

On the Impact of Modelling Choices for Distributed Information Spread

– A Comparative Study –

Rena Bakhshi

Department of Computer Science
Vrije Universiteit Amsterdam
Amsterdam, Netherlands

Ansgar Fehnker

National ICT Australia*
Locked Bag 6016, University of New South Wales
Sydney NSW 1466, Australia

Abstract—We consider a distributed shuffling algorithm for sharing data in a distributed network. Nodes executing the algorithm periodically contact each other and exchange data. The behavior of the algorithm is probabilistic in nature; a node chooses a random peer and sends a random subset of its local data. Moreover, the algorithm exhibits nondeterministic behavior; the order in which nodes initiate an exchange is not specified.

For the shuffling algorithm we build several formal models using the probabilistic model checker PRISM. Despite of the well known state-space explosion problem, we were able to model a network of up to 15 nodes. In addition, we implement two equational models in MATLAB, a discrete model and its continuous alternative, as well as the algorithm itself in the peer-to-peer network simulator PeerSim.

By comparing different modelling frameworks, we further explore the impact of modelling choices, such as different scheduling policies and the notion of rounds. The evaluation of distributed protocols, especially gossiping protocols, is difficult and a comparison of different evaluation techniques is greatly desired, since the evaluation techniques vary a lot and are based on different assumptions. The comparison of different models allowed us to discover hidden assumptions, which helps with the interpretation of the obtained results.

I. INTRODUCTION

With increasing use of distributed algorithms in applications such as distributed data bases or wireless sensor networks comes the need for methods to analyze their behavior before deployment in a specific application. Most alternatives to testing the distributed algorithm in a testbed rely to some extent on an analysis of a model of the system; be it analysis by simulation or be it analysis by model checking.

In this paper we compare different models used to analyze a distributed algorithm for sharing data in a distributed network. The algorithm examined in this paper is the *shuffling algorithm*, a *gossip protocol* presented originally in [1]. In [2], the authors analyzed the spread of a distinguished data item, in a distributed database governed by the algorithm. An exchange of data between nodes occurs pairwise; a node selects one node to communicate. If none of the two nodes has

the data item, neither will have it after they exchanged data. If either or both possess the data item, at least one of the nodes will possess it after the exchange. The probability with which nodes possess the item after communication depends on the amount of different data items, exchange message size and local storage size. Moreover, this probability depends on whether both or only one of the nodes possess the item before communication.

In our paper we focus on the property *coverage* [2], that is, the number of nodes that have seen the data item in the past. We adopt the notion of *rounds*, often used in the evaluation of gossip protocols (e.g. [1], [3], [4]), although is not a part of these protocols. In each round every node initiates one exchange of data. This means that it might also be selected by another node as partner in an information exchange. Note that the order in which nodes initiate data exchange is arbitrary. This raises a natural question whether the order in which nodes initiate data exchange can affect the coverage property.

We build formal models for the probabilistic model checker PRISM (version 3.2), implementing different modelling choices, e.g., different scheduling policies. The first model is nondeterministic and covers all possible schedulers, giving best and worst case probabilities. The other models employ a random scheduler, a round-robin scheduler, an "eager" scheduler that gives preference to nodes that possess the item, and finally a "reluctant" scheduler that gives preference to nodes without the item. The latter two models are used to determine if implementable schedulers exist that get close to the best and worst case bounds.

A distributed network often lacks the ability to enforce that all nodes progress at an even pace. However, it is not uncommon (especially for simulation models) to assume that communication occurs in rounds throughout the network. Therefore, we additionally investigate the effect of omitting the notion of round by assuming that all nodes progress with an even chance.

These results are compared with models that are based on the work presented in [2]. We implemented two equational models of coverage in MATLAB – a discrete model as a difference equation and its continuous alternative in the form

*Funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council.

of a differential equation. Moreover, we compared results from an implementation of the shuffling algorithm running on a peer-to-peer network simulator PeerSim [5]. Even for small networks all models produced different results, although the models are based on rigorous formalizations of the algorithm (in the case of PeerSim, on an actual implementation). The comparison with the formal PRISM model allowed us to discover hidden assumptions in these alternative modelling frameworks, which helps with the interpretation of the obtained results.

The network size in each model is restricted to the smallest upper bound that the respective modelling framework could handle. Despite the well known state-space explosion problem, we were able to model the network of 10 nodes (for a round-based model) and of 15 nodes (for a model without rounds) with PRISM. However, this is rather restrictive, when compared to the models with a few thousand nodes presented in [2]. The results for the shuffling algorithm using the differential and difference equations were obtained under the assumption that the network is completely connected. The simulation and the PRISM models use the same assumption, even though both would work without this assumption.

This paper is organized as follows. We first describe the shuffling algorithm as given in [2]. Section III presents the formal PRISM model and explores the influence of different modelling choices. Section IV compares the results for the PRISM models with the results obtained from alternative modelling frameworks. Section V discusses related work, and Section VI concludes with a discussion of the results.

II. THE SHUFFLING PROTOCOL

The shuffling protocol has been proposed for disseminating information throughout a network of small devices. The behavior of the protocol is probabilistic in nature; a node chooses a random partner to communicate with, and it exchanges a random subset of data items from its local storage (cache). Additionally, the shuffling protocol exhibits nondeterministic behavior in that the order in which the nodes initiate a gossip is not determined.

A random gossip partner is provided by an underlying layer that maintains the neighborhood information. This information could be determined by the connectivity between nodes in a wireless environment, or provided by a gossip protocol like [4], executed at each node in a wired environment.

A. Description

The shuffle is performed as an atomic procedure. That is, once a node initiates an exchange with its random peer, these nodes cannot become involved in another exchange until the current one is finished.

Every node maintains a list of data items in a cache of size c . The shuffling protocol is executed at nodes initiating a contact and their contact partners. Both nodes exchange s items from their local caches in one contact; each node agrees to keep the items received from its gossip partner. Taking into account the size of a cache, a node might have to discard some

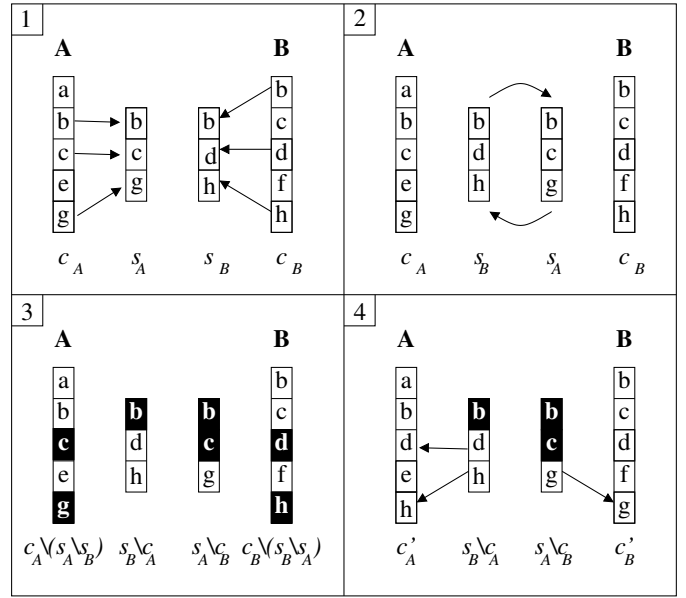


Fig. 1. Illustration of the shuffling algorithm with $n=8$, $c=5$ and $s=3$. (1) Selecting random data. (2) Exchanging data. (3) Determining which data to keep. (4) Merging cache with received data.

items from the cache in favor of the items received during an exchange. The items to be discarded are picked from the ones that have been sent to the peer. Since the peer, in its turn, has agreed to store these items in its cache, discarding items does not lead to the loss of information in the network.

Figure 2 illustrates the shuffling algorithm. The protocol parameters are the following. The size of the caches c_A and c_B is c , the size of the exchanged messages s_A and s_B is s , and the total number of different items in the network is n .

A node A periodically initiates the shuffle with a random neighbor B . It sends B a message s_A , containing a copy of s random items from its cache c_A , and waits for a response from B . Upon receipt of s_B from A , node B sends message s_B with s random items from its cache c_B to A .

Node A and node B then eliminate all received items that are already in their caches from s_B and s_A , respectively. A adds the remaining received items ($s_B \setminus c_A$) to the cache. It adds them by replacing items in its cache c_A that were sent to, but not received from B . This is the set $s_A \setminus s_B$, and its elements may be replaced randomly, while elements in $c_A \setminus (s_A \setminus s_B)$ will be kept. This ensures that not only the total number of items will not exceed c , but also that the items sent from B to A do not get lost. In a similar fashion node B adds the items in $s_A \setminus c_B$ to its cache, by replacing items in s_B that were not in the received message s_A . Figure 2 depicts a pseudo-code for the algorithm. Note that the code for the node that initiates the contact differs from the code of the contacted node.

B. The state transition model

For the purpose of our study, we follow the approach from [2] and distinguish a generic item d in the cache of a node. The performance of the protocol is measured according to the

```

while true do
  wait (some  $t$  time units)
   $B := \text{randomPeer}()$ ;
   $s_A := \text{itemsToSend}(c_A)$ ;
  send  $s_A$  to  $B$ ;
   $s_B := \text{receive}()$ ;
   $c_A := \text{itemsToKeep}(c_A \setminus (s_A \setminus s_B), s_B \setminus c_A)$ ;

```

(a) initiating node

```

while true do
   $s_A := \text{receive}()$ ;
   $s_B := \text{itemsToSend}(c_B)$ ;
  send  $s_B$  to sender( $s_A$ );
   $c_B := \text{itemsToKeep}(c_B \setminus (s_B \setminus s_A), s_A \setminus c_B)$ ;

```

(b) contacted node

Fig. 2. The shuffling protocol algorithm

spread of the data item d in the network. Thus, it is sufficient to use a boolean to indicate the presence (as *true*) or the absence (as *false*) of d in the cache of node.

The shuffling protocol is modelled as a pairwise node interaction. Figure 3 captures all possible scenarios of how data item d is exchanged considering state of nodes caches before and after the interaction of the nodes. There are four possible states of caches of two gossiping nodes, the initiating node A and the contacted node B . Every state is a pair of bits (a, b) for the node A and for the node B , respectively. A transition from the state before an exchange a_1b_1 to the state after the exchange a_2b_2 is labelled by the corresponding conditional probability $P(a_2b_2|a_1b_1)$.

These probabilities, determined in [2], are the following:

$$\begin{aligned}
 P(01|01) &= P(10|10) = \frac{c-s}{c} \\
 P(01|11) &= P(10|11) = \frac{s}{c} \frac{c-s}{c} \frac{n-c}{n-s} \\
 P(10|01) &= P(01|10) = \frac{s}{c} \frac{n-c}{n-s} \\
 P(11|01) &= P(11|10) = \frac{s}{c} \frac{c-s}{n-s} \\
 P(11|11) &= 1 - 2 \frac{s}{c} \frac{c-s}{c} \frac{n-c}{n-s}
 \end{aligned}$$

The state $(0, 0)$ in Figure 3 is not connected with any other

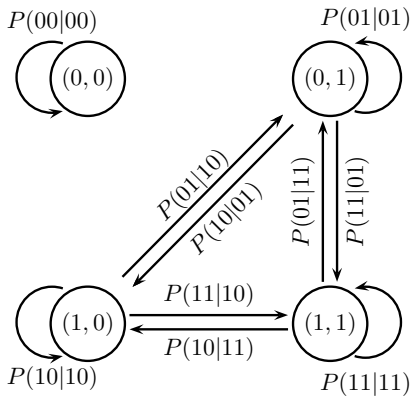


Fig. 3. State transition diagram

state, because such transitions are not possible in the protocol and, thus, the corresponding transition probabilities are equal to 0. We refer to [2] for more details.

III. PRISM MODELS

For formal modelling and analysis of the shuffling algorithm we use the probabilistic model checker PRISM, developed at the University of Oxford [6]. PRISM supports three types of probabilistic models: *Markov decision processes* (MDP), *discrete-time Markov chains* (DTMC), and *continuous-time Markov chains* (CTMC). In this paper we use MDPs and DTMCs.

The PRISM input language is a state-based language for describing probabilistic state transition systems. In a model, a system is regarded as a set of “communicating” modules. Each module consists of a set of guarded commands. A command is composed of the guard (a predicate on the state variables) and a probabilistic update relation (the probabilistic assignment to variables). Modules can synchronize on shared variables or on common actions labels.

Once specified, the PRISM model checker constructs an internal representation of the system model – a Markov-style transition system for the composition of specified modules – which can then be analyzed exhaustively relative to a specified property using a suite of numerical algorithms. The specification language is PCTL, a probabilistic extension of CTL, expressive enough to describe many performance style properties. PRISM computes the probabilities of satisfaction for DTMCs, and the best and worst case probabilities of satisfaction for MDPs. The maximal and minimal probabilities for an MDP refer to the worst and best case resolutions of the non-deterministic choice.

A. The basic PRISM model

All PRISM models described in this section share a similar structure. A node initiates an exchange of data items with a random partner exactly once per round, but may be contacted not at all, once, or multiple times.

For each node i in the network we introduce one global boolean variable ni (whether the node currently possesses data item d) and one module $node_i$. This module has a local variable $senti$, that records whether a node has initiated an exchange in the current round. A node i initiating the contact with a node j is modelled as a local transition of module $node_i$. This transition selects the node j with probability $1/(N-1)$, where N is the number of nodes in the network. It then sets the variable $senti$ to *true*, and updates the state variables ni and nj , using the probabilities as defined in Figure 3. Thus, the entire communication process, including the exchange of data items, is modelled by the probability that the participating nodes acquire the distinguished data item d . Figure 10 in the appendix contains an example PRISM model for 3 nodes. Defining the variables ni and nj as global enables the initiating node to update the state atomically, removes the need for synchronizing transitions between modules, and thus simplifies the model considerably.

Once all modules have initiated a gossip in a round, they perform one synchronous transition on the labelled action done to mark the end of the round. This transition will reset the local variables `senti` back to `false`. Overall, each round in the model has $N + 1$ transitions: for each of the N node one transition to model the initiated contact, and one transition to mark the end of the round.

B. Coverage property in PRISM

The coverage property, analyzed in [2], is expressed as the fraction of nodes that have ‘seen’ item d in the past. This property cannot be expressed directly in PRISM; this tool allows for probabilistic specifications, i.e. it allows to compute the probability that any given node has ‘seen’ a node in the past. Since the network is completely connected and all nodes are symmetric, it suffices to compute the probability for one node. The probability that node i had data item d in the past k rounds is expressed as follows:¹

$$P=?[(\text{true}) \text{U}<=k*(N+1) (\text{ni})] \quad (1)$$

Note that $(N + 1)$ is the number of steps in a round.

C. The PRISM models

We now describe different PRISM models for networks of several nodes. All models assume that the communication channels are reliable, provided by an underlying layer that maintains the neighborhood information.

1) *All Possible Schedules*: The basic model as described above does not specify in which order the nodes initiate communication. We first consider the non-deterministic model which allows all schedulers for communication. The nondeterministic PRISM model of the algorithm is specified as MDP. PRISM then allows to compute the minimum (worst case) and maximum (best case) probabilities of a node to see the item d , over all possible schedulers, until round k .

Example: We consider a network with $N = 10$ nodes, cache size $c = 100$, number of different data items $n = 500$, and the message size $s = 50$. It is assumed that one node, node 1, has the item d initially. Figure 4 depicts the maximal and minimal coverage in detail for the first two rounds, i.e. we are checking $P=?[(\text{true}) \text{U}<=t (\text{ni})]$ rather than (1), for a node i other than node 1.

The results show that during the first two rounds the worst and best case probabilities diverge. Note, that this is the worst case probability for any distinguished node, and not the average probability over all nodes. For example, right before the second last step in round 1 ($t = 8$), the worst case probability of node i will still be 0. This assumes that the other 8 are scheduled first, then, at $t = 9$, node 1, and finally, at $t = 10$, node i . All other nodes will have a positive probability at $t = 8$, since they initiated communication before. The average coverage for these nodes at this point is therefore

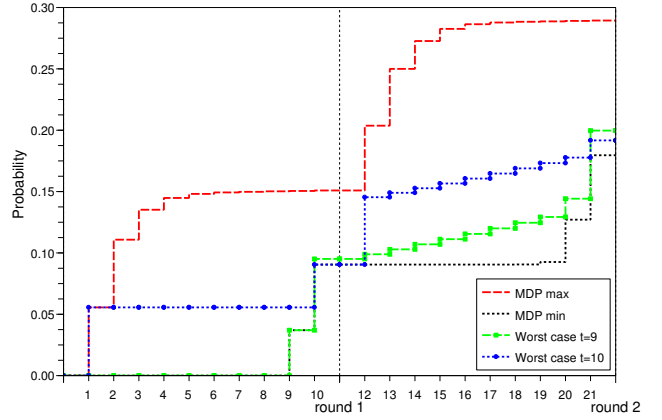


Fig. 4. Worst and best case bounds for MDP model for the initial 2 rounds, and schedules realizing the worst case bounds at $t = 9$ and $t = 10$

strictly greater than 0, while the worst case probability of node i is still 0. There exists no schedule that realizes the lower bound of 0 for all nodes simultaneously.

Furthermore, the worst case bound in Figure 4 is not realized by a single worst case schedule, but by a combination of schedules. The previous paragraph discussed the worst case schedule for node i leading up to the second last step in round 1. The worst case probability for node i at the next step ($t = 10$), when all nodes but one initiated communication, is realized by a schedule in which node i is the first to initiate communication, followed by the other nodes, and then followed by the node 1 at $t = 10$ as the last node in this round. Figure 4 depicts the results for the two different schedules in comparison to the worst case bound for the MDP model. Note also that in the second round, when these schedules are repeated in round-robin fashion, neither of them realizes the worst case bound.

Obviously, similar observations hold for the best case bound, namely that it is the best case probability for any individual node, which will not be realized by any single schedule. For this model we presented the details for the steps within a round, but in the remainder of this paper we only consider the probabilities at the end of each round.

2) *Fair Random Scheduler*: Rather than exploring all non-deterministic choices in the model, we now instantiate them with uniform distribution. In PRISM, this is simply achieved by modelling the system as a DTMC (rather than as a MDP), which effectively implements a fair random scheduler.

3) *Round-Robin Scheduler*: The following model implements a round-robin scheduler to select the next node that initiates communication. The nodes are arranged from node 1 to node N , and a node $i + 1$ can only initiate communication once node i has done so. An exception to this rule is node 1 which can initiate the first communication in each round.

The round-robin scheduler does not reflect a scheduler that would be implemented by an actual network, since it requires a central scheduler with a global view of the network. This would defeat the purpose of the distributed database that

¹For MDP models it is necessary to specify whether to compute the upper or lower bound probability. PRISM provides the PCTL operators $P_{\max}=?$ and $P_{\min}=?$ for this purpose. For simplicity we only use $P=?$ throughout this paper.

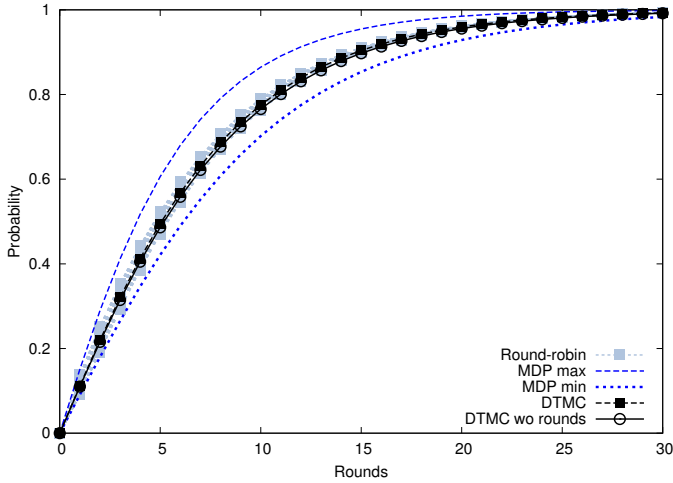


Fig. 5. Coverage for MDP and fair DTMC model.

executes a distributed shuffling algorithm. However, a round-robin scheduler would be a suitable model for any simulator that checks and selects the next node in each round in the same order. In the context of the shuffling algorithm, a round-robin scheduler is nothing but a loop iterating over an array of node IDs. Analysis of models that use a round-robin scheduler enables to estimate the range of coverage which simulators employing a deterministic round-robin scheduler may give.

Example: As before, we consider a network with 10 nodes, cache size $c = 100$, number of different data items $n = 500$, and the size of the message $s = 50$. Node 1 has the distinguished data item d initially. Figure 5 shows the results for the non-deterministic, the fair random, and the round-robin scheduler. For all models the probability that a node has 'seen' the item d in the past will converge towards 1. Also as to be expected, the coverage for the fair random scheduler and the round-robin scheduler lie in-between the upper and lower bounds obtained from the non-deterministic model.

For the round-robin scheduler, coverage depends on when node 1 and node i are scheduled. Selecting nodes in a predetermined order destroys the symmetry between the nodes. The case when node 1 is scheduled first yields a higher coverage for the other nodes, compared to the round-robin scheduler that schedules this node last. Also, if node i precedes in the round-robin order node 1, it has a higher coverage than when it succeeds it.

Another interesting observation is that the average coverage over all round-robin schedulers is not equal to the result of coverage of the fair scheduler. The average coverage for round-robin schedulers is slightly improved with respect to a fair random scheduler.

4) *Model without Rounds:* All previous models assumed that communications is organized in rounds, and that each node initiates communication exactly once in each round. This assumption is motivated by the assumption that all nodes progress at approximately the same speed. However, round-based models do require either that all nodes to have global

knowledge of whether the other nodes initiated exchange in the current round, or a strict underlying time synchronization algorithm. In general, this cannot be assumed for the shuffling algorithm. It is not only difficult to accomplish in large distributed networks, it would also defeat the purpose of a distributed system.

As alternative we built a model without rounds. It selects the next nodes independent of how often they have selected before, with a uniform distribution over all nodes. The model is similar to the fair DTMC model. However, the model neither needs variable `senti` to record whether it has initiated communication in this round, nor the synchronizing transition to mark the end of a round.

Figure 5 also shows the result for this model. To compare the model with the round based models we defined a "round" to be N steps of communicating nodes. It is one step less than for the other models, since the model omits the transition at the end of a round. The results show that the coverage in a model without rounds is smaller. However, they are clearly not only within the bounds given by the MDP, but also within the range of the round-robin schedulers. Even with a pure random scheduler, without rounds, it is guaranteed that the coverage will converge to 1.

Note that this this model without rounds does not capture more intricate timing behavior such as clock drift or jitter. To capture this would require to keep local timing information. While possible, it does, even for networks of very moderate size, quickly result in state spaces that larger than the model checker can handle.

5) *Unfair Schedulers:* The previous results of this section show that the coverage for the fair random and the round-robin schedulers are not close to the best and worst case probabilities of the MDP model. It was already discussed in subsection III-C1 that the worst and best case bounds are for individual nodes only, and that a single probabilistic scheduler that realizes these bounds for all nodes simultaneously may not exist. These bounds provide conservative outer bounds on the coverage. This raises the question, how conservative these

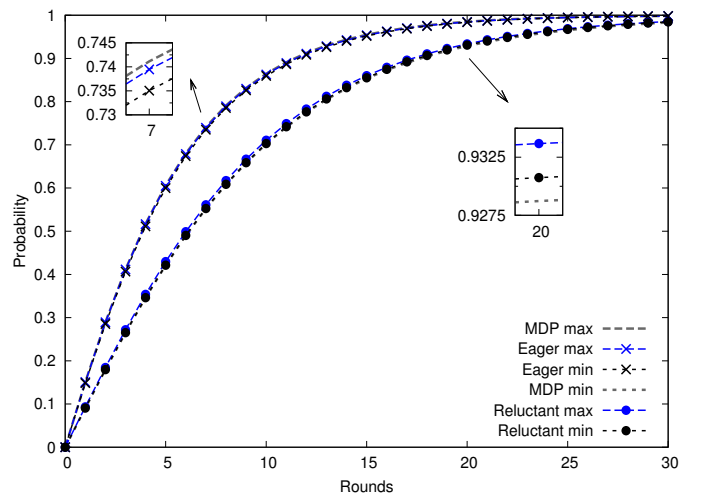


Fig. 6. Coverage for unfair scheduler models, and results for MDP (grey) for comparison. Inset figures show the maximal distance between the results.

bounds are.

To address this question, we studied the behavior of unfair schedulers, that have a global knowledge of the network. These include implementable schedulers, and thus provide an inner approximation on the bounds on the coverage.

The first class are "eager" schedulers that give precedence to nodes that hold the item. Nodes that do not hold the item are blocked from initiating communication as long as there is a nodes with an item that has not initiated communication, yet. The other class are "reluctant" schedulers that give precedence to nodes that do not have d in their cache. Both resulting models remain non-deterministic. Thus, the upper and lower bounds for these classes will constrain the coverage for any implementable scheduler in this class. The lower bound for the "eager" schedulers thus provide an inner approximation on the upper bound for MDP model, while the upper bound for the "reluctant" schedulers gives an inner approximation on the lower bound. Note, that there might exist other implementable schedulers that improve these bounds.

As it is shown in Figure 6, the respective upper and lower bounds for the "eager" schedulers are numerically close to each other. Even though they do not coincide, they cannot be distinguished in this figure. The same holds for the bounds on the "reluctant" schedulers. Figure 6 also shows that the MDP bounds and the inner approximation almost coincide as well. The difference is less than 0.005. This suggests that the that outer bounds are might be somewhat conservative, there exist implementable schedulers that approach these bounds closely.

IV. ALTERNATIVE MODELS

A. Simulation Model

We compared the result for coverage from our PRISM models with coverage observed while running the shuffling protocol in a simulated environment. We deploy the shuffling protocol in a round-based fashion, using the event-based simulator PeerSim [5]. The experiments simulate the case when a new item is introduced at one node in a network, while all caches are full and uniformly populated with 500 items. To ensure this, we let the nodes gossip for 1000 rounds before inserting the distinguished data item.

At start, the simulator creates a random-ordered list of nodes, provided by a configuration file. Along with the network size and the amount of items, other parameters such as a network topology, exchange buffer size are also provided as an input to the simulator. The network topology determines which pairs of nodes can gossip. The experiments are performed on the fully-connected network of 10 nodes. In the beginning of every gossip round, a scheduler in the PeerSim simulator randomly shuffles an order of the nodes in the list. This effectively prevents the emergence of round-robin schedules. Within a round of the protocol simulations, all nodes proceed in a lock-step fashion, executing the protocol in the order decided by the scheduler.

For the experiments we assumed a cache size of $c = 100$. In every round each node randomly selects one of its neighbors and exchanges $s = 50$ items during the interaction. After each

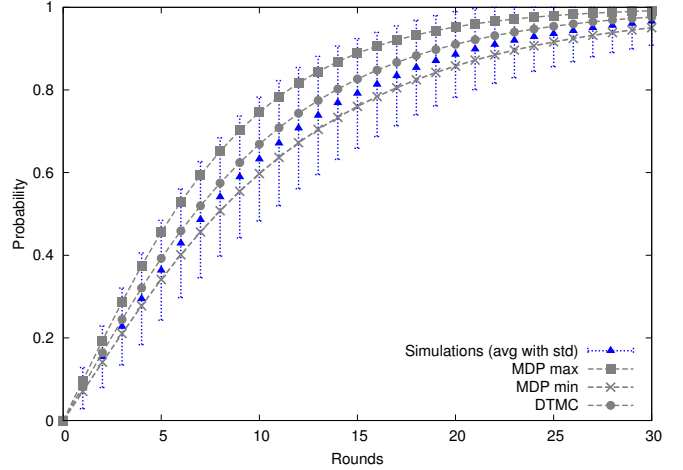


Fig. 7. Comparison of results for PRISM models (grey) and with the results of simulation with standard deviation.

gossiping round, we measure how many nodes in total have seen the distinguished item d at the end of each round.

Note, that the simulator only measures whether a node holds d at the end of the round, and ignores if the node acquired the item, but lost it later in the same round. The same holds for the equational models that will be discussed later in this section. In order to ensure that the results are comparable with the PRISM results we use in the remainder of the paper, instead of (1), the following PCTL property

$$P=?[(\text{true}) U \leq k*(N+1) (\text{ni} \ \& \ \text{sent})] \quad (2)$$

The boolean sent is the conjunction of the variables sent_i , and takes the value true at the end of the round.

The simulation results in Figure 7 are the average of 1000 independent runs. They show that the coverage obtained by simulation is within the MDP bounds and close to the DTMC results. Given the uncertainty inherent within simulation results there is a close match with the PRISM model. They also show that the MDP bounds are within the standard deviation of the simulation results.

B. Equational models for coverage

We compared our model-checking and simulation results with the results obtained in [2]. This paper derives a formal model for coverage a model as differential equation, based on an analysis of the probabilities used as the rates with which nodes exchange data items.

Let $y(t)$ represent the coverage of the item d at time t . The coverage for a completely connected network with N nodes is modelled by the equation:

$$\frac{dy}{dt} = (1 - y(t)) \cdot \sum_{i=0}^{N-1} C(i) \cdot \Phi(i + 1) \quad (3)$$

where $C(i)$ is the probability that a node will be contacted i times in a round, and $\Phi(i)$ the probability that a node that does not hold the item d will acquire it after i contacts. Equation

(3) models the rate of increasing coverage, i.e. the fraction of node that held the data item d in the past, as the fraction $1 - y(t)$ of nodes that have not held item d previously, multiplied with the probability that a node that does not hold it presently acquires it. The latter probability is the sum of the probabilities of a node being contacted i -times multiplied with the probability of acquiring item d within $i + 1$ contacts. This includes the contact initiated by the nodes itself in each round.

The probability that a node is contacted i times by other nodes in the network is:

$$C(i) = \binom{N-1}{i} \left(\frac{1}{N-1}\right)^i \left(\frac{N-2}{N-1}\right)^{(N-1)-i} \quad (4)$$

$\Phi(i)$, the probability that a node that does not hold the data item d , obtains it after i exchanges, is expressed as:

$$\Phi(i) = \sum_{m=0}^{i-1} (1 - \Phi(m)) \cdot P_{get} \cdot (P_{-lose})^{(i-m)-1} \quad (5)$$

where P_{get} is the probability that a node that does not have a copy of d obtains this copy in a shuffle and P_{-lose} is the probability that a node that has a copy of the item does not lose it in a shuffle. These probabilities are expressed as:

$$\begin{aligned} P_{get} &= (P(10|01) + P(11|01)) \cdot x(t) \\ P_{-lose} &= (P(10|10) + P(11|10)) \cdot (1 - x(t)) \\ &\quad + (P(01|11) + P(11|11)) \cdot x(t) \end{aligned} \quad (6)$$

where $x(t)$ is the fraction of network nodes that have item d in their cache at time t . The replication $x(t)$ is modelled:

$$\begin{aligned} \frac{dx}{dt} &= (P(11|10) + P(11|01)) \cdot (1 - x(t)) x(t) \\ &\quad - (P(10|11) + P(01|11)) \cdot x^2(t) \end{aligned} \quad (7)$$

We refer to [2] for further details of these equations, and on the derivation of the probabilities.

Modelling coverage as differential equation assumes that the fraction of nodes that held the item d in the past is continually

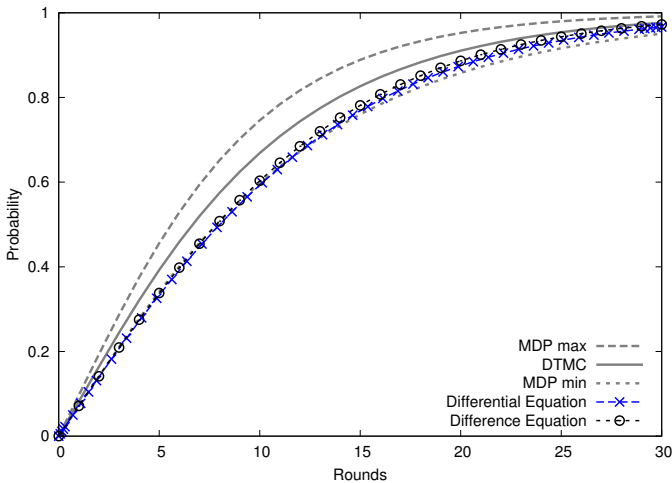


Fig. 8. Comparison of results for PRISM models (grey) and with the results for the equational models.

changing, while it maintains at the same time the discrete notion of a round in equation (3) by including the number of contacts for each round (see (4)). Since the right hand side of equation (3) is actually round-based, where a round takes one unit of time, we modelled coverage alternatively as difference equation, that includes this assumption explicitly. Differential equations (3) and (7) hence become

$$\begin{aligned} y(t+1) &= y(t) + (1 - y(t)) \sum_{i=0}^k C(i) \Phi(i+1) \\ x(t+1) &= x(t) + (P(11|10) + P(11|01)) (1 - x(t)) x(t) \\ &\quad - (P(10|11) + P(01|11)) x^2(t) \end{aligned}$$

A problem with the difference equation is that it uses for all contacts the probabilities at time t , while the probabilities do change as nodes initiate contacts, as illustrated in Figure 4. Since these probabilities increase over time, we should expect that the this model underestimates the coverage.

Example: We compared our model-checking results with the results obtained in [2]. We implemented both equational models in MATLAB. The solution of the differential equation obtained by the numerical integration using the explicit Runge-Kutta (4,5) formula (the Dormand-Prince pair) [7]. For this example, we considered a network with 10 nodes, cache size $c = 100$, number of different data items $n = 500$, and the size of the message $s = 50$. As shown in Figure 8, the coverage for the difference and differential equations roughly coincide. They are lower than the result for the fair DTMC model, and they are mostly within the MDP bounds. One potential cause for this observation was mentioned in the previous paragraph, namely that the difference equation does not account for the fact that the probabilities change during a round. In addition both equational models treat to some extent fractions as probabilities and probabilities as fractions. In the following subsection we investigate effects this might have.

C. Fractions and Probabilities

Treating fractions as probabilities and vice versa is only applicable for models with a sufficiently large number of nodes. But even for large model these notions are fundamentally different. To illustrate, it is best to consider the initial state. All models assume that initially exactly one node holds item d . This means that the fraction of nodes that hold the item initially is $\frac{1}{N}$. Having one or more nodes with d in their cache initially guarantees that the item will spread through the network and that the coverage will eventually converge to 1. If in contrast each node would hold d with probability $1/N$, it would mean that with probability $(1 - 1/N)^N$ no node would hold the data item initially. Hence, the coverage would remain 0 forever. Even if the network size increases toward infinity, we have that with probability

$$\lim_{N \rightarrow \infty} \left(1 - \frac{1}{N}\right)^N = \frac{1}{e}$$

the coverage will remain constant 0, where e is the Euler constant. Thus, the overall coverage will converge to $1 - \frac{1}{e}$.

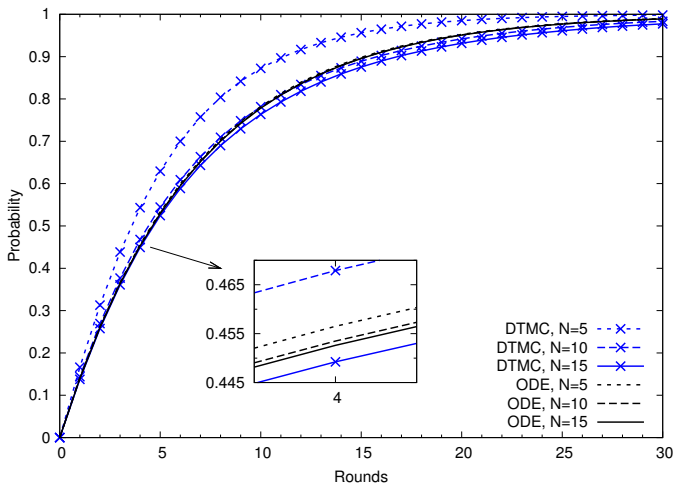


Fig. 9. Comparison of the PRISM results with equational models, using the same initial fraction.

Regarding the initial state, all previous models assume consistently that a fraction of $\frac{1}{N}$ holds the item. However, in later rounds both equational models treat fraction as probabilities and vice versa, should result in a similar, maybe less pronounced, effect.

To investigate this influence we conducted another set of experiments. We considered networks with 5, 10 and 15 nodes, where initially $1/5$ -th of all nodes hold item d . Since differential equation model has the most fuzzy notion of round, we compared it with the probabilistic PRISM model without rounds presented in Subsection III-C4.

Figure 9 depicts the results of the comparison. For the differential equation model we see that the network size has almost no impact. Only equations (4) and (6) incorporate the network size N . It seems that the increasing number of potential contact in a round, does not significantly increase the probability that a node acquires or loses an item. For the PRISM model we observe that, while the parameters s, n and c are the same for all models, and while all models assume that initially $1/5$ of all nodes hold the data item d , it matters whether the network has 5, 10, or 15 nodes. The network with 5 nodes realizes the highest coverage, then the network with 10, and finally 15 nodes. As the number of nodes in the PRISM model increases, and as probabilities behave more like fractions, the coverage converges towards the results for the differential equation. This result agrees with the theoretical result on the limiting behavior of the probabilistic systems, such as epidemics (cf. [8], [9]).

V. RELATED WORK

Fehnker and Gao [10] investigated the impact of modelling choices on the analysis of message dissemination in flooding and probabilistic broadcast protocols. In particular, the authors explore the effect of message collisions, lossy channels, and random delay in the gossiping frequency of nodes in addition to a non-deterministic execution order. For a comparison,

the protocols are modelled in PRISM and implemented in MATLAB for Monte-Carlo simulations.

In [11], a gossip-based protocol Cyclon [3] has been modelled as a DTMC, and implemented in PeerSim. The author analyzed the evolution of the in-degree distribution of nodes under two different schedulers. The first one is a random scheduler, in which each node has the same probability to perform a shuffle step. In the second scheduler, one round ends when all nodes have initiated exactly one interaction. This study has shown how schedulers may affect the complexity of the model and produce different results.

Kwiatkowska et al. [12] presented the case study of a gossip-based peer sampling service [4] with PRISM. The authors considered two properties of the protocol, the longest path between nodes and the time for the network graph to become connected. To clarify the discrepancy between values of the best and worst case behaviors, the authors studied the problem of scheduling. Due to the state-space explosion problem, their analysis is restricted to networks with three and four nodes.

In [13], the authors provided with stepwise guidelines for the mean-field analysis of gossip protocols. The first example is a simple information spread protocol. Each node communicates with a random partner according to a fixed probability, and can only move from the state “uninformed” to the state “informed”. The authors compared the simulations with the hand-crafted DTMC. The latter produced slightly different results for the smaller network size (e.g. 100 nodes), since DTMC assumes every transition to the next time step to be atomic, and owing to the mean-field approximation. The performance measures include distribution of the fraction of the informed nodes at time t and the number of the informed nodes over the time.

VI. CONCLUSION

This paper started from a distributed algorithm for sharing data in a distributed network to explore the different choices for PRISM models. It then compared the PRISM results with the result of alternative approaches; two equational models and a network simulator. While all models are based on a rigorous formalization of the algorithm – or, in the case of the simulator, on an actual implementation – the results did not match. A careful analysis helped to find limitations of PRISM, and uncovered hidden assumptions in the alternative modelling frameworks.

The PRISM models were designed with the state explosion problem in mind. We abstracted the part of the algorithm that deals with pair-wise communication, and modelled it as an atomic probabilistic transition. This abstraction was made possible by the analysis in [2] of transition probabilities of the shuffling protocol. The PRISM model covered only which nodes initiate in what order contact with other nodes to exchange information. This abstraction allowed us to model check networks with up to 15 nodes, and instances of model checking took up to 10 minutes on a Intel Core Duo CPU, 1.2 GHz, with 1 GB of RAM.

This paper uses PRISM to study the different scheduling policies, and to determine best and worst case bounds. However, the specification logic PCTL did not let us define the exact equivalent of *coverage*. PCTL provides the capability to study state probabilities, while coverage is best defined as an expected value over CTL state formulae. Fortunately, this limitation was only relevant for MDP models; in our DTMC models it becomes irrelevant since the network is completely connected and all nodes symmetric.

Another limitation of PRISM and PCTL is that it does not allow to query state probabilities after k iterations. These state probabilities could have been used to compare also other measures, such as *replication*. PRISM provides the transition matrices to compute replication in, for example, MATLAB, but unfortunately, these sparse matrices were too large to manipulate. While this demonstrates the efficiency of the MTBDD encoding used in PRISM [6], it would be an even more useful tool if could give state probabilities after k steps, even, if only for DTMCs.

A contributing factor to the difference between the PRISM and equational models is that the latter treat fractions as probabilities and vice versa. We were able to illustrate that this kind of models are more accurate the larger the number of nodes is. PRISM distinguishes between probabilities and fractions inherently, since it captures fractions explicitly in the state of the model.

While we were able to model check networks with up to 15 nodes in PRISM, it should be noted that the alternative models can be used for a few thousand nodes, as demonstrated in [2]. The PRISM analysis of relatively small network proved useful nevertheless, since it helps to uncover simplifying assumptions that allow these models to deal with much larger networks, and thus with the interpretation the obtained results.

ACKNOWLEDGMENT

We thank Daniela Gavidia for providing the initial implementation of the shuffling protocol for PeerSim and anonymous reviewers for their comments.

REFERENCES

- [1] D. Gavidia, S. Voulgaris, and M. van Steen, "A gossip-based distributed news service for wireless mesh networks," in *Proc. Conf. on Wireless On-demand Network Systems & Services (WONS'06)*. IEEE, 2006, pp. 59–67.
- [2] R. Bakhshi, D. Gavidia, W. Fokkink, and M. van Steen, "An analytical model of information dissemination for a gossip-based protocol," *Computer Networks*, 2009, in Press. An early version appeared in [14].
- [3] S. Voulgaris, D. Gavidia, and M. van Steen, "Cyclon: Inexpensive membership management for unstructured p2p overlays," *J. Network and Systems Management*, vol. 13, no. 2, pp. 197–217, 2005.
- [4] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen, "The peer sampling service: Experimental evaluation of unstructured gossip-based implementations," in *Proc. ACM/IFIP/USENIX Conf. on Middleware (Middleware'04)*, ser. LNCS, vol. 3231. Springer, 2004, pp. 79–98.
- [5] M. Jelasity, A. Montresor, G. P. Jesi, and S. Voulgaris, "PeerSim: A peer-to-peer simulator," <http://peersim.sourceforge.net/>.
- [6] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker, "PRISM: A Tool for Automatic Verification of Probabilistic Systems," in *Proc. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, ser. LNCS, vol. 3920. Springer, 2006, pp. 441–444.
- [7] G. Forsythe, M. Malcolm, and C. Moler, *Computer Methods for Mathematical Computations*. 1977: Prentice-Hall, New Jersey.
- [8] T. G. Kurtz, *Approximation of population processes*, ser. CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, 1981, vol. 36.
- [9] R. Darling and J. Norris, "Differential equation approximations for Markov chains," *Probab. Surveys*, vol. 5, pp. 37–79, 2008.
- [10] A. Fehnker and P. Gao, "Formal verification and simulation for performance analysis for probabilistic broadcast protocols," in *Proc. Conf. on Ad-Hoc, Mobile, and Wireless Networks (ADHOC-NOW'06)*, ser. LNCS, vol. 4104. Springer, 2006, pp. 128–141.
- [11] F. Bonnet, "Performance analysis of Cyclon, an inexpensive membership management for unstructured P2P overlays." Master Thesis, ENS Cachan Bretagne, University of Rennes, IRISA, 2006.
- [12] M. Kwiatkowska, G. Norman, and D. Parker, "Analysis of a gossip protocol in PRISM," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 3, pp. 17–22, 2008.
- [13] R. Bakhshi, L. Cloth, W. Fokkink, and B. R. Haverkort, "Mean-field analysis for the evaluation of gossip protocols," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 3, pp. 31–39, 2008.
- [14] R. Bakhshi, D. Gavidia, W. Fokkink, and M. van Steen, "An analytical model of information dissemination for a gossip-based protocol," in *Proc. Conf. on Distributed Computing and Networking (ICDCN'09)*, ser. LNCS, vol. 5408. Springer, 2009, pp. 230–242.

```

nondeterministic

const int c = 100;
const int s = 50;
const int n = 500;
const int N = 3;

global n1: bool init true;
global n2: bool init false;
global n3: bool init false;

formula p0000 = 1;
formula p0101 = (c-s)/c;
formula p1001 = (s/c)*(n-c)/(n-s);
formula p1101 = (s/c)*(c-s)/(n-s);
formula p0110 = (s/c)*(n-c)/(n-s);
formula p1010 = (c-s)/c;
formula p1110 = (s/c)*(c-s)/(n-s);
formula p0111 = (s/c)*((c-s)/c)*((n-c)/(n-s));
formula p1011 = (s/c)*((c-s)/c)*((n-c)/(n-s));
formula p1111 = 1-2*(s/c)*((c-s)/c)*((n-c)/(n-s));

module node1

sent1: bool init false;

[] !sent1 & !n1 -> 1/(2)*(!n2?p0000:0): (n1'=false) & (n2'=false) & (sent1'=true)
+ 1/(2)*( n2?p0101:0): (n1'=false) & (n2'=true) & (sent1'=true)
+ 1/(2)*( n2?p1001:0): (n1'=true) & (n2'=false) & (sent1'=true)
+ 1/(2)*( n2?p1101:0): (n1'=true) & (n2'=true) & (sent1'=true)
+ 1/(2)*(!n3?p0000:0): (n1'=false) & (n3'=false) & (sent1'=true)
+ 1/(2)*( n3?p0101:0): (n1'=false) & (n3'=true) & (sent1'=true)
+ 1/(2)*( n3?p1001:0): (n1'=true) & (n3'=false) & (sent1'=true)
+ 1/(2)*( n3?p1101:0): (n1'=true) & (n3'=true) & (sent1'=true);
[] !sent1 & n1 -> 1/(2)*(!n2?p0110:0): (n1'=false) & (n2'=true) & (sent1'=true)
+ 1/(2)*(!n2?p1010:0): (n1'=true) & (n2'=false) & (sent1'=true)
+ 1/(2)*(!n2?p1110:0): (n1'=true) & (n2'=true) & (sent1'=true)
+ 1/(2)*( n2?p0111:0): (n1'=false) & (n2'=true) & (sent1'=true)
+ 1/(2)*( n2?p1011:0): (n1'=true) & (n2'=false) & (sent1'=true)
+ 1/(2)*( n2?p1111:0): (n1'=true) & (n2'=true) & (sent1'=true)
+ 1/(2)*(!n3?p0110:0): (n1'=false) & (n3'=true) & (sent1'=true)
+ 1/(2)*(!n3?p1010:0): (n1'=true) & (n3'=false) & (sent1'=true)
+ 1/(2)*(!n3?p1110:0): (n1'=true) & (n3'=true) & (sent1'=true)
+ 1/(2)*( n3?p0111:0): (n1'=false) & (n3'=true) & (sent1'=true)
+ 1/(2)*( n3?p1011:0): (n1'=true) & (n3'=false) & (sent1'=true)
+ 1/(2)*( n3?p1111:0): (n1'=true) & (n3'=true) & (sent1'=true);
[done] sent1 -> sent1'=false;

endmodule

module node2 = node1 [sent1=sent2,sent2=sent1,sent3=sent3,n1=n2,n2=n1,n3=n3] endmodule
module node3 = node1 [sent1=sent3,sent2=sent1,sent3=sent2,n1=n3,n2=n1,n3=n2] endmodule

```

Fig. 10. Non-deterministic PRISM model of the shuffling algorithm for 3 nodes.