

Challenges and Benefits of Time-Parallel Simulation of Wireless Ad Hoc Networks

Ladislau Bölöni, Damla Turgut, Guoqiang Wang, and Dan C. Marinescu
School of Electrical Engineering and Computer Science
University of Central Florida, Orlando, FL 32816-2450
{lboloni,turgut,gwang,dcm}@cs.ucf.edu

ABSTRACT

In this paper we discuss time-parallel simulation of wireless ad hoc networks. Such systems are rarely amenable to analytical modelling due to the complexity of the models involved. Thus, to obtain a quantitative evaluation of such models we often resort to simulation. Yet, current simulation techniques do not allow us to study systems consisting of thousands of nodes for extended periods of time.

We introduce the notion of perturbation of measurements and present a layer-by-layer analysis of the impact of perturbations on the functioning of the wireless network. This model allows us to predict the accuracy of the simulation for several measures of performance through an analysis of the protocols involved at the various layers.

We also present an implementation of time-parallel simulation based on iterative extension of the warmup period. In a series of experiments, we find good concordance between the predicted findings and the results of the simulations.

Categories and Subject Descriptors

H.4 [Information Systems Applications]: Miscellaneous; D.2.8 [Software Engineering]: Metrics—*complexity measures, performance measures*

1. MOTIVATION

Time-parallel simulation, the simulation of large systems by partitioning the time domain, rather than the space domain, has received significant attention in the past. The advantages, as well as, the theoretical and practical limitations of time-parallel simulation have been extensively researched for many applications. As a general rule time-parallel simulation requires the simulated process to be regenerative, a situation rarely encountered in practice. Yet, oftentimes we are satisfied with approximate results which could give us some qualitative rather than quantitative results important for the design of a system.

In this paper we discuss time-parallel simulation of wireless ad hoc networks. Such systems are rarely amenable to analytical modelling due to the complexity of the models involved. Thus, to obtain

a quantitative evaluation of such models we often resort to simulation. Personal networking technologies such as Bluetooth and Zigbee, allow the number of networked components in systems of interest to grow significantly. Current simulation techniques do not allow us to study systems consisting of thousands of nodes for extended periods of time.

Systems consisting of a few thousand nodes are rather common. Indeed, let us consider a simple scenario. In a university classroom, 120 students attend a class; each student has a cell phone (GSM source), a PDA and laptop (two 802.11b WiFi sources). There are five Bluetooth sources: PDA, laptop, cell phone, headset, and mouse. Some of the students might have WiFi enabled cameras, Bluetooth enabled audio players, and matching head phones. All in all, it does not seem out of the ordinary to have 3 WiFi and 7 Bluetooth sources per person. Thus, even without considering that many of the WiFi nodes have a transmission range long enough to cover neighboring classrooms as well, in order to study the networking environment of one classroom we have to simulate a system with 1200 wireless sources operating in the same frequency band.

A wireless network with 1200 nodes pushes the limits of serial simulators such as NS-2. Even when feasible, such simulations require a significant computing power and can take a very long time. An alternative is a parallel implementation of the simulation. Spatial partitioning based parallelism, however, is difficult to apply to our case. In the classroom environment, every transmission can be received by all the other nodes; the fully connected dependency defeats the purpose of spatial partitioning.

The question we pose is if an approximate time-parallel simulation of wireless ad hoc networks is feasible and if the quality of the results produced by such a simulation is acceptable. To analyze and predict the performance of time-parallel simulation of wireless ad hoc networks, we introduce the notion of perturbation of measurements and present a layer-by-layer analysis of the impact of perturbations on the functioning of the wireless network. This model allows us to predict the accuracy of the simulation relative to various measurements through an analysis of the protocols involved at each layer.

This paper is organized as follows. Related work is surveyed in Section 2. Our model of the propagation of perturbations in wireless networks, the impact of the protocols at the various layers of the networking stack and the application for time-parallel simulation is presented in Section 3. Our approach for time-parallel simulation is introduced in Section 4. A series of simulation studies investigating the speedup and precision of the proposed method for typical wireless network simulation scenarios are presented in Section 5. We conclude in Section 6.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Valuetools'06 October 11-13, 2006, Pisa, Italy.
Copyright 2006 ACM 1-59593-504-5 ...\$5.00.

2. RELATED WORK

Simulation along with analytical modelling allow us to carry out performance analysis studies of wireless ad hoc networks. Our studies cover a wide range of systems, from relatively simple systems composed of several nodes connected to a wireless access point to large wireless sensor networks consisting of thousands of nodes. As the size of the simulation problem increases and exceeds the capabilities of a single machine, we are forced to consider parallel simulation. Parallel discrete event simulation (PDES) reduces the overall execution time by parallel execution of the simulation on multiple processors. There are two main avenues for parallel simulation: space-parallel simulation (distributed simulation), and time-parallel simulation. In the space-parallel simulation approach [4,9], the simulation model is decomposed into a number of components on a spatial basis. Each component is modelled 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) [1] project uses a space parallel simulation approach based on the NS-2 network simulator [14]. 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 [2,5,6,8,15,16], a long simulation interval 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 [8], 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 many times. The system is then partitioned at the *regeneration points* which mark a *regenerative substate*. 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 pre-computation [10,11]. Wang and Abrams propose a *pre-simulation* to identify regenerative points based upon Markovian modelling [16]. Although time-parallel simulation rarely allows us to obtain accurate results, approximate results [6,15] can be produced efficiently.

3. MEASUREMENTS AND PERTURBATIONS IN WIRELESS AD HOC NETWORKS

We propose a model to investigate how an event *perturbs* the *measurements* of a simulation of wireless ad hoc networks. While our model might not amount to a rigorous formalism, it accurately captures the underlying phenomena and promotes a framework in which observations can be analyzed and predictions can be made.

We consider a simulation s specified by a geographic area A , a time interval $\tau = [t_{start}, t_{end}]$, an initial state S_{init} and a set of planned events $P = \{p_1 \dots p_n\}$. The result of the simulation process is the calculation of the *trace of the simulation* which is the set of events $\mathcal{T} = \{e_1 \dots e_m\}$ as well as the final state S_{final} . The trace includes the planned events ($P \subset \mathcal{T}$) as well as new events triggered by the planned events and calculated by the simulation process. Every event has a series of *properties*, where one with a special importance is the time when it happens: *time*(e_i). Other properties of the events are in general associated with various measurements we can make on the event (such as delay, packet size and

so on). We will note these measurements $M_1(e_i)$, $M_2(e_i)$ and so on.

The purpose of running a simulation is to gain access to the measurements of the events in the trace. In practice, however, we are not interested in the individual measurements of the events. Rather, we are interested in global measurements which span all the events in the trace $M_i(\mathcal{T})$.

Measurements can be:

(a) *instantaneous* $M^i(t_c)$, reflecting the situation at the current time t_c .

(b) *cumulative* $M^c(t_c)$, reflecting the evolution of the measurement from the beginning of the simulation to the current time t_c . For every instantaneous measurement M^i there is an associated cumulative measurement M^c :

$$M^c(t_c) = \int_0^{t_c} M^i(t) dt \quad (1)$$

(c) *average* $M^a(t_c)$. For every instantaneous measurement M^i there is an associated cumulative measurement M^a defined by:

$$M^a(t_c) = \frac{\int_0^{t_c} M^i(t) dt}{t_c} \quad (2)$$

An example is the current transmission rate (instantaneous measurement), with the associated cumulative measurement total transmitted data, and the associated average measurement average transmission rate.

We call an *approximate simulation for global measure* M , related to precise simulation s , a simulation s' which generates a trace \mathcal{T}' such that $M(\mathcal{T}') \approx M(\mathcal{T})$. We define the error of the simulation

$$\epsilon = \frac{|M(\mathcal{T}) - M(\mathcal{T}')|}{M(\mathcal{T})} \quad (3)$$

It is possible that a simulation results in a good approximation for certain measures of performance, but not for the others. The rationale for an approximate simulation is that it is easier to obtain the results, the simulation is faster, easier to parallelize, and so on.

Now we introduce the notion of a *perturbing event*. We consider a simulation with the planned events $P = \{p_1 \dots p_n\}$ and a perturbed set $P' = P \cup p_p$ where p_p is the perturbing event. The perturbed set of planned events will lead to a simulation trace $(\mathcal{T})'$, and obviously, perturbed measurements $M'(t)$. In the following, when we are talking about perturbations, we mean the impact of a perturbing event on a set of measurements, $\Delta M(t) = M'(t) - M(t)$. We are interested in the size and the temporal extent of the perturbations.

We say that:

- The perturbation has no effect on measurement M if $\Delta M(t) = 0 \forall t$.
- The event e_p creates a time limited perturbation in the measurement M which lasts until the *extinction time* t_e if $\Delta M(t) = 0 \forall t > t_e$.
- The event e_p has a shift effect perturbation on the measurement M if $\lim_{t \rightarrow \infty} (\Delta M(t)) = c$, with c being the *shift constant*. If there is a time point t_s with the property that $\Delta M(t) = c \forall t > t_s$, we call the t_s *stabilization point*.

PROPERTY 1. *If an event e_p causes a time-limited perturbation on the instantaneous measurement M^i with extinguishing time t_e*

it creates a shift effect perturbation on the corresponding cumulative measurement M^c , with the stabilization point $t_s = t_e$. The shift constant can be expressed as:

$$c = \int_{t_p}^{t_e} \Delta M^i(t) dt \quad (4)$$

We say that an event causes a destabilizing perturbation of measurement M if $\exists t_s > t_p$ such that $\Delta M = c \forall t > t_s$.

3.1 Perturbations and measurements in wireless ad hoc networks

As a first approximation, the events in a wireless ad hoc networks are caused by individual transmissions of packets. We consider the perturbing event to be the insertion of a new packet in the network. We do not consider separately the removal of the packet because that is equivalent to simply reversing the perturbed and the original system in the discussion. As we are concerned about the absolute values of the difference on the measurements of these systems, the reversed system will yield the same conclusions.

An additional type of perturbation we consider is the perturbation of the initial state - that is the simulation starts with a different initial state than expected. We discuss this type of perturbation separately.

For events representing the addition or removal of a single packet, the immediate affect of the perturbations is usually minimal. However, the different layers of the networking stack are affected by the packet in various ways, which can trigger significant changes in the network. We investigate the impact of the perturbation at the various network layers.

3.1.1 Physical layer

A new packet will perturb the physical layer measurements only for the duration of the packet.

The length of a packet can be calculated as follows. Assume that we send a packet of 1536 bit, the maximum length supported by the 802.11b protocol. The transmission rate of 802.11b is 1.375 Mbps. The time required to send this packet is composed of:

- DIFS–Distributed InterFrame Space, set to $50\mu s$
- Data packet transmission: $192\mu s$ for the preamble + $1536 / 1.375 \text{ Mbps} = 192\mu s + 1118\mu s$
- SIFS–Small InterFrame Space, set to $10\mu s$
- 802.11 ACK packet: $192\mu s + 14 / 1.375 \text{ Mbps} = 203\mu s$

Thus, the total time becomes 2084μ , or approximately, 2 ms. This might change slightly for different protocols, but the order of magnitude remains the same.

We conclude that the perturbation in the physical layer due to a new packet is time-limited, with a very short (2 ms) extinction time.

3.1.2 MAC layer

The effect of a new packet perturbation over the MAC layer depends on the load of the network, and the type of the new packet. If the packet is significant for the MAC protocol itself (e.g., ACK, RTS, or CTS packet) its influence extends beyond the time frame covered. For instance, an RTS packet make the nodes receiving refrain from transmitting for the duration specified in the packet. The maximum order of magnitude of this interval is the maximum packet transmission time; its order of magnitude is $2\mu s$.

An additional effect, however, is that the delay on the sending of one node can then delay the transmission of another packet, creating a ripple effect.

A rule of thumb is that for a channel of n bps, with a load of $\nu \leq 1$

$$t_e = \frac{\text{PacketSize}}{\text{ChannelCapacity}} \cdot \frac{1}{1-\nu}$$

Note that for a full channel, $\nu = 1$, there is no extinction time.

If we are considering a network which cannot avoid collisions, the perturbation will extend to the length of the contention window. 802.11 uses an exponential backoff algorithm where the contention window can vary between CVM_{\min} and CVM_{\max} . Typical values of CVM_{\min} are between 7-15 and of CVM_{\max} are between 7-255 with the numbers being multiples of a slot time which is $20\mu s$ in 802.11b. Thus, the influence of initial collision can extend to $5000\mu s$. However, if the channel is very busy, the initial collision can lead to further collisions down the line. Furthermore, any collision is extending the collision window through exponential backoff, which will be only gradually reduced.

In conclusion, on highly loaded networks, an introduction of a packet creating a collision can yield to perturbations of the order of magnitude of several seconds in the worst case.

In summary, the TDMA (Time Division Multiple Access) style Medium Access protocols lead to the smallest perturbation levels, followed by CSMA-CA (Carrier Sense Multiple Access with Collision Avoidance). CSMA-CD (Carrier Sense Multiple Access with Collision Detection) shows the most significant perturbations in the worst case.

3.1.3 Routing layer

Perturbations triggered by packet insertion at the routing layer are heavily dependent on the nature of the protocol and the newly considered packet. The effect of the inserted packet was considered at the lower layers. Here, we will consider whether the packet will change the routing of future packets or not. There are several cases to consider. If the packet is part of an established flow of application layer data, it will most likely not affect the routing of the other packets. If the packet is the first packet of a new flow, and the routing protocol is a reactive one, the packet will establish a new route. This frequently requires broadcast of routing information, which in its turn trigger additional packets being transmitted all over the network. Although, this appears to be a destabilizing factor—and a destabilizing perturbation, in reality, the extent of this flooding is carefully constrained by the routing protocol. Thus, the perturbation will be still limited in time. This perturbation interval will last at most several seconds. As we have mentioned before, the process will not be repeated for the future packets of the same flow.

Proactive routing algorithms deploy a routing table which is updated proactively by routing packets. The insertion of a routing packet triggers a perturbation in the network by changing the routing table of a node, and this will affect the routing of the future packets. This is a major perturbation, affecting a large number of packets and a long time interval.

To evaluate the extent of the perturbation, let us consider the ways in which the extinction of the perturbation can happen. A routing table perturbation is extinct when the routing tables of the original and the perturbed system re-converge. This situation occurs when:

- The modified routes are superseded by new, independently discovered routes, in both the original and the perturbed system (for instance, as a result of node mobility).

- The original system acquires the same routes as the modified one.
- The modified routes expire through a timeout and the routing table returns to the unperturbed version.
- An external command or a predetermined timeout flushes partially or completely the routing tables, forcing the recomputing of all routes.

Although it is technically possible to imagine a routing protocol where the loss or addition of a single routing packet would change the routes indefinitely, virtually all the proposed protocols contain methods which will ultimately allow the routing tables to converge.

We conclude that for proactive routing protocols the extent of the perturbations cover a longer interval, of the order of minutes. The protocol designers, in their quest to make the protocols more reliable, have adapted features which make the routing tables converge; this has the indirect effect of limiting the perturbations and improve the accuracy of time parallel simulations.

3.1.4 Transport layer and application layer

Finally, we investigate how a perturbing event affects the transport layer and the application layer protocols. The major difference here is between reliable or non-reliable protocols.

Let us consider the case of the most frequently used reliable transport protocol, TCP. The loss of a single TCP packet can significantly perturb the nature of the subsequent TCP flow: the packet will be retransmitted, the transmission window reset to its minimal value, which will then extend through the slow start algorithm. Thus, the loss of a single packet can exert an influence over the network of several seconds. Conceivably, there can be side effects as well, through which a change in one TCP transmission might affect other transmissions on the network.

In the case of the UDP protocol, which does not implement reliable transmission, the perturbation is minimal or nonexistent. It is however, incorrect to say that the loss of a UDP packet does not introduce any perturbation - applications which deploy UDP usually use application layer protocols to control the flow of data. For instance, the multimedia streaming application RealPlayer is using an application level byte stream protocol RTP (specified in RFC 1889) for the transfer of multimedia information, with UDP being the transport protocol. Packet losses are handled by a complex logic and actions involving the RTSP (Real Time Streaming Protocol), RTCP (Real Time Control Protocol), SDP (Session Description Protocol) and, of course RTP.

We conclude that perturbations have effects up to the application layer. However, at the application layer, the behavior of the system becomes very complex, and in fact it can depend on user interactions.

In research involving ad hoc networks, most of the experimental work stop at the level of routing protocols, the consideration of the transport layer is an exception rather than the rule.

3.2 Perturbations and time-parallel simulation

The analysis from the previous section can be applied to predict the approximation level of time-parallel simulations. Let us summarize the conclusions:

- The largest perturbations are the ones affecting the routing tables.
- There is limited benefit in making a simulation batch smaller than the longest perturbation.

- Perturbations are amplified in highly loaded systems.
- Time parallel simulations can be more useful in the study of the physical and the MAC layer than the routing and the transport layer.
- Time-parallel simulation of reactive routing protocols yields higher precision than that of proactive routing protocols.

3.3 Techniques for improving the accuracy of time-parallel simulations

Time parallel simulation is dividing the simulation in a set of *batches* which correspond to time intervals. The most straightforward way to partition a simulation is by partitioning the time into batches of equal length. For instance, in Figure 1 (a) we see a partitioning of the simulation interval into three equal time intervals: $[0, t_1 = t/3]$, $[t_1 = t/3, t_2 = 2t/3]$ and $[t_2 = 2t/3, t]$. The figure also illustrates a potential source of inaccuracy of the simulation: the perturbations due to two events are overflowing from one batch to the other. However, as the processor working on the second batch has no way of knowing about event e_1 its simulation will be inaccurate.

Time interval shift. One way to increase the accuracy of the simulation is to select the borders of the simulation intervals such that the perturbations are completely contained in the batches, as shown in Figure 1(b). These points are the equivalent of the regenerative states of the queuing systems. If we can not find states where all the perturbations are extinguished, we can search for points where there are a comparatively smaller number of ongoing perturbations (partial regenerative states).

There are several difficulties with this approach. First, there might not be any regenerative states in the simulation, or their distribution might be such that it does not lead to batches of size appropriate for parallelization. The most difficult problem, however is, that it is very difficult to find regenerative or partially regenerative states in a simulation without first running the simulation itself. There are certain circumstances when the identification of such points is possible. If a routing protocol periodically flushes the complete set of routing information, that timepoint is a natural partial regenerative point.

Warmup interval. As time interval shift is possible only in limited circumstances, we consider a different approach. For each batch we will consider two time intervals: during the warmup time, we perform the simulation but do not perform any measurements, which during the measurement phase we perform measurements, which should have a higher degree of precision, due to the fact that some perturbing events are considered in the warmup phase. Let us consider the appropriate size of the warmup window. This window needs to be the minimum size such that it contains all the perturbing events which generate perturbations which extend beyond the start of the measured interval. The Figure 1(c) represents such a case. The size of the required warmup period can vary between various batches, the first batch does not require a warmup. Unfortunately, for practical cases we can not accurately compute the size of the required warmup time without first running the simulation.

In practice, for the simulation study discussed in this paper, we set up all the batches with a warmup period of identical length. Furthermore, we will gradually increase the size of the warmup period, obtaining more and more accurate simulations.

Warmup with compressed history. In a practical run of time-parallel simulation with warmup, the size of the warmup period can be significantly longer than the measured interval; thus most of the computation time is spent into the simulation of the warmup interval, which does not contribute to the measurements. Traditionally,

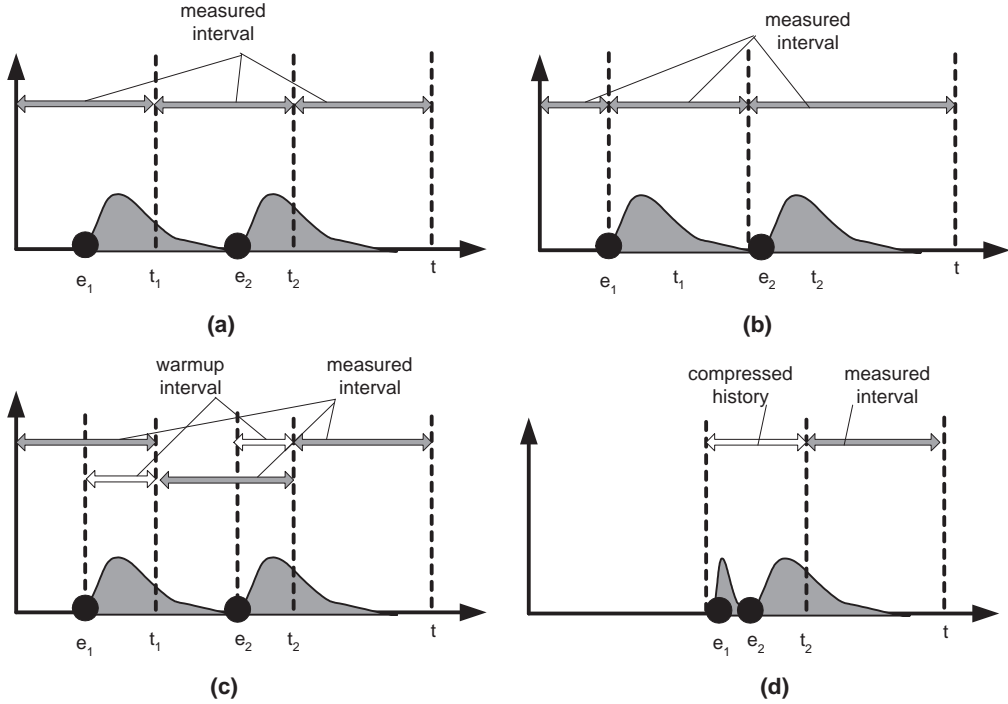


Figure 1: An illustration of techniques for improving the accuracy of time parallel simulations. (a) Unimproved equal length batches. (b) Time interval shift. (c) Warmup intervals. (d) Warmup with history compression.

the warmup interval is the exact match of the simulated events for a period before the measured interval starts. We can replace the warmup interval with a shorter and/or simpler simulation interval which, however, would yield the same results. For instance, we can remove all the events from the warmup period which do not produce perturbations at the measured point. For the events which produce perturbations, we might be able to replace them with events easier and faster to simulate. We call this modified warmup interval a *compressed history*.

4. AN ALGORITHM FOR TIME-PARALLEL SIMULATION

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 $\{\tau_i | \tau_i = [(i-1)\frac{D(\tau)}{m}, i\frac{D(\tau)}{m}], i \in \{1, \dots, m\}\}$, where $D(\tau)$ calculates the duration of time interval τ . Then, we obtain a time-parallel simulation set $I^{(0)} = \{S_i | S_i = (A, \tau_i, E(S_i), \phi, \mathcal{F}(S_i), \mathcal{T}(S_i)), i \in \{1, \dots, m\}\}$. 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} , $i \in \{1, \dots, m-1\}$. However, since there is no prior knowledge for the initial states of S_2, \dots, S_m when they are parallelized, the combined simulation trace $\mathcal{T}(I^{(0)}) = \mathcal{T}(S_1) \cup \mathcal{T}(S_2) \cup \dots \cup \mathcal{T}(S_m)$ 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 $I^{(1)} = \{S_i^{(1)} | S_i^{(1)} = (A, \tau_i, E(S_i^{(1)}), \mathcal{F}(S_{i-1}), \mathcal{F}(S_i^{(1)}), \mathcal{T}(S_i^{(1)})), i \in \{2, \dots, m\}\}$. 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, $\mathcal{T}(I^{(1)}) = \mathcal{T}(S_1) \cup \mathcal{T}(S_2^{(1)}) \cup \mathcal{T}(S_3^{(1)}) \cup \dots \cup \mathcal{T}(S_m^{(1)})$, 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 threshold. The k -th iteration contains $m-k$ simulation segments $I^{(k)} = \{S_i^{(k)} | S_i^{(k)} = (A, \tau_i, E(S_i^{(k)}), \mathcal{F}(S_{i-1}^{(k-1)}), \mathcal{F}(S_i^{(k)}), \mathcal{T}(S_i^{(k)})), i \in \{k+1, \dots, m\}\}$. After k -th iteration, the traces of simulation intervals $S_1, S_2^{(1)}, \dots, S_{k+1}^{(k)}$ become accurate. The combined simulation trace of $I^{(k)}$ can be obtained by

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

We illustrate the approximate simulation in the following example, see Figure 2. Assume a simulation $S = (500 \times 500, [0, 200], E, \phi, \mathcal{F}, \mathcal{T})$ is segmented into 10 equal-duration time intervals, with $\tau_i = [20(i-1), 20i], i \in \{1, \dots, 10\}$. 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
 $\mathcal{T}(I^{(0)}) = \mathcal{T}(S_1) \cup \mathcal{T}(S_2) \cup \mathcal{T}(S_3) \cup \dots \cup \mathcal{T}(S_{10})$;
 $\mathcal{T}(I^{(1)}) = \mathcal{T}(S_1) \cup \mathcal{T}(S_2^{(1)}) \cup \mathcal{T}(S_3^{(1)}) \cup \dots \cup \mathcal{T}(S_{10}^{(1)})$;
 $\mathcal{T}(I^{(2)}) = \mathcal{T}(S_1) \cup \mathcal{T}(S_2^{(2)}) \cup \mathcal{T}(S_3^{(2)}) \cup \dots \cup \mathcal{T}(S_{10}^{(2)})$.

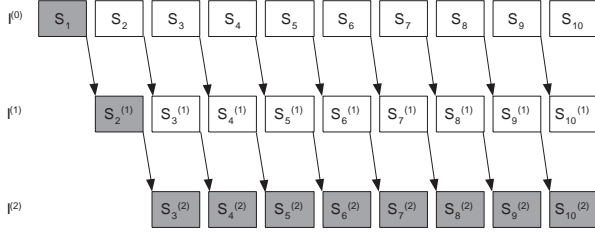


Figure 2: Illustration of the time-parallel simulation approach. The duration of the simulation S is 200. It is segmented into 10 equal-duration simulation intervals. The shaded simulation segments are used to compose the final simulation trace after 3 iterations.

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, RST, 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 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:

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

The time interval η_i starts at a 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):

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

$P_{i,j}$ simulates the same simulation interval as S_{i+j} . For instance, in Figures 2 and 3, $P_{3,2}$, $P_{4,1}$ and $S_{5,0}$ all simulate the simulation interval [80, 100].

We assume the same simulation S in Figure 2 as our example, $m=10, \gamma=2$. The new simulation set $L = \{P_1, P_2, \dots, P_{10}\}$ is shown in Figure 3. 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 2 and 3. 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

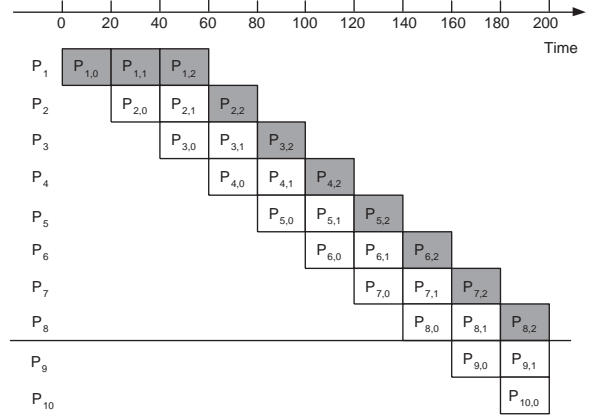


Figure 3: An alternative approach to the simulation in Figure 2. Shaded blocks correspond to partial results used to construct the final simulation results.

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.

5. 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 [3, 7] 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 the speed. Table 1 shows the default settings and the range of the parameters for our experiments.

5.1 Performance Metrics

To establish the accuracy of our time-parallel simulation relative to an exact sequential simulation, we evaluate the *relative error* for several measurements. Let M_0 be the result produced by the exact

Table 1: The default values and the range of the 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)	

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. Notice that both of these are average measurements.

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) [12] and Ad Hoc On-Demand Distance Vector Routing (AODV) [13] 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.

5.2 Simulation results

Segment Duration. This set of experiments (Figure 4) 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 4 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 are a 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 5) 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 5)

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

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 6, we can see that for both protocols the relative error for the metrics we investigate fluctuates around a relatively stable value regardless of the workload. The results are increasingly more accurate as the 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 conclude that *the performance of the parallel simulation is not sensitive to the network load, or to 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 minutes the maximum number of nodes is 200 for the sequential and 1200 for the parallel simulation.

6. CONCLUSIONS

In this paper we explore the challenges and the benefits of time-parallel simulation of wireless ad hoc networks. We introduce the notion of perturbation of measurements and present a layer-by-layer analysis of the impact of perturbations on the functioning of the wireless network. This model allows us to conclude that:

- The largest perturbations are the ones affecting the routing tables.
- There is limited benefit in making the simulation batch smaller than the longest perturbation.
- Perturbations are amplified in highly loaded systems.
- Time parallel simulations can be more useful in the study of the physical and the MAC layer than the routing and the transport layer.
- Time-parallel simulation of reactive routing protocols yields higher precision than that of proactive routing protocols.

Based on this investigation, we find ways to predict the accuracy of a time-parallel simulation, and propose several methods such as

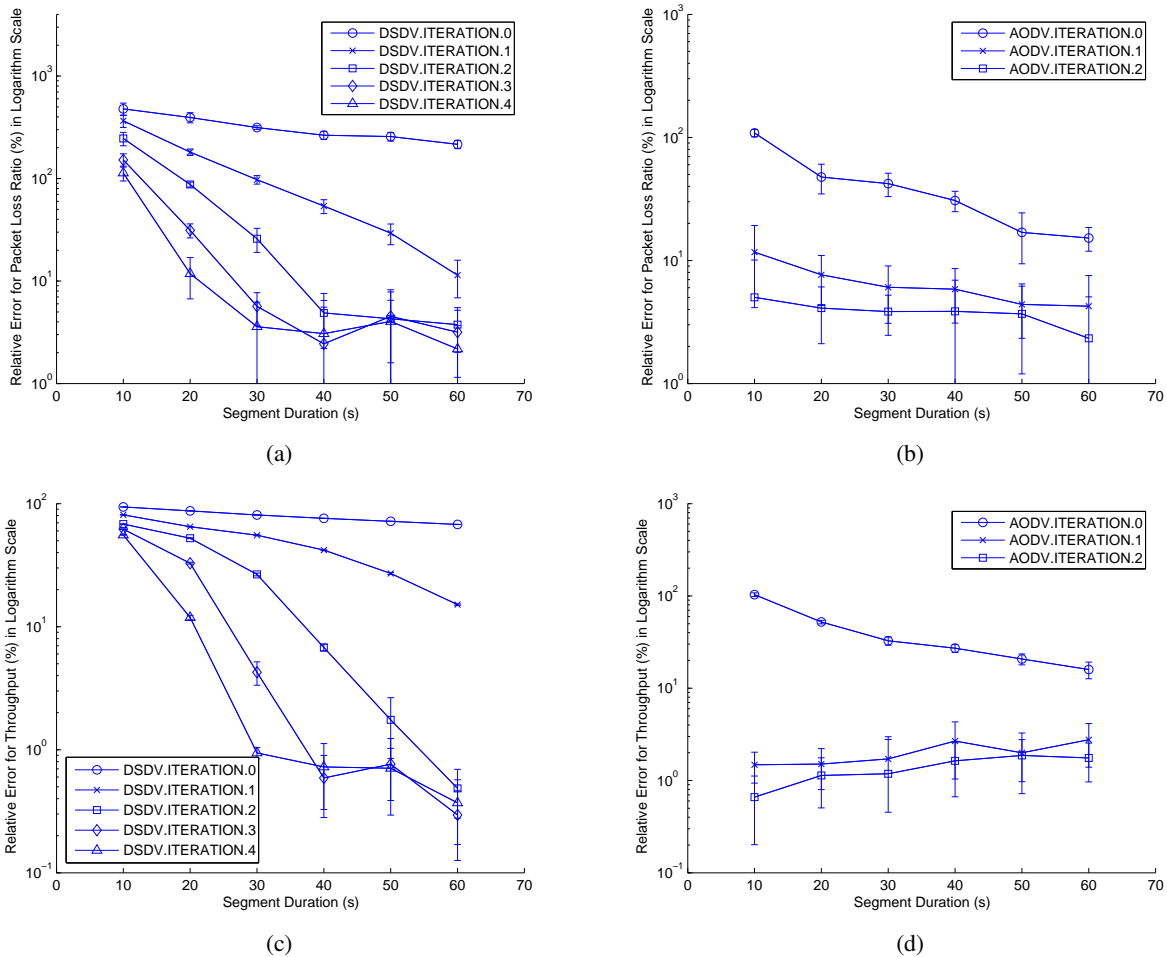


Figure 4: Relative error for packet loss ratio and throughput as function of the segment duration in a logarithmic scale; DSDV (left) and AODV (right). (a) and (b) show the relative error for the packet loss ratio; (c) and (d) show the relative error for the throughput.

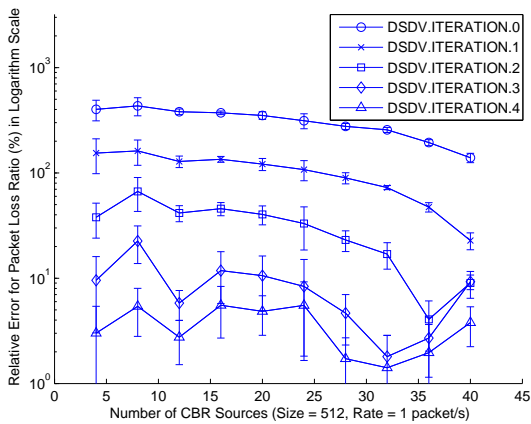
interval shift, warmup, and warmup with history compression to improve the quality of the results. Finally, we present an implementation of the time-parallel simulation using equal intervals and warmup periods of variable length. We study the accuracy of simulated results for packet loss ratio and throughput for two popular ad hoc routing algorithms: AODV and DSDV. The results of the simulations are in good concordance with the predictions made through the model. We conclude that time-parallel simulation allows us to study relatively large wireless ad hoc networks consisting of a few thousand nodes for extended periods of time.

Acknowledgment

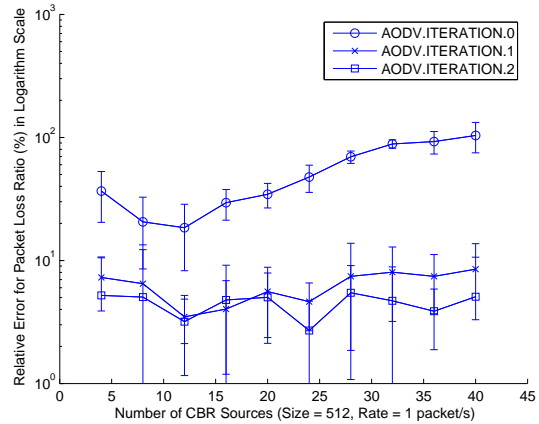
The research reported in this paper was partially supported by National Science Foundation grants ACI0296035 and EIA0296179.

7. REFERENCES

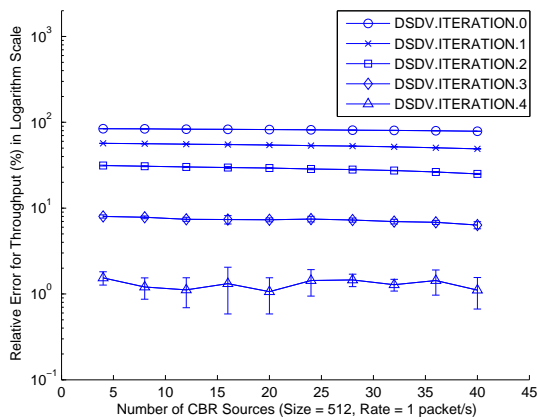
- [1] PDNS – Parallel/Distributed NS. URL <http://www.cc.gatech.edu/computing/compass/pdns/>, 2004.
- [2] S. Andradóttir and T. J. Ott. Time segmentation parallel simulation of networks of queues with loss or communication blocking. *ACM Transactions on Modeling and Computer Simulations*, 5(4):269–305, 1995.
- [3] J. Broch, D. A. Maltz, D. B. Johnson, Y. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. Mobile Comp. & Networking*, pages 85–97, 1998.
- [4] R. M. Fujimoto. Parallel discrete event simulation. *CACM*, 33(10):30–53, 1990.
- [5] M. Hoseyni-Nasab and S. Andradóttir. Parallel simulation by time segmentation: Methodology and apps. In *Proc. 1996 Winter Sim. Conf.*, pages 376–381, 1996.
- [6] M. Hoseyni-Nasab and S. Andradóttir. Time segmentation parallel simulation of tandem queues with manufacturing blocking. In *Proc. 1998 Winter Sim. Conf.*, pages 181–186, 1998.
- [7] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. chapter 5, pages 153–181. Kluwer Academic, 1996.
- [8] 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.
- [9] V. K. Madiseti, J. C. Walrand, and D. G. Messerschmitt. Asynchronous algorithms for the parallel simulation of event-driven dynamical systems. *ACM Transactions on*



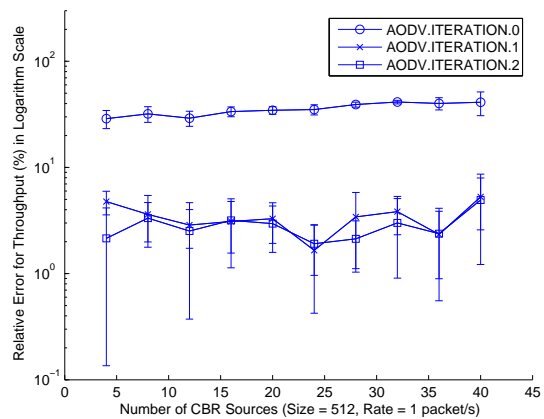
(a)



(b)



(c)



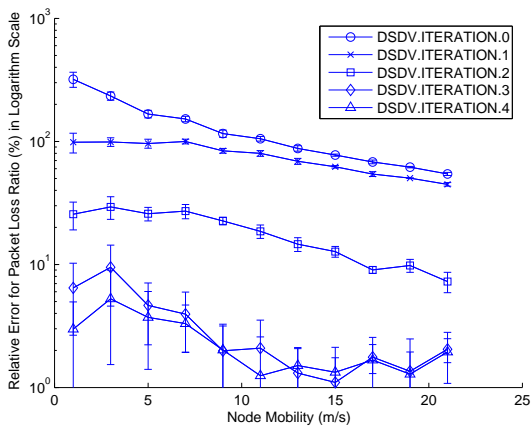
(d)

Figure 5: 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; (c) and (d) show the relative error for the throughput.

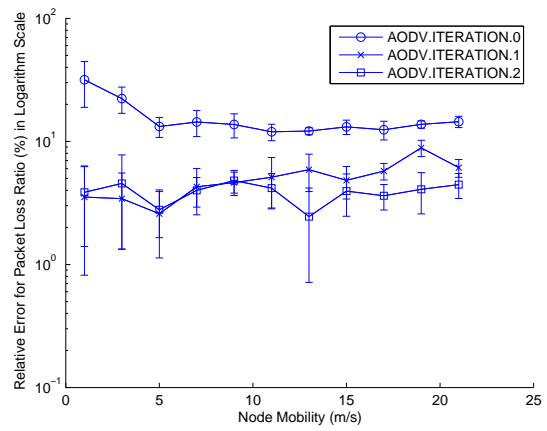
Modeling and Computer Simulations, 1(3):244–274, 1991.

- [10] I. Nikolaidis, R. Fujimoto, and C. A. Cooper. Parallel simulation of high-speed network multiplexers. *IEEE Conf. on Decision and Control*, 3(1):2224–2229, 1993.
- [11] I. Nikolaidis, R. Fujimoto, and C. A. Cooper. Time-parallel simulation of cascaded statistical multiplexers. In *Proc. SIGMETRICS '94*, pages 231–240, New York, NY, USA, 1994. ACM Press.
- [12] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *SIGCOMM '94*, pages 234–244, 1994.
- [13] C. Perkins and E. Royer. Ad hoc On-demand Distance Vector Routing. In *Proc. 2nd IEEE Workshop on Mobile Comp. Sys. & Apps*, pages 99–100, 1999.
- [14] VINT project. the ucb/lbnl/vint network simulator-ns (version 2). URL <http://www.isi.edu/nsnam/ns>.
- [15] J. J. Wang and M. Abrams. Approximate time-parallel simulation of queueing systems with losses. In *Proc. 24th Winter Sim. Conf.*, pages 700–708, New York, NY, USA, 1992. ACM Press.
- [16] J. J. Wang and M. Abrams. Determining initial states for time-parallel simulations. In *PADS '93: Proc. 7th Workshop*

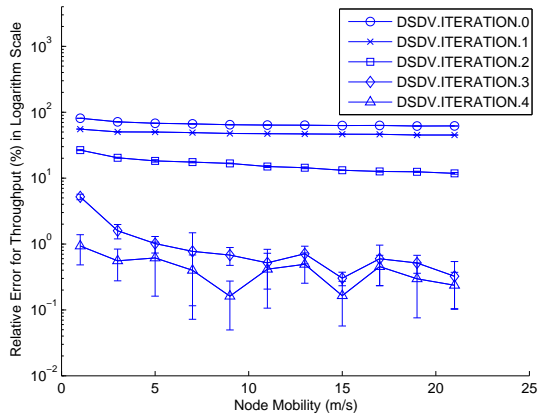
on Parallel & Distrib. Sim., pages 19–26, New York, NY, USA, 1993. ACM Press.



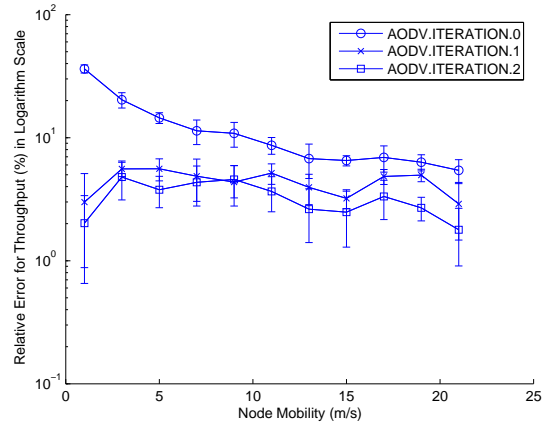
(a)



(b)



(c)



(d)

Figure 6: 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; (c) and (d) show the relative error for the throughput.