 5 with initial view size 2 organized in a ring structure is given in Fig. 3.

The Groove model does not incorporate the notion of communicating processes. Instead, the essential steps of pushing

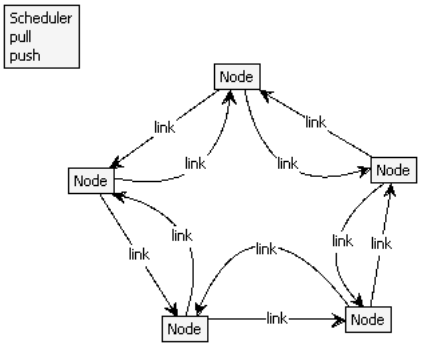
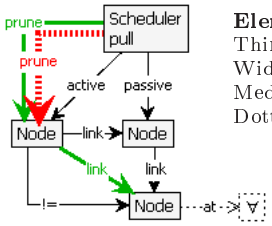


Figure 3: Start graph for the Groove model



Elements	Meaning
Thin black	Present in the graph
Wide dashed	Absent in the graph
Medium gray	Added to the graph
Dotted	Universally quantified

Figure 4: Rule "pull": link edges are added to the active node for all links known to the passive node.

and pulling the views from one node to another are each captured by a single rule, which incorporates at the same time the role of the active and the passive node. For instance, the rule for pulling is displayed in Fig. 4.

Together with rules for choosing the active and passive nodes and for "cleaning up" afterwards, this forms a small protocol like the one displayed in Fig. 2 for mCRL, with as main difference that there are no separate "send" and "receive" actions; rather, these are combined in the "pull" and "push" rules.

5. ANALYSIS

In this section, we describe how we analyze the μCRL and Groove models described in the previous section. This analysis follows the trajectory of Figure 1. We also discuss the complexity of our approach, both in terms of the size of the models and the time needed to analyze the models.

5.1 From mCRL/Groove to CTMC

In Section 4, we have seen that the μCRL and Groove models contain continuous stochastic delays and discrete probabilistic transitions. Following the strategy for the MLotos process algebra [9] we interpret the μCRL and Groove models as labeled CTMCs.

Let's first consider what a μCRL or Groove model of a gossiping network looks like. The μCRL model is generated by composing all the node models in parallel, while the Groove model is generated by exhaustively applying all graph transformations. The choice of the node that will instigate a communication is modeled as a choice between stochastic transitions. After a node X has been selected, the choice in step 1 of the protocol (the peer selection) is a discrete probabilistic choice between the nodes in the view of X. It is important to note that probabilistic choices take place instantaneously and, because of the maximal progress assumption, this prevents any other node from becoming active (i.e. finishing its stochastic delay) before node X is done with its communication. The peer selection is followed by another probabilistic choice of the result of step 5 (pruning). After this, the model returns to a new stable state, where all nodes are waiting on their stochastic delays. A partial example of a model with a single pruning choice can be seen on the left side of Figure 5.

Since all internal transitions are substituted by probabilistic choice, there is no internal non-determinism left. We also see that all probabilistic transitions are *delay-guarded*[1]. This means that the models can be transformed into CTMCs as in [9]. The main principle of this transformation is that a Markovian delay (e.g. with rate λ) followed by a probabilistic choice (e.g. between two transitions, one having probability $\frac{1}{3}$, the other having probability $\frac{2}{3}$) is stochastically equivalent to a choice between Markovian transitions such that the rate of the original Markovian transition is distributed over the new Markovian transitions according to the probabilistic choice (in our example we get Markovian transitions with rates $\frac{1}{3}\lambda$ and $\frac{2}{3}\lambda$ respectively). The state-labels of the μCRL and Groove models are preserved in the resulting labeled CTMCs. Each label describes a configuration of the gossiping network.

In practice the transformation from μCRL or Groove model to CTMC means that every sequence of wait (stochastic delay), peer select (probabilistic choice) and view select (probabilistic choice) transitions is replaced with a group of stochastic delay transitions by distributing the stochastic delay of the wait transition over the probabilistic distributions of subsequent transitions. A partial example of this transformation can be seen in Figure 5.

5.2 From CTMC to MRM

We now have a labeled CTMC with each of its states labeled with a directed graph representing the state of the gossiping network. We now compute for each state in the CTMC, using standard algorithms from graph theory, several measures of the graph associated with the state: the variance of the indegree of each of the nodes, the average shortest path length between all combinations of different nodes and the clustering coefficient [10]. This gives us three MRMs where the reward structure ρ is the indegree variance, average shortest path or clustering coefficient respectively. The indegree variance is a measure on the distribution of indegrees in the network. In a perfectly balanced network all indegrees would be equal and the variance therefore 0. The higher the variance the more unbalanced the network is, which is undesirable. A low average shortest path length is desirable since this will reduce transmission times. And finally the clustering coefficient measures the amount of interconnections between the neighbors of any node. High values for this coefficient mean that the nodes form clusters which unbalances the network and is therefore undesirable.

5.3 From MRM to results

Obtaining the results consists of two steps. First, using the transient and steady state analysis tools from the CADP toolset [7], we compute the probability to reside in each state

[1] A delay-guarded probabilistic transition is (eventually) preceded by a stochastic transition. See [9] for more details

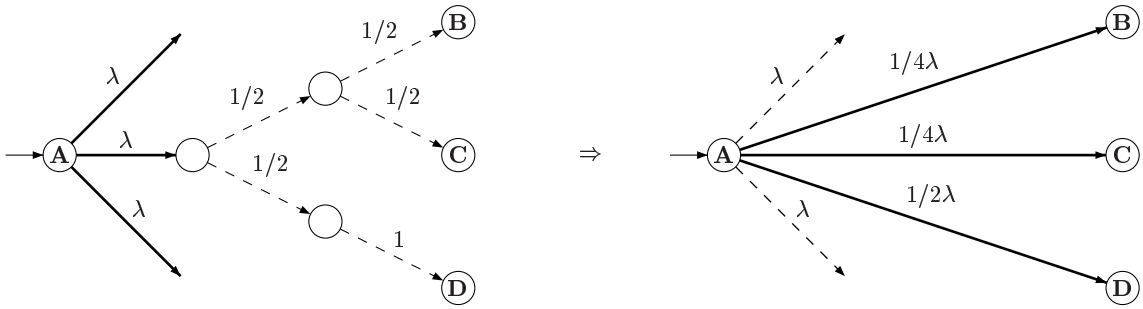


Figure 5: Example of the transformation of part of a μCRL/Groove model to a CTMC.

at certain time points and on the long run. This can be done once for the CTMC obtained after removing all labels. The intermediate result is lumped, using rate-preserving branching bisimulation minimalization.

Next, we model-check each MRM separately, using the extension to CSRL first suggested in [1]. We have implemented this extension using the extensible XTL model-checker of the CADP tool-set [12]. The CSRL extension is also supported by the PRISM model checker [11]. This extension to CSRL provides us with both *instantaneous rewards*, i.e. the expected value of one of the measures at some time point, as well as the *long run reward rate*, i.e. the expected average value of the measures in the long run.

5.4 Complexity

We now consider the complexity of our analysis method. We first notice that the state-space of the models, from μCRL or Groove to MRM, is bounded by the different possible network configurations (times a constant factor because of the internal states), taken modulo isomorphism in the case of Groove. For a gossiping network with N nodes and view-size (or constant out-degree) C we find $\left(\begin{array}{c} N-1 \\ C \end{array}\right)^N$ different configurations: each node has C out of $N-1$ peers in its view ($\left(\begin{array}{c} N-1 \\ C \end{array}\right)$ possibilities) and there are of course N different nodes. We disregard the possibility of nodes having a view smaller than the maximum view-size since we consider only models where all nodes start with maximum capacity views. Now each state is labeled with a directed graph representing the network. To calculate the graph measures we need to compute the shortest path length for all combinations of nodes. This is done by using Dijkstra's shortest path algorithm which has a complexity of N^2. Calculating the other two measures costs less time. For meaningful values of N this calculation is dominated however by the need to calculate steady-state results for the resulting MRM. The complexity of this operation is x^3 where x is the number of states in the model[2]. Overall we then find a time complexity of $O(\left(\left(\begin{array}{c} N-1 \\ C \end{array}\right)^N\right)^3)$.

For the case of Groove, due to symmetry reduction the state space is (much) smaller, but we know of no analytical way to predict the effective reduction. Note, however, that every configuration of a network of size N, interpreted up to isomorphism, can represent at most $N!$ different "plain" configurations. This provides an *upper* bound to the degree of symmetry reduction. In Table 1 we compare the calculated number of "plain" configurations (P) with the simulated number of configurations modulo symmetry (S), insofar we have been able to compute the latter. The reduction (P/S) is clearly large (in fact, the reader can check that it approaches the maximal reduction of $N!$ to more than 95%), but equally clearly, the size of the reduced state space is still more than linear exponential in the network size, and so the problem is intractable even for small network sizes.

[2]We disregard here the possibility of iterative algorithms, for which the complexity depends on the desired accuracy.

6. RESULTS

In this section we give the results of our analysis. We start by giving the long-run averages for indegree variance (IV), average shortest path length (PL) and clustering coefficient (CC). We then present graphs showing the expected evolution of these measures and compare the results with the conclusion found in [10].

6.1 Long-run averages

Table 2 gives the long run average results for gossiping networks for the three different transmission policies pull, push and pull-push (marked "both" in the table), for different network and view sizes. Moreover, the table also indicates the size of the models in μCRL and Groove. The ratio between these two numbers is similar to the potential reduction predicted in Table 1. Note that for N = 7 we were not able to compute the μCRL models; with Groove we could generate up to network size 7, but the results could not be analyzed. We conclude from these results that, across the board, pull-push is the best transmission policy, followed closely by push, while pull is much worse than the others. Note that this corresponds to the findings of [10].

Another observation is that the number of reachable (stable) network configurations (modulo isomorphism) is almost always equal to the total number of configurations according to Table 1, except for the push policy for N = 6, 7 and C = 2, where apparently a very few configurations are *not* reachable. We have not analyzed this further.

A very interesting set of results emerges for N = 6 and C = 2. For the push and the pull-push strategies we see that on the long run the network will have an indegree variance of 0, a relatively high path length of 3.33 and a clustering coefficient of 1. For pull, a similar effect occurs, but in this case the indegree variance is rather large, instead of 0.

The reason for these values is that (for push and pull-push strategies with N = 6 and C = 2) a gossiping network will always eventually partition into a configuration consisting of two fully connected groups of 3 nodes, shown in Figure 6 (left). This indeed has IV = 0, CC = 1 and average PL = $3\frac{1}{3}$. There is no way for the network to recover from

	$C=2$			$C=3$			$C=4$			$C=5$		
N	Plain (P)	Sym. (S)	P/S	P	S	P/S	P	S	P/S	P	S	P/S
4	81	6	14									
5	7776	79	98	1024	13	79						
6	1.0×10^{6}	1499	667	1.0×10^{6}	1499	667	15625	40	391			
7	1.7×10^{8}	35317	4838	1.3×10^{9}	257290	4975	1.7×10^{8}	35317	4838	279936	100	2799
8	3.8×10^{10}	967255	39103	2.3×10^{12}	–	–	2.3×10^{12}	–	–	3.8×10^{12}	–	–

Table 1: Network configuration counts and symmetry reduction for various network and view sizes.

N	C	Policy	IV	PL	CC	Full state space		Stable
						μCRL	Groove	Groove
4	2	Pull	1.50	1.38	1.00	981	87	6
4	2	Push	1.03	1.16	0.79	945	80	6
4	2	Both	0.94	1.14	0.77	1989	96	6
5	2	Pull	2.93	2.16	1.00	121176	2006	79
5	2	Push	1.51	1.67	0.68	117936	1850	79
5	2	Both	1.53	1.63	0.64	408456	2064	79
5	3	Pull	2.40	1.48	1.00	16144	338	13
5	3	Push	1.15	1.07	0.81	17984	321	13
5	3	Both	1.02	1.05	0.79	39184	419	13
6	2	Pull	4.31	1.00	3.00	1.9×10^{7}	56843	1499
6	2	Push	0.00	1.00	3.33	1.8×10^{7}	56843	1498
6	2	Both	0.00	1.00	3.33	8.2×10^{7}	64389	1499
6	3	Pull	4.75	2.28	1.00	2.4×10^{7}	56843	1499
6	3	Push	2.02	1.39	0.70	2.3×10^{7}	56843	1499
6	3	Both	1.83	1.35	0.67	9.5×10^{7}	64389	1499
6	4	Pull	3.33	1.56	1.00	403075	1307	40
6	4	Push	1.15	1.02	0.83	386125	1247	40
6	4	Both	0.99	1.01	0.82	858475	1604	40
7	2	Pull	–	–	–	–	1515526	35317
7	2	Push	–	–	–	–	1405080	35314
7	2	Both	–	–	–	–	1429880	35317

Table 2: Long run average results for gossiping networks with N nodes and view size C; IV = Indegree Variance, PL = average shortest Path Length, and CC = Clustering Coefficient. Additionally the size (number of stable states) of the μCRL and Groove models is given.

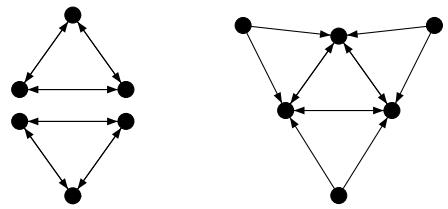


Figure 6: Degenerate network configurations: none of the strategies can recover from the left hand side, and pull cannot recover from the right hand side.

this situation. We expect to see a long-run partitioning for any gossiping network where $N \geq 2(C+1)$. However, the transient analysis will show that it usually takes a long time for a network to partition. For the pull policy, the right hand configuration of Figure 6 (which is a star topology in terms of [10]), together with other star configurations, form a similar "trap", but this time with a very high indegree variance (IV = 4.5 for the configuration shown).

6.2 Transient results

Figures 7-11 show the evolution of the values of the different measures over time. A single time unit corresponds to the expected time a node will take to execute its active thread once.

From Figures 7-9 we can see that the networks of size 5 stabilize fairly quickly. Figures 10 and 11 look at the behavior of networks of varying view size and network size, respectively, under the "winning" pull-push strategy. Here, we can see that the shape of the function for the network of size 6 with view size 2 is different from the others: the indegree variance of this network (depicted in Figure 10) first seemingly stabilizes, but then slowly drops towards zero. For the clustering coefficient we see the same effect: at first it appears to stabilize before it rises to 1 (as shown by the steady-state analysis). Both effects are due to the fact that the network will eventually reach, with probability 1, the configuration of Figure 6 (left). It is also clear, however, that on average it takes a relatively long time for gossiping networks to reach this degenerate state.

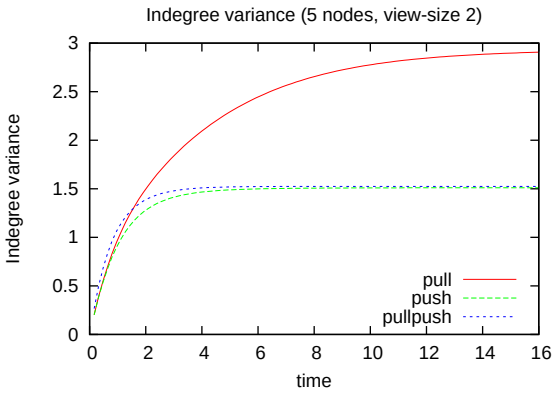


Figure 7: Indegree variance graph for a 5-node network with view-size 2.

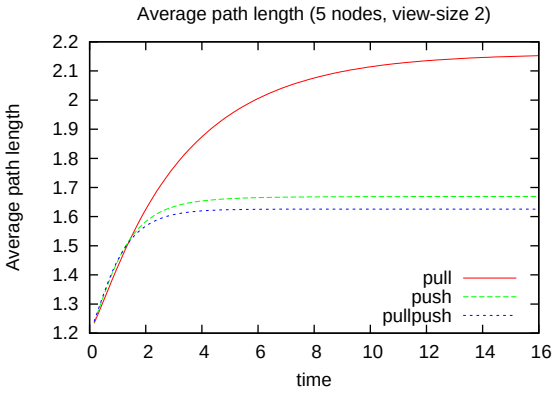


Figure 8: Average shortest path length graph for a 5-node network with view-size 2.

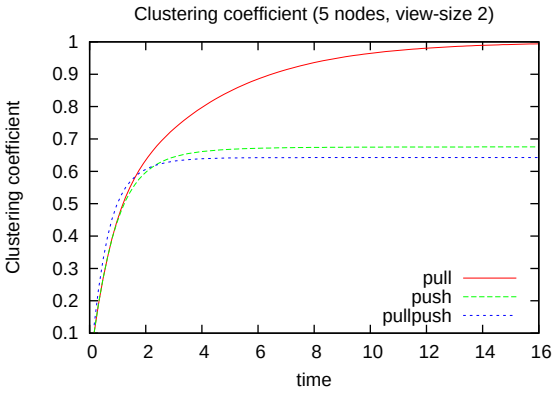


Figure 9: Clustering coefficient graph for a 5-node network with view-size 2.

6.3 Traceability

An interesting aspect of formal methods is that they are traceable. This means that when we find a model which behaves in a specific way we can ascertain why it behaves in such a way. We take as an example the pull policy for gossiping protocols. In [10] it is reasoned that this is a poor policy since such gossiping networks revert to a star topology. This happens when a node has no more incoming links. No other node then connects to it, so no one can *pull* the identity of this node. In other words, a new link to the node cannot be established and the node will forever have no incoming links. With a push policy this is not the case, as a node will push its own identity to other nodes in the network. Figure 12 (left) is the MRM generated in the analysis of a Groove model of a network with N = 4 and C = 2 using the pull policy. We can clearly see the detrimental behavior of the pull protocol. When the network reaches the left-most state it can never leave it again. The "star" topology here is formed by the 3 totally connected nodes (the center of the star) and the upper-left node with no incoming links (the single point of the star). In contrast, the MRM model of a 4-node network using the push policy, depicted on the right hand side, does not show any sink states and does not converge to a star topology.

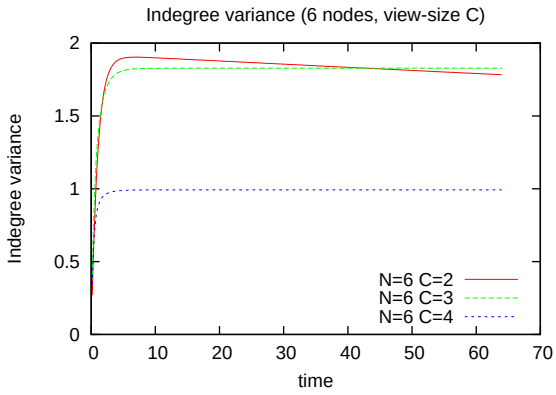


Figure 10: Indegree variance graph for networks with varying view-size with pull-push strategy.

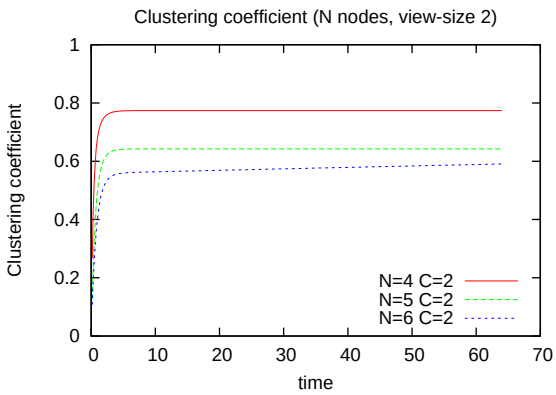


Figure 11: Clustering coefficient graph for networks with varying sizes with pull-push strategy.

7. FUTURE WORK

Since this paper is meant as a first exploration of the practicality of using formal methods to analyze gossiping networks there is a lot of room for further research.

The use of stochastic delays and discrete probabilistic choice has not yet been formally incorporated in the μCRL and Groove formalisms. Based on our experiences in mod-

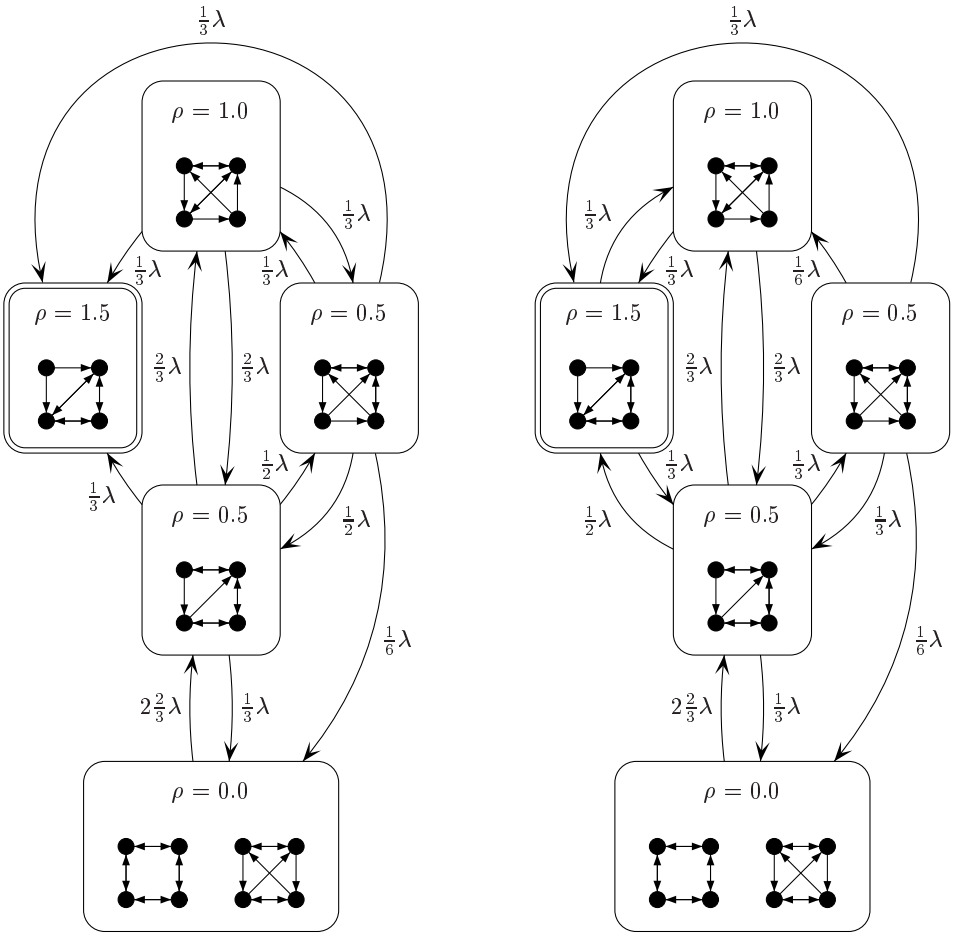


Figure 12: Lumped MRMs of a gossiping network with 4 nodes, view size 2 and *pull* (left-hand side) and *push* (right-hand side) policy. The transmission rate of each node is λ. The reward structure represents the indegree variance of the gossiping network. The states are labeled with graph configurations (The bottom state in fact represents two states with bisimilar behaviors but different graph configurations).

eling gossiping networks we do not foresee any major theoretical difficulties in incorporating stochastics in μCRL and Groove. It is important to develop this theory further as this will also allow other interesting stochastic systems besides gossiping networks to be modeled using these powerful formalisms.

The design space for gossiping networks is quite large. In [10], various strategies for peer selection, view selection as well as the different peer-exchange policies have been studied. In the future, we plan to model and analyze gossiping networks with peer and view selection strategies other than purely random ones. This requires us to model the age of the links in the gossiping networks. This can be easily modeled using the complex data types of μCRL, where the selection strategies are parameterized by a predicate (cf. Section 4.2). However, our initial experiments soon presented a new challenge, as the age of the links may be unbounded, leading to infinite models. A first approach to dealing with this problem would be to investigate the mechanism used in actual implementations of gossiping networks. Another logical approach is to use some form of abstraction to model the age of the links.

Another aspect of gossiping networks that is very important to investigate in the future is dynamically appearing and disappearing nodes as discussed in [10]. The modeling of systems that can grow larger and smaller over time is notoriously difficult with classical process algebras, but special *mobile* formalisms exist, such as the π-calculus (see [13]). In the graph transformation approach of Groove, on the other hand, it should be easy to incorporate this type of behavior.

Regarding the type of analysis we have done, with hindsight we can observe that the long-run values do not give interesting measures. As discussed in Section 6, we conjecture that real-life networks, whose size far exceeds the view size, will always tend to partition, giving rise to atypical long-run averages. It is more interesting to investigate questions of the type "how long will it take until the network partitions with a probability of x", where the desired probability x is a parameter. Our method in principle allows to answer this type of question.

As expected, we conclude that scalability is a real problem when formally modeling gossiping networks. Modeling net-

works with more that six nodes turns out to be practically impossible using μCRL. Although using Groove's in-built symmetry reduction allows us to analyze larger networks the size of the models still grows exponentially, limiting the approach to 7 nodes. Furthermore the modeling of more advanced communication protocols with complex peer and view selection strategies would cause the models to become even larger. It is then obvious to look for abstraction techniques to counter this state space explosion. Large networks could be tackled by only modeling a small amount of nodes explicitly and modeling the rest of the nodes as a single entity behaving according to some average expected behavior. The problem of representing the age of links could be handled with a form of predicate abstraction: instead of denoting the ages of the links explicitly the model could simply list the *order* of the ages. Many other abstraction techniques are of course conceivable.

8. CONCLUSION

Gossiping networks can be analyzed using formal methods. The structure of a network can be captured by using the abstract datatypes of μCRL. Alternatively, the changes in the network can be captured by the graph transformations of Groove, which models the network simply as a directed graph. Furthermore, the combination of concurrent, probabilistic and stochastic behavior can be interpreted as a CTMC in the style of the MLotos process algebra (see [9]), although the theory behind the transformation of μCRL and Groove to labeled CTMCs needs further research. The μCRL and Groove models can then be interpreted as a CTMC labeled with network structures. By calculating interesting graph-measures for these network structures we then obtain MRM models which can be analyzed using an extension to CSRL (see [1, 11]).

It was particularly interesting for us to observe the deviation in the results that occurs for networks of size 6, with view size 2, because we had not predicted or expected this. The explanation of this phenomenon, viz. that on the long run, networks with a certain ratio of size to view size tend to partition, implies that other types of analysis may be called for.

Much research remains to be done in this area (see Section 7). It is desirable, but also challenging, to model more advanced gossiping protocols. Studying larger networks, by means of some form of abstraction is also a promising avenue of research. Simply abstracting from node identities (thus only considering the shape of a network) by using symmetry reduction with Groove already provided great reductions in state space size, but not sufficient for scalability.

The main drawback to the precise explicit approach is the lack of scalability. In practice, we were only able to generate models of up to 6 nodes using μCRL or up to 7 nodes using Groove. However, the results found for these small models confirm the simulation and emulation results found in [10], suggesting that small-scale analysis can lead to insights in the behavior of large-scale networks. Furthermore, the traceability of the models can give a deeper understanding of the emergent behavior of a gossiping network.

9. REFERENCES

[1] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. On the logical characterisation of performability properties. In *ICALP*, volume 1853 of *LNCS*, pages 780–792, 2000.

[2] R. Bakhshi, F. Bonnet, W. Fokkink, and B. Haverkort. Formal analysis techniques for gossiping protocols. *ACM SIGOPS Oper. Syst. Rev.*, 41(5):28–36, 2007.

[3] J. Bergstra and J. Klop. Process algebra for synchronous communication. *Information and Control*, 60(1):109–137, 1984.

[4] S. Blom, W. Fokkink, J. F. Groote, I. van Langevelde, B. Lisser, and J. van de Pol. μCRL: A toolset for analysing algebraic specifications. In *CAV*, volume 2102 of *LNCS*, pages 250–254, 2001.

[5] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. C. Swinehart, and D. B. Terry. Epidemic algorithms for replicated database maintenance. *ACM SIGOPS Oper. Syst. Rev.*, 22(1):8–32, 1988.

[6] P. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulieacute. Epidemic information dissemination in distributed systems. *IEEE Computer*, 37(5):60–67, 2004.

[7] H. Garavel and H. Hermanns. On combining functional verification and performance evaluation using CADP. In L.-H. Eriksson and P. Lindsay, editors, *FME*, volume 2391 of *LNCS*, pages 410–429. Springer, 2002.

[8] B. Haverkort. Markovian models for performance and dependability evaluation. In *Euro Summer School on Trends in Computer Science*, volume 2090 of *LNCS*, pages 38–83, 2000.

[9] H. Hermanns and M. Rettelbach. Towards a superset of basic LOTOS for performance prediction. In *PAPM*, 1996.

[10] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip-based peer sampling. *ACM Trans. Comput. Syst.*, 25(3):8, 2007.

[11] M. Kwiatkowska, G. Norman, and D. Parker. Stochastic model checking. In *SFM-07:PE*, volume 4486 of *LNCS (Tutorial Volume)*, pages 220–270. Springer, 2007.

[12] R. Mateescu and H. Garavel. XTL: A meta-language and tool for temporal logic model-checking. In *STTT*, volume NS-98-4 of *BRICS Notes Series*, 1998.

[13] R. Milner. *Communicating and Mobile Systems; The Pi Calculus*. Cambridge University Press, 1999.

[14] X. Nicollin and J. Sifakis. An overview and synthesis on timed process algebras. In *CAV*, volume 575 of *LNCS*, pages 376–398, 1991.

[15] A. Rensink. The GROOVE simulator: A tool for state space generation. In *AGTIVE*, volume 3062 of *LNCS*, pages 479–485, 2004.

[16] A. Rensink. Isomorphism checking in GROOVE. In A. Zündorf and D. Varró, editors, *Graph-Based Tools (GraBaTs), Natal, Brazil*, volume 1 of *Electronic Communications of the EASST*, September 2007.

[17] W. Vogels, R. van Renesse, and K. Birman. The power of epidemics: robust communication for large-scale distributed systems. *ACM SIGCOMM Comput. Commun. Rev.*, 33(1):131–135, 2003.