Rapid Simulation of Wireless Systems

L. Felipe Perrone
Dept. of Computer Science
College of William and Mary
perrone@cs.wm.edu

David M. Nicol
Dept. of Computer Science
Dartmouth College
nicol@cs.dartmouth.edu

Abstract

The simulation of wireless systems such as cellular or personal communication systems comprises both discrete and continuous time processes. To accelerate these simulations, we propose the use of interval jumping*, a novel technique that allows the execution of a continuous time model to proceed in irregularly sized jumps rather than in the traditional time-stepped manner. The foundations for this mechanism are laid out in the light of the simulation of a complex simulation model which includes radio propagation, channel allocation, transmitter power control and user mobility. We conclude with experimental results comparing sequential and parallel execution of these accelerated simulations which indicate the good potential of our technique.*

1. An Overview of Wireless Systems

Cellular phone systems, paging networks and personal communication systems (PCS) are particular instances of the broader class of *wireless systems*. A central idea common to these is that the spatial domain is divided into smaller areas of service called *cells*. Within each cell, mobile subscribers use low power radio transceivers to connect to a server positioned at its center. The servers route user calls within the wireless system and/or to a conventional wired network.

The radio link used can be analog, digital or hybrid, supporting both types of signals. In any case, some means must be used to provide server access to multiple subscribers at the same time. Depending on the nature of the radio signal, different technologies are available, such as Time Division Multiple Access (TDMA), Frequency Division Multiple Access (FDMA), etc.

Regardless of the implementation of multiple access, radio connections must use the lowest possible transmission power. The same channel may be used by a number of cells simultaneously and low power helps minimize *cochannel interference* allowing a certain *quality of service* (QoS) to be achieved. This is the concept of *channel reuse*, one of the central ideas behind the development of wireless systems.

The QoS of a system is defined in terms of the ratio of strength of the signal generated by a transmitter measured at the receiver's geographic location to the total noise perceived at the same point. This ratio is commonly referred to as signal-to-noise ratio (SNR). Service providers determine that a connection is only as good as its SNR: it is preferable to not offer or interrupt a connection if the clarity of signal does not achieve a desired standard.

Other measures of importance in wireless systems are *blocking probability*, the likelihood that a call attempt fails due to system saturation, and *drop probability*, the likelihood that a call is interrupted. A call, may be blocked either if the cell that provides service has no more free channels or has no free channel that can perform at the desired SNR. A call may be dropped when the user moves outside the current service area into another that has no free channels or if no channel can carry it at the target SNR. We can say that the three values, SNR, blocking and drop probabilities, define an overall performance measure of a wireless system.

It is difficult to predict the performance of a specific wireless system because there are no analytical models that embody the complicated dynamics of all its components, what justifies the use of computer simulation. The models for these systems, however, are usually so complex and big that simulation times can easily reach an order of tens of hours.

In this work we identify a major bottleneck in these simulations and propose a new technique to overcome it. Section 2 presents a brief discussion of related work to situate this effort within the literature. Section 3 defines our simulation model and basic terminology. Section 4 introduces *interval jumping* and sections 5 and 6 present experimental results with sequential and parallel simulations, respectively.

2. Related Work

The latest years have witnessed the publication of several articles on parallel simulation of wireless systems. The level of abstraction in the models used in these works varies according with the depth and completeness expected from the simulation studies.

In [8] one finds a valuable starting point: the authors introduce issues of Parallel Discrete-Event Simulation at the same time that a detailed model for PCS simulation is presented, providing a simultaneous tutorial on both topics.

In [3], models for teletraffic and simplified mobility are used under a static channel allocation (SCA) scheme to simulate, in parallel, realistically sized PCS networks. Later articles [1,2] carry on with a more detailed mobility model in order to study the effect of different call workloads and population movement patterns on the system's performance. All these studies were conducted using Time Warp.

A class of models for dynamic channel assignment (DCA) is identified in [6] and a conservative technique for accelerating the simulation on a massively parallel SIMD machine is presented. Using a large cellular network model, experiments are executed on a 16K processor MasPar MP-1 achieving speedups up to 120 with teletraffic simulation.

Our work differs from those mentioned above at different levels. One of the motivations behind this study was to construct a more complete simulation model including the tightly coupled components of wireless systems. With this model, which brings simulations closer to real systems, we devised a technique offering good potential for performance improvements and started studying issues related to its parallelization.

The reader well versed in the area will notice that some ideas explored here are closely related to those presented in [7]. First, we share the same rationale for development of a parallel simulation model. Furthermore, as detailed in Section 6, we agree that the natural partitioning strategy for this problem is to break it up by channels and not by subdividing the simulated space. Each channel or pool of channels can be simulated by a different processor fairly independently of others, specially when adjacent channel interference is minimal. In addition to the new technique for acceleration proposed, our work differs from [7] by including cross-channel interference in the model, and analysis of how this impacts synchronization in a parallel simulation context.

3. A Simulation Model

Wireless systems subdivide the area of coverage into smaller regions, or cells, conceptually modeled by hexagons or squares. Each cell provides simultaneous full-duplex service to a number of subscribers equal to the number of transceivers its server contains.

We consider our model's geographical space partitioned into regular hexagons, all of the same size. The cells are arranged as a torus to avoid the influence of edge-effects in a fashion similar to that described in [9].

Signals propagate in this space subject to a law of exponential decay with distance so that a receiver at d units of linear distance from a transmitter operating with a power level p_t perceives a received power p_r of:

$$p_r \;=\; p_t\, a \;=\; p_t\, \frac{1}{d^\alpha} \tag{1}$$

where a is the attenuation coefficient in the propagation path and α, usually set to 4 to account for obstacles, determines a fast attenuation with distance. The propagation model may include a **lognormal** factor in the numerator of the attenuation coefficient [4,7], re-sampled as time progresses, to account for transient changes in propagation conditions. We omit this factor from our model for the time being.

We abstain from defining a multiple access technology at this point understanding that we could add an extra layer to the model later, if necessary. Instead, we work with the conceptual notion of *channel*: the medium capable of implementing a full-duplex connection between a user and the server at the center of a cell. Each channel can be seen as composed of two radio links: one that takes signals originating in a mobile transmitter to a cell server, called *uplink*, and another, called *downlink*, to carry signals from a cell server to a mobile receiver. Channels may be orthogonal (no cross-channel interference) or not.

Power control is described by a differential equation in continuous time, or for computational purposes, as a finite differences equation in a discretized timescale. Our simulation model uses the power control law from the *Local Autonomous Dynamic Channel Allocation* (LADCA) algorithm in [4,5]. If $p_i^c(k)$ is the power level of transmitter i on channel c at time k, we can write this law as:

$$p_i^c(k+1) \;=\; (1-\beta)p_i^c(k) + \left[\beta\rho\frac{v_i^c(k)}{a_{ii}(k)}\right] \tag{2}$$

where β is a convergence parameter, ρ is the target SNR, v_i^c is the total noise perceived by the receiver listening to transmitter i and a_{ii} is the attenuation coefficient in the propagation path between transmitter and receiver. The set of all transmitter power levels, $\{p_i^c(k) : \forall i \forall c\}$, is often called a *power assignment*.

Equation 2 represents a fixed-point computation developed for a model with stationary users. It guarantees that for an arbitrary spatial arrangement of users, the optimal power assignment (power allocations only large enough to satisfy a target SNR ρ) is computed after a small number of iterations. We assume that for a model including user mobility, the same power control law can be used so that the optimal solution for power allocation can be continuously tracked, if not achieved, as time progresses. Clearly, this assumption is only valid if the portables' speeds are small in comparison with the rate at which power levels are updated.

A subscriber generates call requests for the wireless system according to an arbitrary random distribution of call inter-arrival times. The length of each call is determined by another random variable with arbitrary distribution. Subscribers have unrestricted movement in the simulation space as long as their trajectories are composed by segments of straight line and that in each segment the mobile has constant velocity.

When a subscriber attempts to place a call, the system must decide which cell will serve it; currently, our model chooses the cell containing the initiating portable. If the chosen cell has no free channel, the call is **blocked**, otherwise it goes into the *probing phase* of the LADCA algorithm. First, a channel is randomly chosen and the transmission of a pilot signal is started while the system monitors the evolution of transmission power. If the target SNR can be met for a number Γ of consecutive time-steps, the call enters *service phase.* If the transmitter power reaches its maximum during probing, the call is switched to another randomly chosen channel. If all channels are visited and none found where the target SNR can be met, the call is **blocked**.

Once a call enters *service phase*, if its channel becomes so noisy that, even at maximum transmitter power, the target SNR cannot be achieved for a number Ψ of consecutive time-steps, the call is switched to another channel chosen at random. If all channels have been visited and none could meet the target SNR, the call is **dropped**.

Since subscribers roam the service area, a user can leave the boundaries of the cell currently providing service while a call is in progress. This defines a **hand-off**: a new cell and perhaps a new channel must be chosen to continue carrying the call. If the chosen cell has no free channel to offer, the call is **dropped**. We detect hand-offs checking a portable's position against the boundaries of the cell that contains it.

Throughout the duration of the call, probe and service phases included, transmitter power levels are constantly changing according to the power control algorithm. Its goal is to minimize power usage and thus it must strive to keep levels only so high as to meet but not surpass the target SNR.

A key point of interest in this model is that the continuous process described by Equation 2 can cause discrete-events (e.g. a change in channel) and the discrete events can affect the continuous model (e.g. a new call arrival). The goal of our work has been to accelerate the simulation of such models.

4. Interval Jumping

The time-stepped simulation of our model can be outlined by the following algorithm, where Δ is a time increment small enough to guarantee power control convergence:

```
   while (!simulation_done) do
1:     compute noise
2:     update power levels
3:     apply LADCA algorithm
4:     update user positions
5:     advance time by Δ
   end while
```

where steps 1 through 3 iterate over each element in the set of channels defined for the system. One can view this as several one-channel system simulations interacting only when a call switches channels.

The choice of Δ bears heavy impact in the execution time of the simulation: a large value could impede the convergence of power control while a value too small would slow down the simulation unnecessarily. We assume a value of Δ compatible with the sampling rate used by the real system (down to the electronic devices level), a value with an order of magnitude of 10^{-2} seconds.

Next, consider what is involved in the computation of noise in this model, assuming that the number of users in the system is N and the number of channels is C. Brute-force computation of the total interference on a receiver i tuned to a channel c requires the summation of a number of pairwise interactions: we must add the contributions of every other transmitter tuned to c and also all others tuned to adjacent channels (assumed to be no greater than 2). Given an even distribution of transmitters to channels, the per-mobile cost is $O(N/C)$; considering all mobiles the total cost is $O(N^2/C)$. This cost might be reduced using N-Body algorithms [10]. In our experiments we limit the cost by restricting the subdomain considered to affect a cell; this is for computational expediency and exploration of interval jumping rather than knowledge that it is numerically acceptable.

Observing the numerical algorithm embodied in Equation 2, we see that in its core, it is equivalent to a modest integration by a trapezoidal rule with equally spaced abscissas. Numeric integration has evolved much past that point and so can this simulation.

Once one realizes that during subintervals of simulated time power levels may change slowly, it becomes clear time could be advanced in irregular, adaptive steps. The technique we propose is based on identifying such subintervals and exploiting this very idea.

Equation 2 shows that the driving forces behind transmitter power levels are v_i, the interference over the receiver listening to transmitter i and a_{ii}, the attenuation coefficient between the transmitter and receiver pair. In the simulation, we can predict roughly how parameters that affect power levels change with time and use this knowledge to define adaptive time-steps.

Consider that user movement is modeled by a *trajectory* represented by an ordered collection of segments of straight line. Let the speed in each segment be constant. We assume that, at any point in the simulation, we can compute the trajectory of any user up to an arbitrary simulation time. Say we do this for a time interval $[t_a, t_b]$: we can compute upper and lower bounds for all attenuation coefficients a_{ii} and a_{ij}, respectively, during this time span.

Now, if we can determine that during $[t_a, t_b]$ no discrete-events that may cause some transmitter to raise its power to the maximum value will be scheduled, we should be able *to advance simulation time directly from t_a to t_b*. Of course, this is only practical if we can bring the model to the same state it would reach by time-stepping the whole interval. We explain ahead the conditions that must be met to make this possible.

To discover the point in time to which we can fast-forward the simulation, we identify the earliest span of time in which no call arrivals and no hand-offs occur; these events are of particular interest because they increase the number of users on channels raising interference noise. The power control algorithm responds to higher noise by raising power levels so that the target SNR may be achieved. As we have seen, if power levels stay at maximum for a length of time, calls may be blocked or dropped to bring interference down.

A "jumpable interval" is defined by setting t_a to the current simulation time and t_b as the time of the earliest known call arrival or hand-off. Once such an interval is found, we study the evolution of attenuation coefficients in this time frame and determine if circumstances may force some transmitter to maximum power. We need to look a bit closer at the problem to develop a theoretical foundation for this idea. We start by stating a few assumptions:

Assumption 1: *A power control law is a fixed-point computation which converges to a power assignment that satisfies a well-defined goal, independently from the initial state of the system.*

Assumption 2: *A power control law strives to maintain the instantaneous SNR, $\rho_i(k)$, for every receiver i, equal to a pre-defined target ρ. To do so, it computes the power levels at time k based on the state of the system (values of interference and attenuation coefficients) at time $k-1$.*

Let us define $a_{ij}(k)$, the attenuation coefficient between transmitter j and receiver i at time k as:

$$\begin{aligned} a_{ij}(k) &= \tfrac{1}{d_{ij}^{\alpha}(k)} \\ d_{ij}(k) &= \text{dist. from transmitter } j \text{ to receiver } i \end{aligned}$$

Also, if c' and c'' are the two channels adjacent to channel c where a receiver i operates, we compute the cross-talk over i at time k as:

$$x_i^c(k) \;=\; \sum_{j\neq i} f\, p_j^{c'}(k)\, a_{ij}(k) \;+\; \sum_{j\neq i} f\, p_j^{c''}(k)\, a_{ij}(k)$$

where $0 < f < 1$ represents the fraction of transmitted power which extends into the portion of spectrum used by an adjacent channel.

We can now write a generic power law (in discrete time) for channel c:

$$\rho_i^c(k) \;=\; \frac{p_i^c(k-1)\, a_{ii}(k-1)}{v_i^c(k-1)}, \quad \text{where:} \tag{3}$$

$$\begin{aligned} \nu &= \text{receiver's thermal noise} \\ v_i^c(k) &= \nu + x_i^c(k) + \textstyle\sum_{j\neq i} p_j(k)\, a_{ij}(k) \end{aligned}$$

Equation 3 can be rewritten as:

$$p_i^c(k+1) \;=\; d_{ii}^{\alpha}(k)\, \rho_i(k)\, v_i^c(k) \tag{4}$$

Next, consider an artificially constructed system S^*, such that the attenuation coefficients in the computation of power for every transmitter is the worst case observable in interval $[t_a, t_b]$. This is equivalent to saying that in S^*, $d_{ii}(k)\mid_a^b = \overline{d_{ii}}$, the largest distance between a transmitter-receiver pair i, and $d_{ij}(k)\mid_a^b = \underline{d_{ij}}$, the shortest distance between transmitter j and receiver i during interval $[t_a, t_b]$.

We can then state the theorem that is the foundation of our adaptive time-step technique, valid for any power control algorithm that satisfies assumptions 1 and 2.

Theorem 1: *If the power level of some transmitter rises to the maximum in the full time-stepped simulation of interval $[t_a, t_b]$, then the same will happen when we compute power levels for S^* constructed as above.*

Proof of Theorem 1: (by contradiction) Suppose there is a time $s \in [t_a, t_b]$ in the time-stepped simulation when the power level of a base transmitter i is maximal and this is not so in S^*. The positioning of mobiles at time s is such that all mobiles are at least as far away from transmitters as they are in S^*; we observe that $v_i(k)$ is a *decreasing* function of each $d_{ij}(k)$, and an *increasing* function of each $p_j(k)\, x_c$. S^* is constructed to minimize each $d_{ij}(\cdot)$ and to maximize each $d_{ii}(\cdot)$, so the noise value computed in S^* is larger than in the real system. Since the noise value in S^* is larger, the power to which the fixed point algorithm converges must be larger than that in the real system, establishing the contradiction. (Note we present here only the downlink case understanding that by symmetry the same reasoning can be applied to uplinks.) $\square$

From theorem 1, we define a predicate *probe*(t_a, t_b) which evaluates to *true* if no transmitter may achieve maximum power in $[t_a, t_b]$ and to *false* otherwise. We can then describe *interval jumping* as follows:

while (!*simulation_done*) **do**
  compute interval $[t_a, t_b]$
  if $t_b - t_a > m\Delta$ **and** *probe*(t_a, t_b) **then**
    advance time to t_b
    update mobiles positions to time t_b
    compute noise and power assignment
    apply LADCA algorithm
    run time-stepped simulation for $\Gamma\Delta$ time
  else
    run time-stepped simulation to t_b
  end if
end while

The cost of applying *probe* on an interval is not negligible; in fact, it may be comparable to going through a few time-steps. Many bounds have to be evaluated and the power assignment for S^* has to be computed, thus only intervals with a minimum length should be considered for jumping (assume the **if** clause in the algorithm above is evaluated in short-circuit mode). One must set m appropriately to keep computational costs down.

Assumption 1 gives us an important leverage: all we need to compute the power assignment at t_b is the spatial arrangement of mobiles. However, to ensure the simulation resumes from the correct state after a jump, we iterate the power control law until a fixed point is reached. This produces no advance in simulation time and guarantees that no error is introduced.

Note that after the power assignment has been computed, we must still apply the actions of the LADCA algorithm considering the advance in time. Also, it is important to time-step the model for a minimum of $\Gamma\Delta$ time after the event at t_b has been processed for it may force a change in channel allocation.

To assess the potential of this technique, we ran a typical simulation and recorded the size of jumpable intervals in terms of the number of time-steps contained in each. The simulation model parameters used for this test were: 1000 users with constant speed of 1m/s, per user call interarrivals exponentially distributed at a rate of 3.0 calls/h, calls lengths exponentially distributed with mean of 360s, a network of 10 by 10 hexagonal cells with 1000m of side and time-step size of 0.01s. We observed that under these circumstances the system is properly saturated, that is, the blocking probability registered is greater than 1%.

In this scenario, the aggregate distribution of call interarrival times is exponential. When this fact is put together with the procedure for interval construction explained in the previous section, one sees that the distribution of jumpable interval lengths is also exponential. We confirmed this through exploratory data analysis of the results of our preliminary run and observed a mean interval length of 75 Δ. This number indicates a considerable potential for accelerating simulations of this model both sequentially and in parallel.

5. Sequential Simulation

In this section we present experiments with a sequential simulator built to provide insight into interval jumping's performance and, also, to serve as a basis for speedup computation. Before going into experimental data, we discuss a few implementation details which are also relevant for the parallel simulator.

We compute noise over one particular receiver summing only interactions produced in its *sphere of influence*. The radius of this region is determined *a priori*, based on the worst possible arrangement of transmitters in space and an arbitrary, maximum acceptable error for interference computation [11].

Consider a receiver i positioned inside an arbitrary cell $\mathcal{C}$: the number of transmitters j that can create interference on this receiver is equal to the number of cells in $\mathcal{C}$'s sphere of influence. Note that the distance between each transmitter j outside $\mathcal{C}$ and receiver i inside $\mathcal{C}$ is bounded from below by cell geometries. By the same token, the maximum distance between the transmitter/receiver pair in the cell is bounded from above. We explore these facts to pre-compute all values $\overline{a_{ii}}$ and $\underline{a_{ij}}$ for one generic cluster of cells, which can be remapped to the real space when we compute bounds for a specific receiver in *probe*.

The quality of bounds used to implement *probe* has an impact on the performance of interval jumping. The tighter the bounds, the more accurately the predicate will be able to indicate the threat of a discrete-event. If worst-case bounds are too loose, *probe* may indicate the interval should not be jumped when it should be safe to do so. Note that an efficient method to compute bounds on the fly for each particular interval, makes possible a refinement of our technique. If an entire interval cannot be jumped at once, the simulator might be able to identify safe subintervals to jump over.

The experiments we present next, though, are simulations in which an interval is either entirely jumped over or fully time-stepped, a consequence of using worst-case bounds in *probe*. In this case, what determines whether the predicate returns true or false is the concentration of callers in the same channel around each cell.

Figures 1 and 2 illustrate the performance of the jumping technique as a function of step-size. As could be expected, we see a fast degradation with increasing step-sizes: the coarser resolution makes intervals of time without call arrivals or hand-offs harder to find.

In these experiments, we set $m = 2$ in the interval jumping algorithm. This value seems to be enough to keep the cost of interval jumping balanced against time-stepping. As figure 1 shows, interval jumping's performance does not become worse than time-stepping, even for larger Δ.

These results, although encouraging, do not reflect the gains predicted by our statistics of average interval length. This fact indicates there might be large fixed costs associated with both time-stepped and jumping simulations and requires further investigation. We have observed an increase in the accelerations obtained by redesigning the simulator to make better use of memory and threading mechanisms. It is possible that with further improvements the accelerations will be more reflective of the potential of interval jumping.

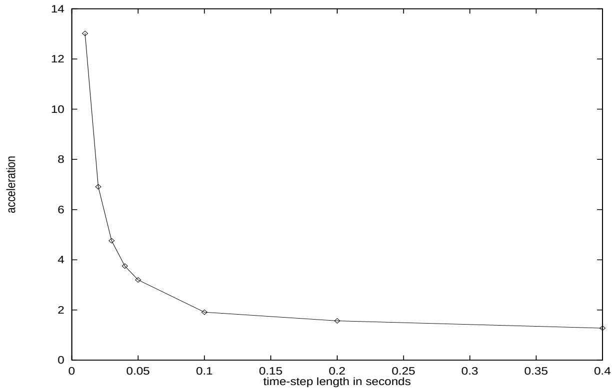

Fig. 1. Acceleration vs. step-size using interval jumping

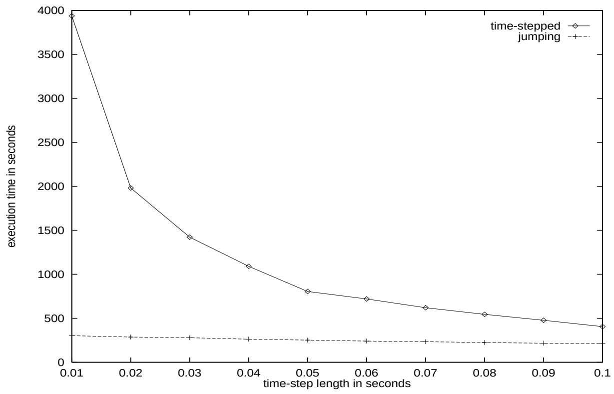

Fig. 2. Execution time vs. step-size for time-stepped and jumping simulations

6. Parallel Simulation

Finally we consider implementation of this algorithm on parallel architectures. As observed in [7], the most natural partitioning scheme for simulations of wireless systems is to break the model up according to channels rather than simulated space. This partitioning and the underlying mathematical model strongly affect how one can address the synchronization problem.

In the serial implementation of interval-jumping, the interval endpoint t_b is the earliest next time in the system where a mobile may join a cell (either by hand-off or by new arrival); suppose the associated channel is c. A probe function is applied at channel c and time t_b, and if the test passes, all channels are brought up to time t_b and the power vector at that instant is computed; otherwise we time-step the entire system up to time t_b. The question now is whether in a parallel simulation these probes and updates can be conducted asynchronously. Suppose that channel c can induce interference in adjacent channels c' and c''. If the probe applied at c and t_b passes, the aggregate power used in channel c rises, thereby raising the noise level in c' and c''. This means the operating state in those channels may change—power levels may have to increase to meet the target SNR and it is possible that the increased noise can be enough to cause this to be impossible, triggering a channel switch, call block or call drop. Even if this does not occur, mathematically the power levels in c' and c'' will rise, thereby inducing greater noise in *other* channels adjacent to them, where the same logic applies. Extending this line of reasoning out transitively, all channels must have their power levels recomputed at time t_b. If we are to synchronize conservatively in a parallel context, we must either use more sophisticated bounds to avoid all these channel updates, use some other trick, or simply do all the updates. Given the present scheme for computing bounds on attenuation coefficients based solely on *numbers* of users in cells and worst-case assumptions about their spatial coordinates, we are forced to keep all processors in tight synchrony if we are to synchronize conservatively. This is, in fact precisely what we do. Our parallel version does what the serial version does, with the parallelism coming solely from the obvious parallelization of the power-vector update step. As illustrated by Figure 3, the processors cooperate to compute the time of least next arrival ($\min\{g, f, d\}$). The probe is applied there, and if successful, all processors engage in the power vector update at that time (time-steps from d to e).

There is not a great deal of lookahead to exploit in this model: we cannot predict the evolution of power levels, which can trigger changes in channel alloca-

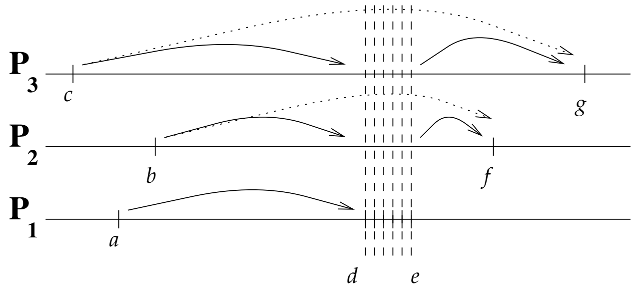

Fig. 3. Synchronizing parallel jumps

tion. With the knowledge of model parameters and the LADCA algorithm, we can squeeze *some* lookahead out of the model:

- A call in probing phase is switched to a different channel as soon as power levels reach maximum. We cannot determine this moment, but knowing the order channels will be visited, we can predict when a call would migrate to each different processor, if it at all.
- A call in service phase is switched to another channel after at least $\Phi\Delta$. Again, knowing the order channels are visited we could exploit this knowledge just as above.

Another line of thinking to pursue is to refine attenuation coefficient bounds dynamically by exchanging bounds on future call arrivals by analyzing future event lists. The current bounds, computed based on the state at time t_a could include the effects of future arrivals in future time-span. This idea was explored in [6], and would allow a processor to not block while another engaged in a probe.

The experiments were present here derived directly from our serial method, which being an adaptive time-stepping scheme, is parallelizable in a natural way. All empirical data presented henceforth was obtained with an SGI Origin2000, a shared-memory parallel machine. The model parameters used are the same as those of the experiment presented in Section 4 except for a larger total of 32 channels, which we divided into pools of the same size across the processors, and a simulation horizon of one hour.

As the algorithm presented in Section 3 will, at times, force the parallel simulation to proceed in lockstep, we decided to run entire simulations in this fashion to have a performance reference. Each processor was allowed to compute the events in one time-step and then forced to pause at a barrier until all others were ready to proceed. Figure 4 gives the execution times as a function of step-size and Figure 5 the corresponding speedups.

Next, we implemented the parallel version of interval jumping, which makes the simulation alternate between time-stepping and and jumping over intervals. Figure 6 and Figure 7 present the performance results obtained with this method.

The speedups obtained so far (relative to sequential interval jumping) are consistent with those of the simple time-stepping method, and like in that case, are not as large as we'd like. For the interval-jumping case, one of the probable causes is that probes are serial bottlenecks—parallelism is exploited only at power vector updates. However this does not explain the disappointing relative speedup of straight time-stepping. We are still investigating the root causes of these performance deficiencies.

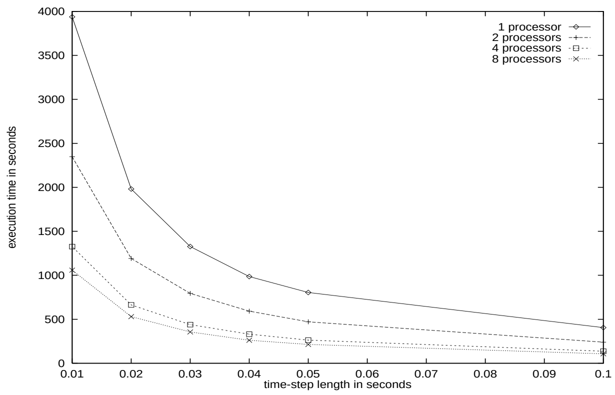

Fig. 4. Execution time vs. step-size with simple time-stepped simulations run in lockstep

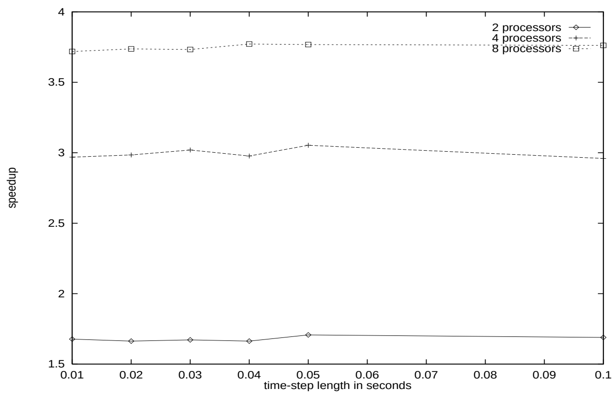

Fig. 5. Speedup vs. step-size with simple time-stepped simulations run in lockstep

More often than not the success of a probe at t_b and admittance of a new call will *not* trigger handoffs in other channels, so our conservatism may be unwarranted. One could approach synchronization optimistically, allowing each processor to asynchronously attempt to interval jump based solely on its assigned channels. Let us consider the implications. A probe calculation at time t_b may need power levels from adjacent channels, possibly assigned to different processors, which may have not yet simulated up to that time. Since we know that the precise power state at t_b depends on those levels, to process optimistically is to

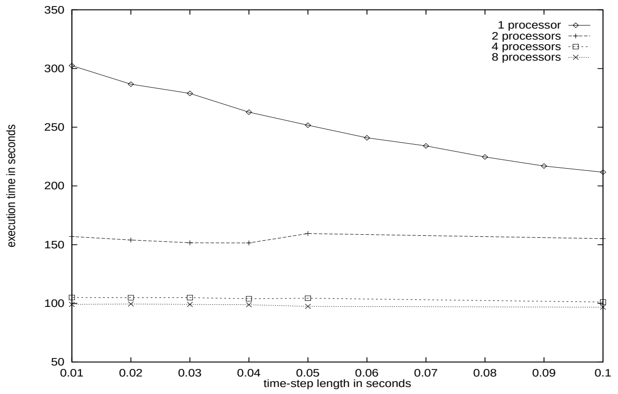

Fig. 6. Execution time vs. step-size with jumping simulations run with next-event synchronization

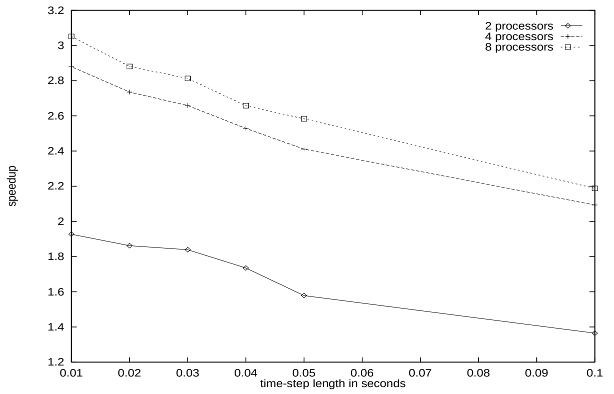

Fig. 7. Speedup vs. step-size with jumping simulations run with next-event synchronization

make some assumptions we hope will be correct and rollback only if they are not. One approach would have a processor responsible for channel c report all changes in usage of c to processors responsible for adjacent channels c' and c''. When a processor needs state information about a channel managed elsewhere, it uses the last known update and assumes it will not have changed. If the assumption turns out to be incorrect, a time-stamped update will arrive in its past to correct the assumption. Such an approach may work if one channel's noise does not depend on very many other channels, but this misses a key opportunity to exploit parallelism further. *Most state changes on one channel do not induce events on other channels.* What is needed is a mechanism so that a processor can apply a probe to channel c at time t_b using out-of-date information on channels c', and c'', but still come to the correct conclusion. This could be accomplished by having c's processor send messages to those of c' and c'' indicating worst-case assumptions being made about their state in c's probe, with the understanding that the recipient will respond, with a NACK and fuller description of state, *only* if its state at that time is "worse" than assumed. Thus a rollback and corrective action occurs not if the noise values used in the probe were incorrect, but only if they were too low.

7. Conclusions and Further Work

We have presented a novel technique for accelerating simulations of wireless systems and conducted preliminary studies towards its parallelization. The empirical results show some promise, but leave ample room for more work. In this paper we have identified some ways of better exploiting parallelism, including through optimistic synchronization, and intend to investigate these further. We also intend to investigate refinements on bounds for interval probing, to extend the length of jumps.

References

[1] C. D. Carothers, R. M. Fujimoto, and Yi-Bing Lin. A case study in simulating PCS networks using time warp. In *Proceedings of the 9th Workshop on Parallel and Distributed Simulation (PADS '95)*, pages 87–94, 1995.

[2] C. D. Carothers, R. M. Fujimoto, and Yi-Bing Lin. Simulating population dependent PCS network models using time warp. In *Proceedings of the 1995 Winter Simulation Conference*, pages 555–562, December 1995.

[3] C. D. Carothers, R. M. Fujimoto, Yi-Bing Lin, and P. England. Distributed simulation of large-scale PCS networks. In *2nd International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '94)*, 1994.

[4] Gerard J. Foschini and Zoran Miljanic. Distributed autonomous wireless channel assignment algorithm with power control. Internal AT&T Bell Laboratories Document.

[5] Gerard J. Foschini and Zoran Miljanic. A simple distributed autonomous power control algorithms and its convergence. *IEEE Transactions on Vehicular Technology*, 40(4):641–646, November 1993.

[6] Albert Greenberg, Boris Lubachevsky, David Nicol, and Paul Wright. Efficient massively parallel simulation of dynamic channel assignment schemes for wireless cellular communications. In *Proceedings of the 8th Workshop on Parallel and Distributed Simulation (PADS '94)*, 1994.

[7] Michael Liljenstam and Rassul Ayani. A model for parallel simulation of mobile telecommunication systems. In *4th International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '96)*, pages 168–173, 1996.

[8] Yi-Bing Lin and Paul A. Fishwick. Asynchronous parallel discrete event simulation. *IEEE Transactions on Systems, Man and Cybernetics*, 26(4):397–412, 1996.

[9] Yi-Bing Lin and Victor W. Mak. Eliminating the boundary effect of a large-scale personal communication service network simulation. *ACM Transactions on Modeling and Computer Simulation*, 4(2):165–190, April 1994.

[10] Pangfeng Liu and Sandeep N. Bhatt. Experiences with parallel N-body simulation. In *6th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA '94)*, pages 122–131, 1994.

[11] Luiz Felipe Perrone. *Accelerating Simulations of Wireless Systems*. PhD thesis, The College of William and Mary, 1998. In preparation.