Power-aware State Dissemination in Mobile Distributed Virtual Environments

Weidong Shi, Kalyan Perumalla, Richard Fujimoto
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
{shiw,kalyan,fujimoto}@cc.gatech.edu

Abstract

In distributed simulations, such as multi-player distributed virtual environments (DVE), power consumption traditionally has not been a major design factor. However, emerging battery-operated mobile computing platforms require revisiting DVE implementation approaches for maximizing power efficiency. In this paper we explore some implications of power considerations in DVE implementation over mobile handhelds connected by wireless networks. We focus on the state dissemination problem in DVEs and propose a new power-aware dead reckoning framework for power-efficient state dissemination. We highlight a fundamental tradeoff between state consistency and power consumption, and present an adaptive dead reckoning algorithm that attempts to dynamically optimize the tradeoff at runtime. We present a quantitative evaluation of our approach using a synthetic DVE benchmark application.

1. Introduction

Traditional distributed interactive simulations such as distributed virtual environment (DVE) applications run on workstations connected by high-speed networks that are not power constrained. However, newly emerging interactive applications execute over battery operated mobile platforms. Examples of these applications include wireless multiplayer gaming and augmented reality systems used for military and civilian applications. One important design goal of mobile platforms is to increase the lifetime of batteries.

An important problem in realizing distributed simulations for DVE applications concerns dissemination of state (ground truth) information in order to ensure different users observe a consistent view of the virtual world. This paper describes an approach to minimizing the energy consumed for state dissemination by the wireless network interface in interactive simulations for DVEs, by exploiting application level information. We investigate the tradeoff between simulation quality and energy savings. Based on an understanding of this underlying tradeoff, we develop a novel power-aware state update scheme that maintains distributed simulation state consistency while at the same time yields lower energy.

Our approach uses a closely coupled feedback loop between the simulation and the device power management service to achieve a balance between power conservation and simulation quality. A distinguishing characteristic of our approach is its adaptive nature – it is designed to dynamically extract maximal suspension time without adversely affecting state consistency. It is adaptive because it is able to automatically discover the appropriate suspension period of the network interface as the simulation proceeds.

The paper is organized as the follows. In section 2, we characterize power consumption of mobile platforms. Section 3 describes our power-efficient state update approach using a new power-aware dead reckoning algorithm. In section 4, we report results of a simulation study analyzing this approach. Section 5 describes related work, followed by conclusions and future work in the final sections.

2. Mobile Platform

We review relevant features of mobile wireless platforms in order to highlight their implications for power-aware state dissemination.

2.1. Hardware

Current mobile computing platforms include devices with different form factors and capabilities. Laptop computers equipped with a built-in or attached wireless network interface card (WNIC) are commonly available. Many of the current Personal Digital Assistant (PDA) devices are also capable of wireless network communication using WNIC attachments. Examples include the Compaq IPAQ handheld running Windows CE or Linux, with a PCMCIA jacket to hold a WNIC. When these devices are mobile, their power is limited by

their battery life.

Wireless networks can be set up to operate in infrastructure or ad-hoc mode. Here, we focus on an infrastructure mode of operation, in which a dedicated node always remains powered on, and acts as a buffering and synchronization point for the rest of the nodes.

2.2. Network Interface Power

Wireless network interface cards consume a significant portion of the total power consumed by a communicating mobile device. This fact is confirmed by several studies conducted in recent years [1, 3, 10, 11] to characterize power consumption of mobile devices via direct measurements. Our own measurement on Compaq IPAQ handheld devices with PCMCIA wireless cards shows that the wireless network interface card can consume up to 50% of the total power.

2.3. Power Modes and Timing

A typical WNIC supports multiple modes of operation, with varying levels of power consumption. The common modes include the following, roughly in descending order of power consumption: Transmit, Receive, Idle, Suspended and Off modes. The Transmit and Receive modes consume comparable amounts of power, especially in ad-hoc mode. The Suspended mode is a low power mode similar to the Off mode, but provides faster resumption than switching on from the Off mode. In ad-hoc mode, the power consumption in Idle mode of a Lucent IEEE 802.11 WaveLAN PC card is nearly as large as that of receive mode [1]. Results of [4] also show that power consumption in an Idle or Transmit/Receive state is an order of magnitude higher than the power consumption in Suspended mode.

For our purposes, these modes can be reduced to two: Active and Suspended. It is clear that maximal power savings are obtained by maximizing the time the WNIC is maintained in the Suspended mode. Our own measurements also show that periodic suspension and resumption of the WNIC does indeed substantially increase overall device battery life.

2.4. Switching Power Modes

Although switching the WNIC to suspended state can save power, the latency for switching to suspended mode, and later back to active mode, can be high. The WNIC is unusable during the time it is switching between modes, and hence introduces additional message latency for any new or undelivered messages. The suspension-resumption latency greatly depends on the specific driver and operating system. We have observed latencies on the order of 600ms on the Compaq IPAQ handheld platform with Lucent Orinoco WNIC. Much lower latencies, around 70ms to 100ms, using proprietary WNIC drivers are reported in the literature. The long latencies are most likely due to inefficient design of the MAC controller chip or the wireless adapter interface. We expect that 50-100 millisecond switching latency will become standard in the near future with more power friendly MAC controller design. Beside the latency issue, switching modes may interfere with the operation of transport level protocols such as TCP. This issue has been identified previously, and modifications to the standard TCP state machine have been proposed by some researchers to work with MAC level power management. Our approach can benefit from those efforts by adapting their solution to our power aware state update scheme.

2.5. Power Consumption vs. Latency

Suspending the WNIC can result in increased message latency and a higher probability of lost packets. Message latency is increased as a result of the extra delay introduced by device suspension at either the sender's or the receiver's side. When a mobile host sends a state update, its communication device may already have been suspended. In order to send the update, either it must wait until the proposed suspension time has expired, or it must resume the device prior to the expiration of the suspension period. Regardless of which approach is used, there will be additional delay to send the message. On the receiver side, its communication device may be in sleep mode when some other host sends a message to it. In this case, the earliest time for it to receive the message is after the communication device resumes its operation.

Naïve power saving approaches that do not properly coordinate application requirements with device suspension will incur the above undesirable effects and may severely degrade the quality of distributed interactive simulations in which network delay has a large bearing on the quality of the simulation [12].

3. Power-Aware State Updates

In distributed interactive applications, each host is responsible for simulating one or more entities whose states are broadcast/multicast to other remote hosts by messages transmitted over a network. One example is a simulated vehicle transmitting its location and speed in a distributed driving simulator. Because of network delay, at any given time, the state copy maintained by a remote host is always behind its true state. This problem is defined as space-time consistency when the state

information is spatial in nature. Since the number of state updates in a distributed simulation can overwhelm the available network bandwidth, techniques have been introduced to reduce the amount of state updates without great loss of space-time consistency. One widely used technique is dead reckoning [13, 14, 15].

In the traditional approach to dead reckoning, an "error threshold" level is first determined based on the maximum state inconsistency tolerable in the simulation model. Whenever the difference between the true local state and the remotely tracked state exceeds the error threshold, a state update is sent over the network to synchronize the local and remote states. In addition, periodic state updates are also sent even if the error threshold has not been exceeded.

Our approach to power-aware state updates is to let the application expose an appropriate slack time to the power management service so that communication devices can be suspended with minimal, bounded impact on space-time consistency.

3.1. Adapting to Dynamic Behavior

Studies of dead reckoning algorithms have revealed that state updates are often highly bursty and unpredictable [7]. Due to the complexity and unpredictability of state updates in most distributed applications, it is nearly impossible to have the suspension time pre-determined and guaranteed to work effectively throughout the entire simulation. Sometimes, entities may move slowly and the interval between state updates is long enough to allow aggressive suspension of the communication devices. However, at some other times, the entities may move quickly and send frequent state updates, allowing little or no time for suspension. An adaptive scheme is required that can dynamically determine the best suspension time as the simulation proceeds.

3.2. Algorithm Rationale

As discussed earlier, the larger the delay that the simulation can tolerate, the greater the power it can save by suspending and resuming the communication device. In order to determine a proper suspension period based on simulated model information, we developed a new state update approach for entity simulations that can dynamically determine a local suspension time under a given consistency constraint. In a simulation involving many hosts, each host may have its own local suspension period, and the actual suspension time used by the power management service may be decided according to the global minimum among all the hosts' suspension periods. Why does a local suspension need to consider the suspension period of other hosts? This is because each host also receives state updates published by other hosts. Assume that a host x simulates a slowly moving entity that has a large tolerance on suspension interval and another host y simulates a more rapidly moving entity with only a low tolerance on suspension interval. If host x only considers its own suspension period determined by its simulation of the slowly moving entity and uses that for its suspend-resume time, its consistency error for the data published by host y will be high because of the extra transmission delay caused by x's suspension. This means that the actual suspension interval must be computed in a global fashion. The actual suspension time also depends to a great extent on the network topology of the simulation environment.

The tolerable latency value determined by a host can be given as a hint to its power management service. As such the local communication device need not be suspended for exactly that amount of time. The actual suspension interval also depends on the hint values provided by other hosts participating in the simulation.

3.3. Algorithm

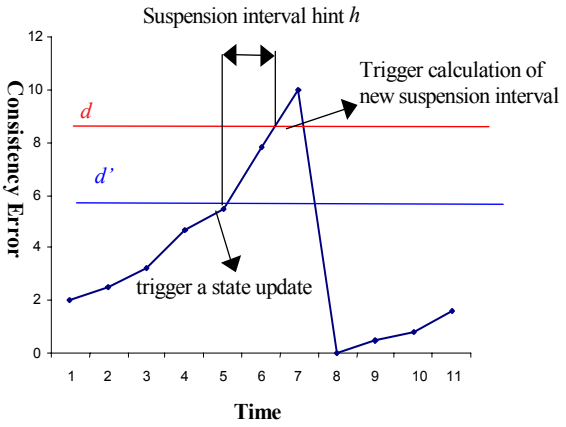

Figure 1. Illustration of power-aware dead reckoning algorithm operation

We now describe our power-aware dead reckoning algorithm executed at each node. Let $P_{true}(t)$ be the true position of an entity at time t, $P_{dr}(t)$ be the entity's dead reckoned position, and d be the dead reckoning threshold. A second, smaller, dead reckoning threshold d' is defined that is used to trigger generation of state updates earlier in order to compensate for the increased latency due to the use of power-aware messaging. In the experiments described later, d' is set to $0.75*d$. The d and d' values are used to compute h, the maximum latency requested by the application. The value of h is viewed as a hint that is provided to the power-aware messaging system. The

variables k and *step* are parameters used to control the rate at which h is modified to adapt to changing latency requirements. The operation of our dead reckoning scheme is depicted in Figure 1.

The state update algorithm described next assumes a reliable message transport mechanism is used, and receivers send an acknowledgement for each message. Relaxation of this assumption is an area of future research.

```
1.  Update P_true(t) and P_dr(t) with usual
    dead reckoning
2.  If there is no outstanding publication of
    state update waiting to be acked  {
3.    If | P_true(t) - P_dr(t) | > d'  {
4.      If local host's communication device
          is powered off, resume it
        Send P_true(t) to other host(s)
5.      Set t_pub = t
6.      Mark that there is an outstanding
            state publication not acked yet
      } else {
7.      h = h + k*step;
8.      If h> MAX_SUSPENTION_TIME
9.        Let h=MAX_SUSPENSION_TIME
      }
    } else {
10.   If | P_true(t) - P_dr(t) | > d and there is
        an un-acked state publication {
11.     Set h = t – t_pub
12.     Send threshold-triggering event
        to all other hosts
      }
    }
13. Post the new h value to the underlying
    messaging mechanism.
```

Figure 2. Power-aware dead reckoning algorithm executed by each host

3.4. Discussion

In each simulation cycle, the host executes the steps specified in Figure 2. For simplicity, assume that each host is responsible for simulating a single entity and publishing its state to the other hosts. Like the traditional dead reckoning algorithm, each host keeps track of both the entity's true position as well as its dead reckoned position. In each simulation cycle, the host first updates the entity's true position and dead reckoned position using a standard dead reckoning algorithm (step 1). Then it calculates the distance between the two positions. If the distance is greater than d' and there is no outstanding state update waiting to be acknowledged, the host sends its own entity's state to all receivers monitoring the vehicle and marks that there is an outstanding state send (steps 4 and 6).

When the application sends a state update, the update is not transmitted immediately if the host's network interface is in sleep mode. If this is the case, the host first resumes its communication device and then sends the state information. Before a state update is acknowledged, the host continues to use the old state for calculating $P_{dr}(t)$. This is different from the traditional dead reckoning scheme where $P_{dr}(t)$ is modified immediately using the new published state. Since the host still uses the old state for calculating $P_{dr}(t)$, the disparity between $P_{true}(t)$ and $P_{dr}(t)$ will continue to grow until the state publication is acknowledged. If the difference between the true position and the dead reckoned position exceeds the true dead reckoning threshold (d), and the state publication is still not acknowledged, a new local suspension interval hint (h) is calculated (step 11). At step 12, the application posts the new value of h to the underlying messaging layer that performs the actual message transmission and power management. The interval between state publication and its acknowledgment depends on the transmission delay and receiver suspension delay. Since transmission delay is much less than the suspension period, the dominating factor is the receiver suspension period.

In our scheme, the suspension interval hint (h) is increased linearly when there is no state publication. The rate of its growth depends on the value of the parameter k. Step 8 is necessary for recovery of the suspension hint. One approach to setting k is to assign a constant value. Next, we will discuss a way to dynamically determine k, yielding a better trade-off between power saving and accuracy.

During the simulation, each host receives state updates from other hosts. In doing so, each host also tracks how many state updates are received "late" due to suspension of its local communication device. Receipt of a state publication is considered as late if it is received after the dead reckoning threshold (d) is exceeded. Because each host receives both state updates and their associated threshold-triggering events, it is able to determine whether a state publication is received late based on the receipt time and the time of threshold triggering event. Our dynamic method for determining k requires that each host record the interval between the last two late receipts of state publications. If the interval is smaller than a threshold (say, one second), the value of k will be decreased; otherwise it will be increased. The value of k may also increase periodically if there is no late receipt of state update messages. Adaptively changing k allows a more aggressive power reduction mechanism without greatly degrading consistency.

4. Performance Study

We explore the performance of our power aware dead reckoning and messaging scheme first with a set of simulation studies using application traces.

4.1. Experimental Setup

A simulation of the power-aware distributed simulation system was developed to evaluate its effectiveness. We used a simple interactive application with traced user inputs to model the distributed simulation. Our model simulates a set of hosts, each responsible for simulating one vehicle entity controlled by a user. Each host publishes the state of its simulated entity to all the other hosts using first order dead reckoning. The user controls the vehicle through keystrokes. The user-changeable attributes are speed and direction. By pressing a key, a user can speed up the vehicle. Releasing the same key stops the acceleration and the vehicle maintains its current speed. The magnitude of speed change depends on the interval between key press and key release. Speed reduction works the same way, but using a different key. For changing direction, when a user presses a special key, the vehicle starts to spin either clockwise (or counter-clockwise depending on the key pressed). When the key is released, the vehicle stops spinning and maintains its current direction. The simulation is capable of running with either synthetic input traces or traces from real user input. Each entry in the trace specifies which key is pressed or released and at what time, with millisecond precision.

The underlying network topology and power aware messaging protocol are also modeled. A simple model is used for the MAC level data transmissions. Link throughput was set to 11Mbps and latency is set to 500us, corresponding to the characteristics of wireless cards based on the IEEE 802.11b standard.

We conducted several simulations using different consistency requirements, different human input rates, different minimal suspension time threshold, and different entity movement speeds. The performance metric of interest is the accumulated suspension period for each host. The accumulated suspension period is a useful metric because it is platform-independent, making it possible to interpret the results in a platform-neutral fashion. It represents the amount of time a host maintain its network interface in sleep mode. The suspension time can be converted into power savings using a power model constructed from real measurement. In the simulation, we used the power model defined in [1]. The ratio of power consumption for transmit, receive, and idle modes is 1.5:1.2:1.0. The model is based on actual power measurements on Lucent wireless LAN cards.

First, we used a set of different dead reckoning thresholds (*i.e.*, different values for *d*) and evaluated how power conservation is affected by the consistency requirement. The smaller the dead reckoning threshold, the tighter the consistency requirement becomes. For the smallest suspension time, we use 50ms, and an average user input rate of 1 key per second. The speed of each entity is kept constant at 50 pixels per second. The entity state is updated every 25ms, and one entity is simulated per host. The simulation stops when 1,000 seconds of simulated time has elapsed. This is equivalent to generating 40,000 state updates per entity. We collected the accumulated suspension time for each host, and normalized it against the baseline case where the communication device is always powered-on. The results also include average simulation error under power conservation and the baseline situation.

4.2. Results

Overall, as expected, tighter consistency requirements result in lower power savings, highlighting the tradeoff between power conservation and consistency. For the tested scenario, we observe maximal power savings of 69% when the dead reckoning threshold is 16 pixels and a minimal (31%) power savings when the dead reckoning threshold is 2.0. Our scheme is comparable with a traditional dead reckoning scheme in terms of consistency. Since in our scheme, each host sends its entity state at an earlier time than the traditional dead reckoning method, in some cases, the observed consistency error can actually be smaller than the error using traditional dead reckoning.

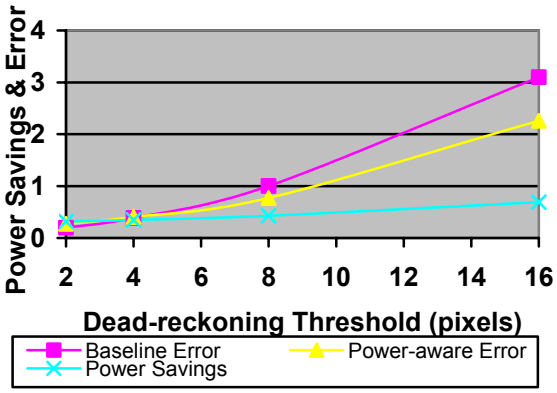

Figure 3. Power savings and consistency error with different dead reckoning thresholds (processors=3, resume latency=50ms).

Results in figure 3 suggest that when the dead reckoning threshold is small (2 or 4 pixels), the consistency error of our scheme is close to the traditional

dead reckoning approach. When it is relatively large (8 or 16), the consistency error of our scheme is more likely to be smaller than the traditional dead reckoning approach. This is achieved by using more published entity states.

Throughput

More published entity states imply a larger bandwidth requirement. Also, our scheme requires each host to send a suspension request to the station before it can actually power off its communication device. This adds to the bandwidth overhead of our scheme. To investigate the overhead, we compared the required transmit throughput per host with that of traditional dead reckoning.

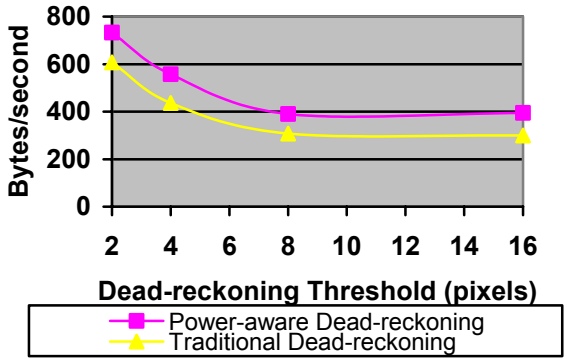

Figure 4. Throughput overhead of our power-aware scheme vs. traditional dead reckoning (processors=3, resume latency=50ms).

As shown in figure 4, the absolute amount of throughput overhead of our scheme is small considering the amount of power saving that can be achieved. Also power conservation only occurs when the traffic for state updates is low. When the throughput requirement of state exchange is close to the maximal throughput the data link can sustain, there will be no opportunity to suspend the communication device. The essence of our scheme is to discover the right suspension time and interval when such an opportunity exists without significantly degrading consistency.

User Input Rate

We next studied the effect of human key input rate on power savings. A real world distributed interactive simulation may have a large variance in terms of human input rate. The input rate may be low when the user is thinking, but may become high at other times. To examine the effects of input rate on power conservation, we observed the effect of two input rates: 1 and 4 keys/second. The 4 keys/second represents an extreme situation because in a real interactive simulation, human users seldom give input at such a high rate. The results are shown in figure 5.

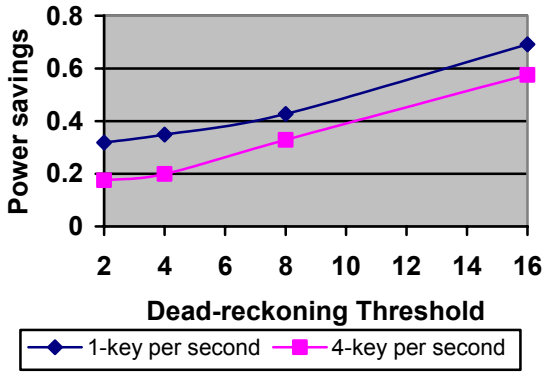

Figure 5. Power savings with varying key input rate (processors=3, resume latency=50ms).

The results show that the input rate does have an impact on power conservation. The amount of power savings decreases when the user input rate is increased. The effect is more significant when the dead reckoning threshold is low. However, even at 4 keystrokes per second and a tight consistency requirement (2 pixels), we can still achieve 17% power savings without greatly increasing consistency error.

Fast Mobility

Another important factor is the speed of the simulated entity. Under the same consistency requirement, a fast moving entity may have less accumulated suspension time than a slow moving entity. In the next simulation, we changed entity speed from 50 pixels per second to 500 pixels per second. This means about 12 pixels per state update. Figure 6 shows the results.

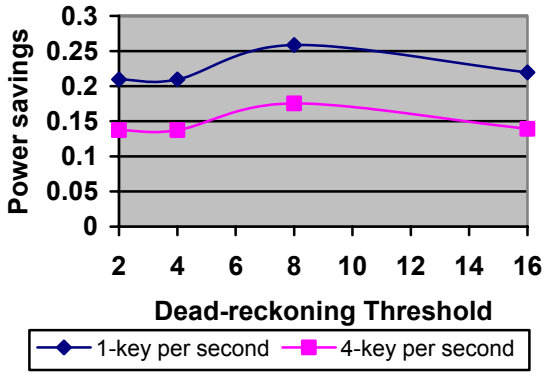

Figure 6. Power savings with *fast entity mobility*. (processors=3, resume latency=50ms).

As expected, higher speeds lead to less power saving. When both speed and input rate are high, power savings decline to as low as 13%. Note that higher speed does not always result in larger consistency errors. The rate of state updates under 500 pixels per second is much higher than the rate of state updates fewer than 50 pixels per

second. Under a dead reckoning threshold of 16, the rate of state updates at a speed of 500 pixels per second is almost twice the rate of updates at a speed of 50 pixels per second. This explains why in some cases, the actual consistency error becomes smaller when the speed is higher.

Suspension Time

In all the preceding studies, we kept the suspension time threshold at 50 milliseconds: all suspension attempts with a suspension period below 50 milliseconds are ignored. The minimum suspension time threshold is important because it changes for different mobile platforms, varying from tens of millisecond to hundreds of millisecond. We evaluated two additional suspension time thresholds, 100 milliseconds and 200 milliseconds. The results are shown in figures 7 and 8.

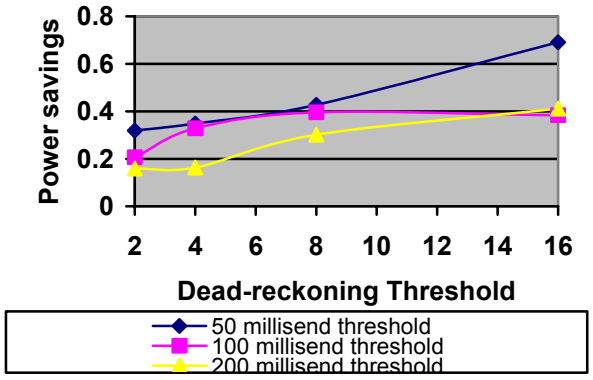

Figure 7. Power reduction vs. suspension threshold (processors=3).

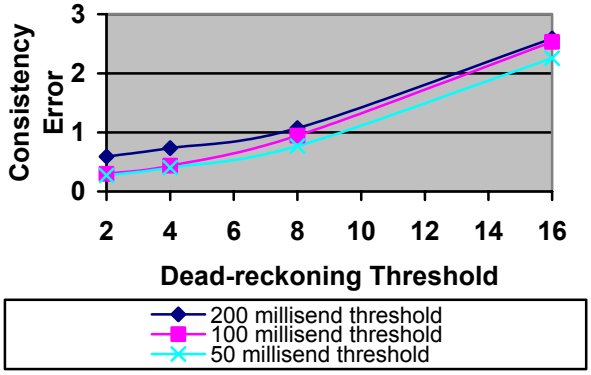

Figure 8. Average consistency error vs. suspension threshold (processors=3).

As shown in the figures, suspension threshold has an influence on both consistency and power savings. The result shows that smaller suspension thresholds out-perform larger thresholds in terms of both consistency and power conservation in our scheme.

Scalability with Number of Hosts

The last experiment explores scalability. We varied the number of simulating hosts from 3 to 8 (input rate 1 keystroke per second, entity speed 50 pixels per second). The results shown in figures 9 and 10 indicate that as the number of simulating nodes increases, the amount of power saving decreases. About 9% power saving is obtained even in the case of eight simulating nodes and a tight dead reckoning threshold. This shows that our scheme is able to scale to a reasonable number of platforms. Another interesting observation is that the amount of consistency error remains stable under different numbers of simulated nodes showing that our scheme maintains adequate quality in the simulation.

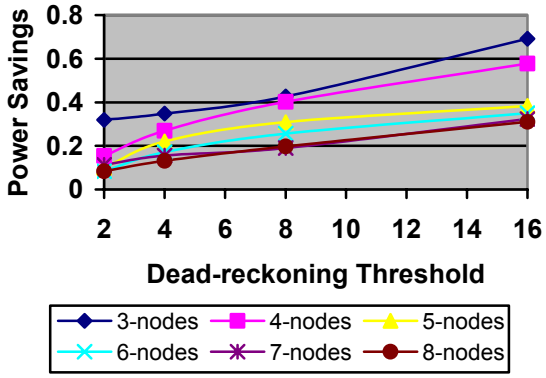

Figure 9. Power savings with varying number of hosts (resume latency=50ms).

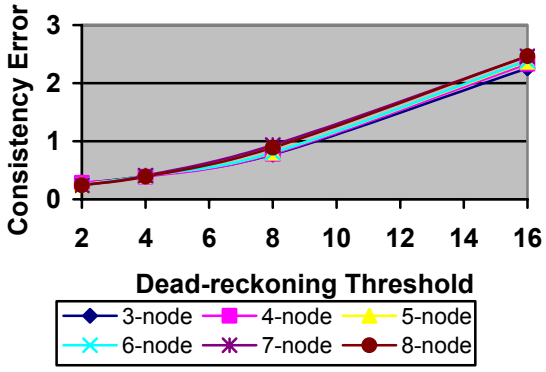

Figure 10. Average consistency error with varying number of hosts (resume latency=50ms).

5. Related Work

In recent years, several researchers have proposed suspension of the wireless devices during idle periods of communication to reduce power consumption [2, 8, 9]. In terms of latency, a few power-aware routing schemes achieve energy reduction without significant impact on

latency, while others may increase the latency. In most schemes, the latency can be bounded through analysis [5]. A few studies have tried to use application assistance to decide when and how long the network interface can be suspended. The tradeoff between reducing power consumption and reducing delay for data is investigated in [6]. An alternative approach that does not require application involvement is to set a timer with a time-out value. When the timer expires and no network activity has been detected since the last time-out, the power management service may conclude that there is no need for communication and hence force shut down the network interface. However, without application information, it is hard for the power management service to decide how long the communication should be suspended. For simple applications such as web browsing and email, an estimate of suspension period may be determined based on application traces. The uniqueness of our study is that we focused exclusively on distributed interactive simulations in which the communication pattern is much less predicable than in previously studied applications.

6. Future Work

Our study in this paper represents an initial step towards the goal of power-aware runtime infrastructure that can be deployed on mobile simulation platforms. Clearly, there are many open issues left for further investigation. Among them is to observe how the scheme performs with an actual implementation over currently available hardware and compare the results with simulation. Furthermore, our approach requires base-station that runs with dedicated power supply. How to adapt the approach to work in an ad-hoc environment represents a challenge and remains a direction of future research.

7. Conclusions

In this paper, we proposed a new power conservation scheme for distributed simulations running on mobile devices. The scheme achieves power savings via opportunistic suspension/resumption of the communication devices. It is designed to meet the consistency requirement by using a new state update protocol that can heuristically expose estimates of safe suspension times to the power management service for intelligent suspension/resume. Results from simulation study using both actual and synthetic input traces show that our scheme is able to optimize the trade-off between state consistency and power savings.

More generally, power considerations constitute a new exciting dimension that needs to be considered in parallel and distributed simulation research, especially due to the recent proliferation of mobile platform and new mobile simulation applications. The power dimension not only bears practical relevance but also raises new technical challenges. We believe the work presented in this paper only scratches the surface with respect to potential open research.

8. References

[1]. L.Feeney and M.Nilsson. investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In Proc. IEEE INFOCOM, anchorage, AK, April 2001.

[2]. R.kravets and P.Krishnan. Application-driven power management for mobile communication. ACM Wireless Networks, 6(4): 263-277, 2000.

[3]. T.simunic, l.Benini, P.W. Glynn, and G.D. Micheli. Dynamic power management for portable systems. In Proc. ACM MOBICOM, pages 11-19, Boston, MA, 2000.

[4]. Michele Zorzi and R.R. Rao, Energy Management in wireless Communications., Proc.6th WINLAB Workshop on Third Generation Wireless Information Networks, March, 1997.

[5]. Adaptive Energy-Conserving routing for multi-hop Ad hoc Networks.

[6]. S. Singh, C.S. Raghavendra. Power efficient MAC protocol for multi-hop radio networks.

[7]. C. Durbach, J-M. Fourneau. Performance evaluation of a dead reckoning machanism. Proceedings of the Second International Workshop on Distributed Interactive Simulation and Real-Time Applications.

[8]. S. Singh, and C.S. Raghavendra. PAMAS: Power aware multi-access protocol with signaling for ad hoc networks. Computer Communication Review 28(3), 5-26.

[9]. S. Singh, M. Woo, and C.S. Raghavendra. Power-aware routing in mobile ad hoc networks. In Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking, pages 181-190, October, 1998.

[10]. J.-P. Ebert, B, Burns, and A. Wolisz. A trace-based approach for determining the egergy consumption of a WLAN network interface. In Proceedings of European Wireless, pp. 230-236, Feb, 2002.

[11]. R. Kravets and P. Krishnan. Power management techniques for mobile communication. in Porc. of 4th Annual international Conference on Mobile Computing and Networking (MOBICOM 98), 1998.

[12]. L. Gautier and C. Diot. Mimaze, a Mulltiuser Game on the internet. in Proceeding of IEEE Multimedia systems Conference, June 1998.

[13]. Saunders, R., Formal Expression of Dead Reckoning: Mathematical and Representation Recommendation, DIS Workshop, Sept. 1991.

[14]. Lin, K. C., Dead Reckoning in Distributed Interactive Simulation, DIS Workshop, June 1992.

[15]. Towers, J., and Hines, J., Equations of Motion of the DIS 2.0.3 Dead Reckoning Algorithms, DIS Workshop, Feb. 1994.