Accuracy-Speedup Tradeoffs for a Time-Parallel Simulation of Wireless Ad hoc Networks

Guoqiang Wang, Damla Turgut, Ladislau Bölöni, and Dan C. Marinescu
School of Electrical Engineering and Computer Science
University of Central Florida, Orlando, FL, USA
{gwang, turgut, lboloni, dcm}@eecs.ucf.edu

Abstract

We introduce a scalable algorithm for time-parallel simulations of wireless ad hoc networks and report on our results. Our approach decomposes the simulation into overlapping temporal components; the individual components are computed using an unmodified sequential network simulator such as `NS-2`*. Our algorithm is iterative and the accuracy of the results increases with the number of iterations. We find that the approach allows the simulation of ad hoc networks with a number of nodes larger than those feasible with sequential network simulators on single CPUs. The algorithm is scalable, we can simulate larger time intervals by increasing the number of virtual processors carrying out the simulation. We identify the parameters that can be investigated with the algorithm and report on the accuracy of our results and on the achieved simulation speedup.*

1 Introduction

Simulation of wireless networks is important for protocol design and wireless system research. As the research progresses from relatively simple systems composed of several nodes connected to a wireless access point to large systems which might be composed of thousands of nodes (such as wireless sensor networks), the size of the simulation problem becomes so large that it clearly exceeds the capabilities of a single machine. Therefore, it is important to look into parallel simulation approaches. Parallel discrete event simulation (PDES) is proposed in recent years to reduce the overall execution time by parallel execution of the simulation on multiple processors. The parallel simulation approaches merge to two main categories: space-parallel simulation (distributed simulation), and time-parallel simulation. In the space-parallel simulation approach [6, 13], the simulation model is decomposed into a number of components on a spatial basis. Each component is modeled by a *logical processor*. Logical processors establish a communication mechanism among each other to avoid or fix possible causality errors. The Parallel/Distributed NS (PDNS) [21] project uses a space-parallel simulation approach based on the `NS-2` network simulator [18]. However, the applicability of PDNS is limited to *wired* networks, and the traffic simulated at different spatial partitions cannot affect each other.

In the time-parallel simulation approach [1, 7, 8, 12, 19, 20], the long period of simulation time is partitioned into smaller adjacent simulation intervals, and each simulation interval is assigned to a processor with a *guessed* initial state. The simulation terminates when the final state of each interval matches the initial state of its successive interval. Thus, state matching is one of the key problems of time-parallel simulation. In [12], the authors propose a time-parallel simulation algorithm based on state matching. A simulation is defined as *partial regenerative* if there exists a subset of the system state variables such that the subsystem represented by the subset can repeat its state infinitely. The system is then partitioned at the *regeneration points* which starts a *regenerative substate*. [14, 15] indicate that in some cases the *regeneration points* of a regenerative simulation can be found without performing a detailed simulation; the state matching problem can be solved by performing a precomputation. [20] proposes a *pre-simulation* to identify regenerative points by using Markovian modeling. Although it is hard to obtain accurate simulation results efficiently with the time-parallel simulation approach, approximate simulation results [8, 10, 11, 19] can be produced efficiently.

A parallel simulator, SWiMNet [2, 3, 4], is used for personal communication services (PCS) networks. It is based on a combination of optimistic and conservative paradigms and makes use of the event precomputation by the model independence within the PCS model. Independencies between processes allow to achieve parallelism. SWiMNet is used in simulation of PCS networks with fixed channel assignment by specifying fine grained mobility, variable call

1-4244-0419-3/06/$20.00 ©2006 IEEE

process, and arbitrary coverage area.

The major difficulty in the parallel simulation approaches is to solve the dependencies among the partitions. A careful study of the temporal dependencies for wireless networks with various traffic patterns shows that there are few strong dependencies which span a large temporal interval. By ignoring the weak dependencies we can achieve fast simulation results which are a good approximation of the exact results. Furthermore, we can assemble a system where coarse approximations are obtained very quickly, while the continuation of the simulation process yields results with increasingly higher precision. In the knowledge of the approximate results, the researcher might decide whether the expensive continuation of the simulations is justified or not.

In this paper, we propose a time-parallel simulation approach for wireless ad hoc networks. Our approach is based on the `NS-2` network simulator and is independent of the networking protocol (although it shows variations on the achievable speedup depending on the protocol). We identify the set of parameters that can be obtained with our approach. We also include an evaluation and validation of approximate simulation results. The paper is organized as follows. A time-parallel simulation approach is introduced in Section 2. A series of simulation studies investigating the speedup and precision of the proposed method for typical wireless network simulation scenarios are presented in Section 3. We conclude in Section 4.

2 Time-parallel simulation algorithm

A simulation S of an ad hoc network calculates the simulation trace $\mathcal{T}$ of event set E which occurs in geographic area A, within time interval τ, when the initial state is $\mathcal{I}$, and the final state is $\mathcal{F}$. Formally, we denote a simulation as a six-tuple set $S = (A, \tau, E, \mathcal{I}, \mathcal{F}, \mathcal{T})$. We partition the temporal dimension of the simulation S into m time intervals $\{\tau_i | \tau_i = [(i-1)\frac{D(\tau)}{m}, i\frac{D(\tau)}{m}], i \in \{1, \ldots, m\}\}$, where $D(\tau)$ calculates the duration of time interval τ. Then, we obtain a time-parallel simulation set $I^{(0)} = \{S_i | S_i = (A, \tau_i, E(S_i), \phi, \mathcal{F}(S_i), \mathcal{T}(S_i)), i \in \{1, \ldots, m\}\}$. Thus the time-parallel simulation set will operate on the full spatial component A, but a subset of the temporal span τ_i. In case of an exact simulation, the final state of S_i needs to match the initial state of S_{i+1}, $i \in \{1, \ldots, m-1\}$. However, since there is no prior knowledge for the initial states of $S_2, \ldots, S_m$ when they are parallelized, the combined simulation trace $\mathcal{T}(I^{(0)}) = \mathcal{T}(S_1) \cup \mathcal{T}(S_2) \cup \ldots \cup \mathcal{T}(S_m)$ is far from being approximate.

After the execution of $I^{(0)}$, the simulation trace of S_1 becomes an accurate trace. A natural solution to refine the other simulation traces is to apply the final state of a simulation interval to the initial state of its successive simulation interval, $\mathcal{I}(S_i^{(1)}) \leftarrow \mathcal{F}(S_{i-1})$ and re-execute the new simulation set $I^{(1)} = \{S_i^{(1)} | S_i^{(1)} = (A, \tau_i, E(S_i^{(1)}), \mathcal{F}(S_{i-1}), \mathcal{F}(S_i^{(1)}), \mathcal{T}(S_i^{(1)})), i \in \{2, \ldots, m\}\}$. The causality dependencies of events of two consecutive time intervals are now removed, meanwhile the dependencies for events in a time interval to affect later time intervals are weakened, since the strongest causality dependencies (dependencies of two consecutive time intervals) are already removed. The combined simulation trace after the first iteration, $\mathcal{T}(I^{(1)}) = \mathcal{T}(S_1) \cup \mathcal{T}(S_2^{(1)}) \cup \mathcal{T}(S_3^{(1)}) \cup \ldots \cup \mathcal{T}(S_m^{(1)})$, should become more accurate, compared to $\mathcal{T}(I^{(0)})$.

As we repeat this process, all strong dependencies are further removed and the simulation trace becomes more accurate. The parallel simulation continues until the error of simulation results in all relevant metrics is lower than a predefined treshold. The k-th iteration contains $m-k$ simulation segments $I^{(k)} = \{S_i^{(k)} | S_i^{(k)} = (A, \tau_i, E(S_i^{(k)}), \mathcal{F}(S_{i-1}^{(k-1)}), \mathcal{F}(S_i^{(k)}), \mathcal{T}(S_i^{(k)})), i \in \{k+1, \ldots, m\}\}$. After k-th iteration, the traces of simulation intervals $S_1, S_2^{(1)}, \ldots, S_{k+1}^{(k)}$ become accurate. The combined simulation trace of $(I^{(k)})$ can be obtained by

$\mathcal{T}(I^{(k)}) = \mathcal{T}(S_1) \cup \mathcal{T}(S_2^{(1)}) \cup \mathcal{T}(S_3^{(2)}) \cup \ldots \cup \mathcal{T}(S_k^{(k-1)}) \cup \mathcal{T}(S_{k+1}^{(k)}) \cup \mathcal{T}(S_{k+2}^{(k)}) \cup \ldots \cup \mathcal{T}(S_m^{(k)})$.

We illustrate the approximate simulation in the following example, see Figure 1. Assume a simulation $S = (500 \times 500, [0, 200], E, \phi, \mathcal{F}, \mathcal{T})$ is segmented into 10 equal-duration time intervals, with $\tau_i = [20(i-1), 20i], i \in \{1, \ldots, 10\}$. The simulation sets of iteration 0, 1, 2 are as follows

$I^{(0)} = S_1 \cup S_2 \cup S_3 \cup S_4 \cup S_5 \cup S_6 \cup S_7 \cup S_8 \cup S_9 \cup S_{10}$;

$I^{(1)} = S_2^{(1)} \cup S_3^{(1)} \cup S_4^{(1)} \cup S_5^{(1)} \cup S_6^{(1)} \cup S_7^{(1)} \cup S_8^{(1)} \cup S_9^{(1)} \cup S_{10}^{(1)}$;

$I^{(2)} = S_3^{(2)} \cup S_4^{(2)} \cup S_5^{(2)} \cup S_6^{(2)} \cup S_7^{(2)} \cup S_8^{(2)} \cup S_9^{(2)} \cup S_{10}^{(2)}$.

The simulation traces after iterations 0, 1, 2 are as follows

$\mathcal{T}(I^{(0)}) = \mathcal{T}(S_1) \cup \mathcal{T}(S_2) \cup \mathcal{T}(S_3) \cup \ldots \cup \mathcal{T}(S_{10})$;

$\mathcal{T}(I^{(1)}) = \mathcal{T}(S_1) \cup \mathcal{T}(S_2^{(1)}) \cup \mathcal{T}(S_3^{(1)}) \cup \ldots \cup \mathcal{T}(S_{10}^{(1)})$;

$\mathcal{T}(I^{(2)}) = \mathcal{T}(S_1) \cup \mathcal{T}(S_2^{(2)}) \cup \mathcal{T}(S_3^{(2)}) \cup \ldots \cup \mathcal{T}(S_{10}^{(2)})$.

The approach outlined above requires the ability to accurately capture the complete state of a simulation, and to be able to restart the simulator with an arbitrary initial state. For instance, to accurately capture the state of the simulation at the MAC layer, we need to save information such as the packets in the priority queue, the status of timers for NAV, RTS, CTS, and all the other MAC related information. In addition, we need the ability to start a simulation with an arbitrary value of these parameters. This is not possible with the stock `NS-2` distribution. Furthermore, as different protocols require different state data, we would need to develop

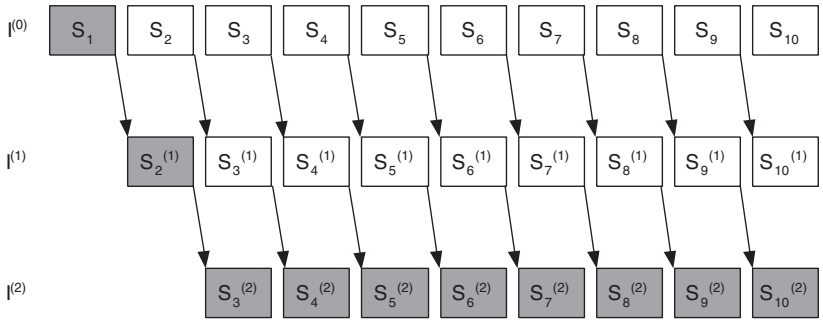

Figure 1. Illustration of the time-parallel simulation approach. The duration of the simulation S is 200. It is segmented into 10 equal-duration simulation intervals. The shaded simulation segments are used to compose the final simulation trace after 3 iterations.

new patches for every new protocol considered.

In the following, we present a method to rearrange the calculations in such a way that the state saving and restoration is not necessary, and we can use the stock, unpatched simulator for arbitrary protocols. Let us consider the approximate simulation trace obtained after $\gamma + 1$ iterations, at the end of iteration γ. We partition the simulation interval into m overlapping time intervals:

$$\eta_i = \begin{cases} [(i-1)\frac{\mathcal{D}(\tau)}{m}, (i-1+\gamma)\frac{\mathcal{D}(\tau)}{m}], \ i \in \{1 \dots m-\gamma\}; \\ [(i-1)\frac{\mathcal{D}(\tau)}{m}, \mathcal{D}(\tau)], \qquad i \in \{m-\gamma+1 \dots m\}. \end{cases}$$

The time interval η_i starts at the same time with τ_i, but it lasts for $\gamma + 1$ times the duration of τ_i for the first $m - \gamma$ time intervals; η_i ends at $\mathcal{D}(\tau)$ for later time intervals. We obtain a time-parallel simulation set $L = \{P_i | P_i = (A, \eta_i, E(P_i), \phi, \mathcal{F}(P_i), \mathcal{T}(P_i)), i \in \{1, \dots, m\}\}$. η_i can be divided into a set of sub time intervals, each with a duration of $\frac{\mathcal{D}(\tau)}{m}$.

Call $P_{i,j}$ the j-th sub-interval of P_i (enumeration starts at 0):

$$P_i = \begin{cases} P_{i,0} + \ldots + P_{i,\gamma}, \ i \in \{1 \dots m-\gamma\}; \\ P_{i,0} + \ldots + P_{i,m+1-i}, i \in \{m-\gamma+1 \dots m\}. \end{cases}$$

$P_{i,j}$ simulates the same simulation interval as S_{i+j}. For instance, in Figures 1 and 2, $P_{3,2}$, $P_{4,1}$ and S_5 all simulate the simulation interval [80, 100].

We assume the same simulation S in Figure 1 as our example, $m = 10, \gamma = 2$. The new simulation set $L = \{P_1, P_2, \dots, P_{10}\}$ is shown in Figure 2. We observe that $P_{i,0}$ and S_i are essentially equivalent since they simulate the same time interval without prior knowledge of initial states (see Figures 1 and 2). Thus, $I^{(0)}$ can be rewritten as $I^{(0)} = \{S_i | i \in \{1, \dots, m\}\} = \{P_{i,0} | i \in \{1, \dots, m\}\}$.

The final state of simulation $P_{i,0}$ is naturally transferred as the initial state of simulation $P_{i,1}$, thus $P_{i,1}$ and $S_{i+1}^{(1)}$ are essentially equivalent since they simulate the same interval with the same initial states. Thus, $I^{(1)}$ can be rewritten as $I^{(1)} = \{S_i^{(1)} | i \in \{2, \dots, m\}\} = \{P_{i,1} | i \in \{1, \dots, m-1\}\}$.

Similarly, $I^{(\gamma)} = \{S_i^{(\gamma)} | i \in \{\gamma+1, \dots, m\}\} = \{P_{i,\gamma} | i \in \{1, \dots, m-\gamma\}\}$.

The alternative simulation algorithm has several desirable properties: (i) the final state corresponding to time interval τ_i is naturally and accurately accepted as the initial state of its successive time interval, τ_{i+1}, and the obstacle to save the state information is removed; (ii) if the number of iterations ($\gamma + 1$) to obtain the approximate simulation trace is known, the subset $\{P_i | i \in \{m-\gamma+1, \dots, m\}\}$ is not needed in the computation of $\mathcal{T}(I^{(\gamma)})$, thus the required number of computational nodes can be reduced from m to $m - \gamma$. In our example, the simulation set $\{P_9, P_{10}\}$ is not needed, and the required number of cluster nodes can be reduced from 10 to 8.

3 Simulation study

We evaluate a set of metrics to establish the level of accuracy of the results obtained by our time-parallel simulation approach relative to an exact sequential simulation. We study the number of iterations to obtain approximate simulation results, as the simulated time is cut into segments of different durations. We are also concerned with the sensitivity of our method to the network load and network mobility.

We use the "random waypoint" model [5, 9] to simulate the node movement. Traffic patterns are generated by *constant bit rate* (CBR) sources sending 512-byte UDP packets at a rate of 1 packet per second. The simulation area is 500×500 and the default number of nodes is 80. All the nodes have a transmission range of 100 meters. The simulation time of 600 seconds is segmented into 20 time intervals of 30 seconds. We run several simulation experiments by varying the segment duration, number of CBR sources, and

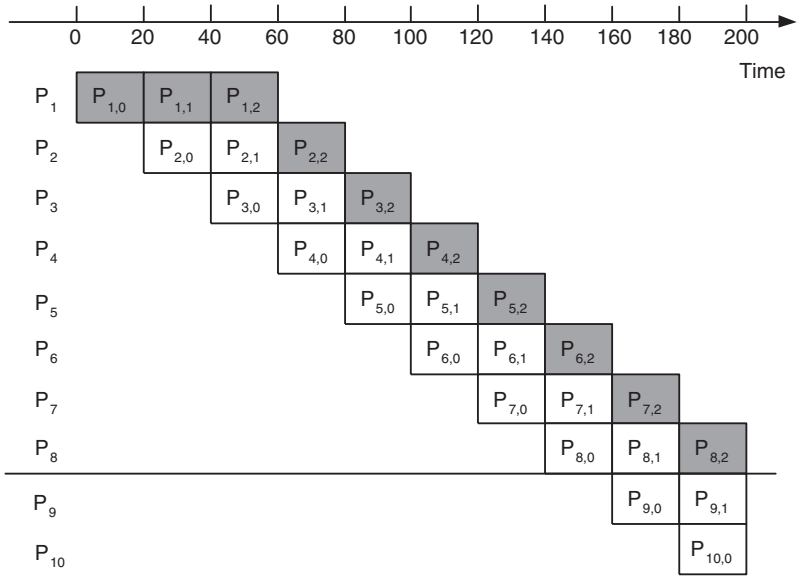

Figure 2. An alternative approach to the simulation in Figure 1. Shaded blocks correspond to partial results used to construct the final simulation results.

the speed. Table 1 shows the default settings and the range of the parameters for our experiments.

Table 1. The default values and the range of the simulation parameters.

Field	Value	Range
simulation area	$500 \times 500(m^2)$	
number of nodes	80	
transmission range	100 (m)	
speed	1 (m/s)	1 - 21 (m/s)
pause time	15 (s)	
simulation time	600 (s)	
segment duration	30 (s)	10 - 60
number of CBR sources	20	4 - 40
CBR packet size	512 (bytes)	
CBR sending rate	4 (kbps)	

3.1 Performance metrics

To establish the accuracy of our time-parallel simulation relative to an exact sequential simulation, we evaluate the *relative error* for several performance indicators. Let M_0 be the result produced by the exact sequential simulation and M the one produced by our time-parallel algorithm; the *relative error* for this metric is $\varepsilon = \frac{M-M_0}{M_0} \times 100\%$. We investigate the relative error for the packet loss ratio and the throughput of the given algorithm.

For each experiment, we randomize the source-destination pairs of CBR sources, and execute 10 times to obtain the average, as well as, 95% confidence interval for each quantity. We use Destination Sequenced Distance Vector Routing (DSDV)[16] and Ad-hoc On-Demand Distance Vector Routing (AODV) [17] routing protocols to investigate the performance of our time-parallel simulation algorithm. The simulation was run on a cluster computer composed of 128 64-bit Opteron processors.

3.2 Simulation results

Segment Duration. This set of experiments (Figure 3) allows us to investigate the effect of segment duration and determine the number of iterations required to obtain results with a given level of accuracy; we choose as a threshold for the relative error $\alpha = 5\%$. We experiment with segment durations of 10, 20, 30, 40, 50, and 60 seconds. As the simulation time is fixed at 600 seconds, the number of segments are 60, 30, 20, 15, 12, and 10, respectively. We compare the simulation results after each iteration with the simulation results obtained by exact sequential simulation.

The curves labeled *PROTOCOL.ITERATION.i* in Figure 3 show the relative error after iteration i for the two protocols. Table 2 summarizes the number of iterations needed to achieve a relative error not larger than 5%, as well as the speedup compared to sequential execution.

We note that the speedup for a single iteration is equal with the number of time segments (provided that there

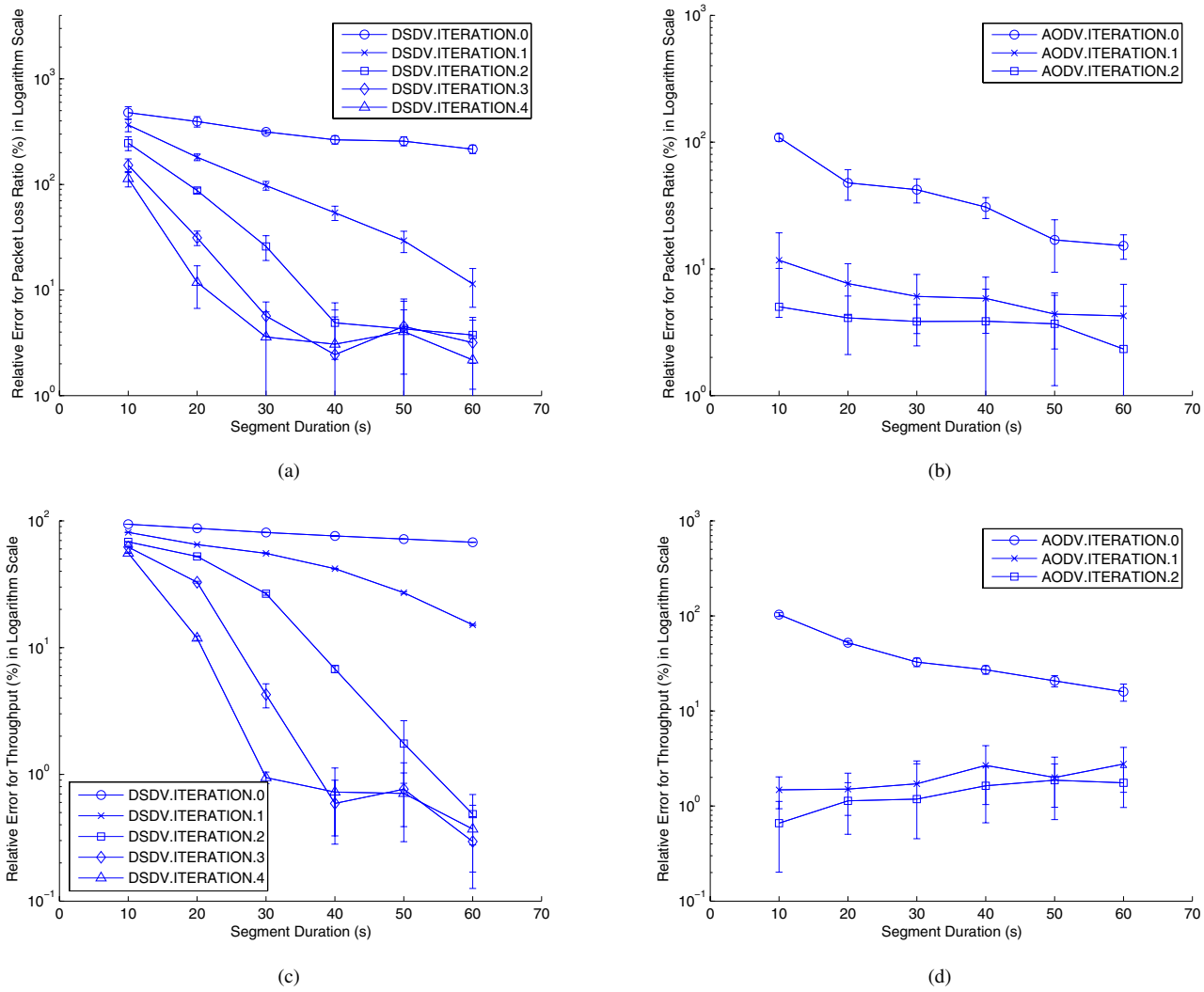

Figure 3. Relative error for packet loss ratio and throughput as function of the segment duration in a logarithmic scale; DSDV (left) and AODV (right). (a) and (b) show the relative error for the packet loss ratio; (b) and (d) show the relative error for the throughput.

are sufficient number of computational nodes); however, smaller segments require a larger number of iterations to achieve equivalent precision. Overall, however, the speedup tends to increase with the decrease of segment size. Thus, a proper segment duration need to be chosen according to the simulation time and the number of available computational nodes.

Network Load. This set of experiments (Figure 4) investigate the effect of the network load upon the number of iterations and the accuracy. The number of CBR sources ranges from 4 to 40. The size of a CBR packet is 512-bytes, and the rate for each source is 1 packet per second.

We notice that for both protocols the relative error fluctuates around a stable value regardless of workload after a certain number of iterations. The number of iterations and the speedup for DSDV are: 4 and 5 respectively. The same figures for AODV are 3 and 6.7 respectively. The number of iterations required to obtain the simulation results is about the same regardless of the number of CBR sources.

Node Mobility. We conducted this set of experiments (Figure 4) to reveal whether the node mobility will affect the number of iterations required to achieve a relative error threshold. The node mobility ranges from 1 to 21 m/sec.

From Figure 5, we can see that for both protocols the rel-

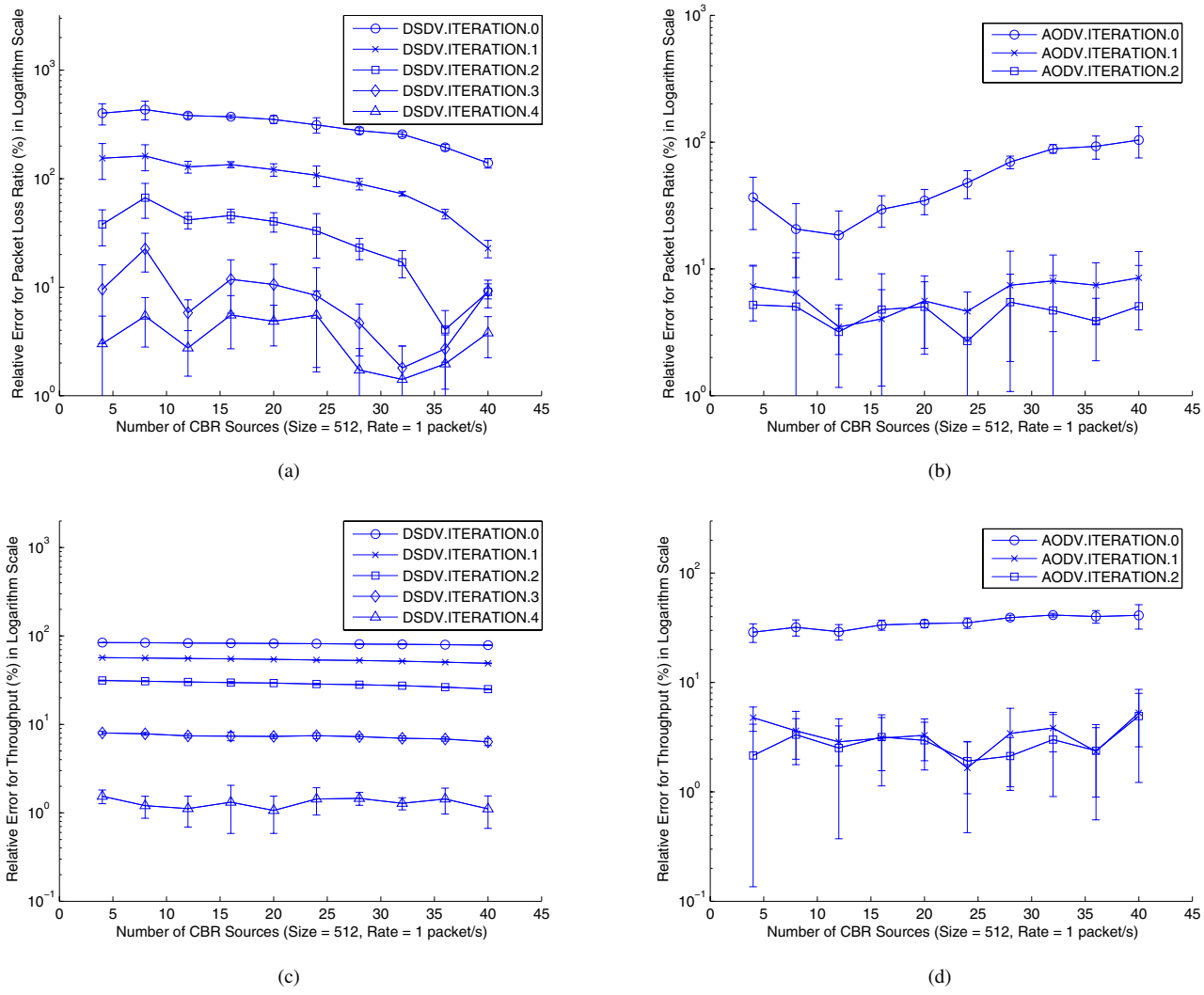

Figure 4. Relative error for packet loss ratio and throughput as function of the network load in a logarithmic scale; DSDV (left) and AODV (right). (a) and (b) show the relative error for the packet loss ratio; (b) and (d) show the relative error for the throughput.

ative error for the metrics we investigate fluctuates around a relatively stable value regardless of the workload. The results are increasingly more accurate as the increased number of iterations increases. After i iterations, the relative error for packet loss ratio drops below the threshold α, and is about the same regardless of the number of CBR sources. The number of iterations and the speedup for both protocols are identical with those presented earlier when we allowed the number of CBR sources to vary.

We can draw the conclusion that *the performance of the parallel simulation is not sensitive to the network load or the node mobility*.

Speedup and Maximum Network Size. In our experiments, the number of nodes range from 100 to 1500. The density of the map is fixed at $\frac{1}{2500}m^{-2}$, and the other parameters are set as default (Table 1).

For DSDV the actual execution time for 1400 simulated nodes is approximately 110 minutes. The parallel simulation requires almost 24 minutes and has a speedup of almost 5. If we restrict the execution time to 10 minutes the maximum number of nodes is 180 for sequential simulation and 1000 for parallel simulation. The execution time for AODV is: 90 minutes for sequential simulation and 15 minutes for the parallel one. When we restrict the execution time to 10

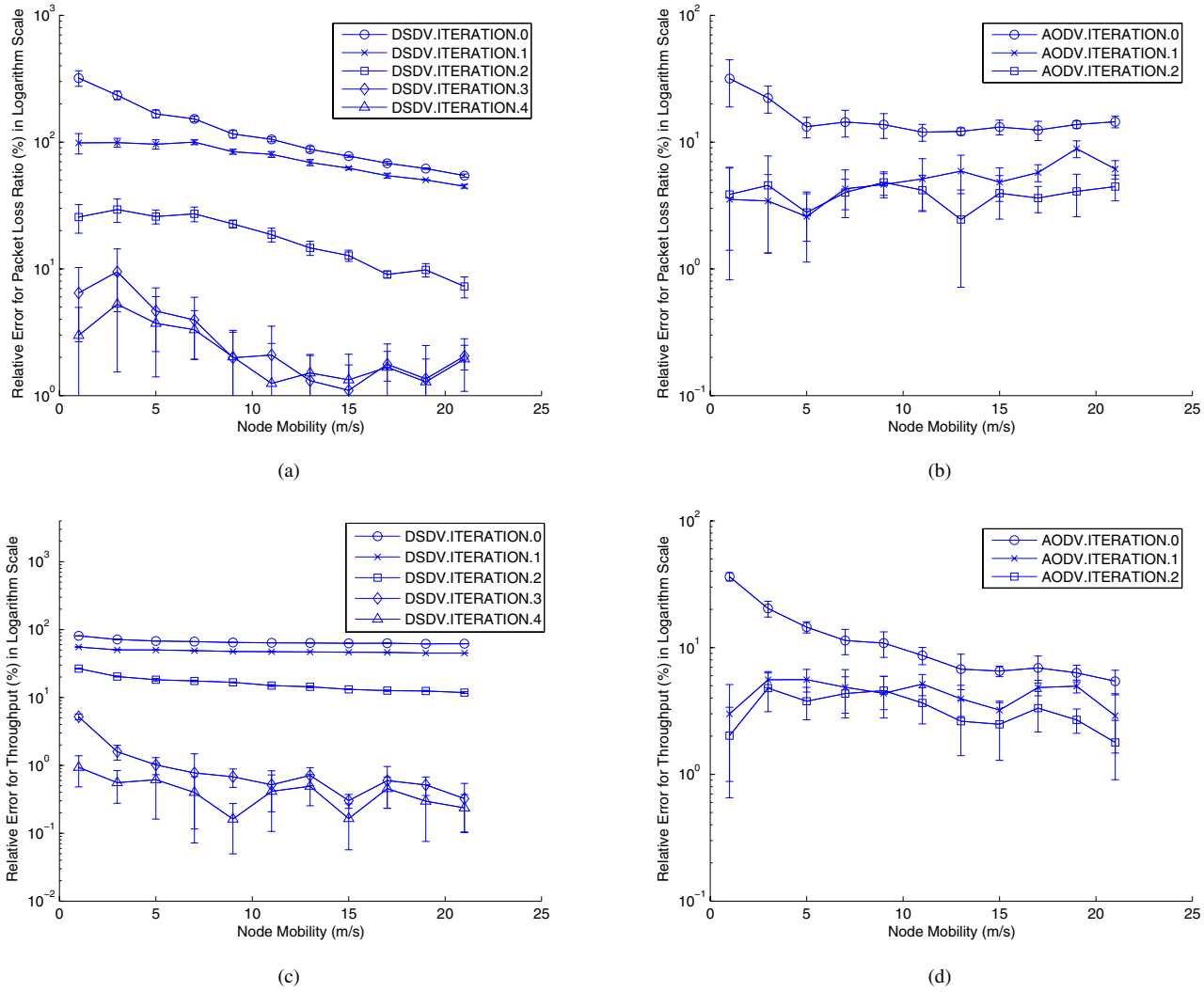

Figure 5. Relative error for packet loss ratio and throughput as function of the node mobility in a logarithmic scale; `DSDV` (left) and `AODV` (right). (a) and (b) show the relative error for the packet loss ratio; (b) and (d) show the relative error for the throughput.

minutes the maximum number of nodes is 200 for the sequential and 1200 for the parallel simulation.

4 Conclusions

The time-parallel simulation algorithm presented in this paper allows a tradeoff between the accuracy of the results and the simulation time. Rather than seeking to produce results that are in perfect agreement with those produced by a "classical" sequential simulation we are content with approximate results which would allow us to draw qualitative, rather than quantitative conclusions regarding the system's performance.

A significant advantage of our method is that the simulation segments are executed using an unmodified `NS-2` simulator; this allows us to support all the protocols supported by `NS-2`. It is possible to extract approximate results in real time from a simulation in progress. The accuracy of the results is increasing with the number of iterations. For a given accuracy, the speedup compared to the sequential execution is increasing when the segment size is decreasing; it is a good practice to choose the segment size as the the simulated time interval divided to the number of available computational nodes.

Table 2. The number of iteration required to achieve an error threshold $\alpha = 5\%$ and the speedup, function of segment duration.

Segment Duration (s)	Iteration required		Speedup	
	DSDV	AODV	DSDV	AODV
10	9	3	6.7	20
20	5	3	6	10
30	4	3	5	6.7
40	3	3	5	5
50	3	2	4	6
60	2	2	5	5

The performance metrics studied in this paper are the packet loss ratio and the throughput. The results for some other important metrics, such as the routing overhead, could not be included due to space limitations; however others, such as *packet latency* cannot be evaluated accurately by our method.

We found that the algorithm is not sensitive to the network load or node mobility. We found that the source initiated (reactive) routing protocols such as AODV require a lower number of iterations to achieve the same accuracy as the table driven (proactive) ad hoc routing protocols such as DSDV. In general, we found that the more state is maintained by a protocol, the lower the achievable speedup; a more exact quantification of this conjecture across a wider range of protocols is subject of future work.

Acknowledgment

The research reported in this paper was partially supported by National Science Foundation grants ACI0296035 and EIA0296179.

References

[1] S. Andradóttir and T. J. Ott. Time segmentation parallel simulation of networks of queues with loss or communication blocking. *ACM Transactions on Modeling and Computer Simulations*, 5(4):269–305, 1995.

[2] A. Boukerche, S. K. Das, and A. Fabbri. SWiMNet: a scalable parallel simulation testbed for wireless and mobile networks. *ACM/Kluwer Wireless Networks*, 7(5):467–486, 2001.

[3] A. Boukerche, S. K. Das, A. Fabbri, and O. Yildiz. Exploiting model independence for pcs parallel simulation. In *Proceedings of 13th ACM/IEEE workshop on Parallel and Distributed Simulation*, pages 166–173, 1999.

[4] A. Boukerche and A. Fabbri. Partitioning parallel simulation of wireless networks. In *Proceedings of 2000 Winter Simulation Conference*, pages 1449–1457, 2000.

[5] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of MOBICOM*, pages 85–97, 1998.

[6] R. M. Fujimoto. Parallel discrete event simulation. *Communications of ACM*, 33(10):30–53, 1990.

[7] M. Hoseyni-Nasab and S. Andradóttir. Parallel simulation by time segmentation: Methodology and applications. In *Proceeding of the 1996 Winter Simulation Conference*, pages 376–381, 1996.

[8] M. Hoseyni-Nasab and S. Andradóttir. Time segmentation parallel simulation of tandem queues with manufacturing blocking. In *Proceeding of the 1998 Winter Simulation Conference*, 1998.

[9] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[10] T. Kiesling. Approximate time-parallel cache simulation. In *Procedings of 2004 Winter Simulation Conference*, pages 345–354, 2004.

[11] T. Kiesling. Using approximation with time-parallel simulation. *SIMULATION*, 81(4):255–266, April 2005.

[12] Y.-B. Lin and E. D. Lazowska. A time-division algorithm for parallel simulation. *ACM Transactions on Modeling and Computer Simulations*, 1(1):73–83, 1991.

[13] V. K. Madisetti, J. C. Walrand, and D. G. Messerschmitt. Asynchronous algorithms for the parallel simulation of event-driven dynamical systems. *ACM Transactions on Modeling and Computer Simulations*, 1(3):244–274, 1991.

[14] I. Nikolaidis, R. Fujimoto, and C. A. Cooper. Parallel simulation of high-speed network multiplexers. *IEEE Conference on Decision and Control*, 3(1):2224–2229, 1993.

[15] I. Nikolaidis, R. Fujimoto, and C. A. Cooper. Time-parallel simulation of cascaded statistical multiplexers. In *SIGMETRICS '94: Proceedings of the 1994 ACM SIGMETRICS conference on Measurement and modeling of computer systems*, pages 231–240, 1994.

[16] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM '94*, pages 234–244, 1994.

[17] C. Perkins and E. Royer. Ad hoc On-demand Distance Vector Routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 99–100, 1999.

[18] VINT. The UCB/LBNL/VINT network simulator-ns (version 2). URL `http://www.isi.edu/nsnam/ns`.

[19] J. J. Wang and M. Abrams. Approximate time-parallel simulation of queueing systems with losses. In *WSC '92: Proceedings of the 24th conference on Winter simulation*, pages 700–708, 1992.

[20] J. J. Wang and M. Abrams. Determining initial states for time-parallel simulations. In *PADS '93: Proceedings of the seventh workshop on Parallel and distributed simulation*, pages 19–26, 1993.

[21] PDNS – Parallel/Distributed NS. URL `http://www.cc.gatech.edu/computing/compass/pdns/`, 2004.