Energy Conservation in Ad Hoc Multimedia Networks using Traffic-Shaping Mechanisms

Surendar Chandra

University of Notre Dame, Notre Dame, IN 46556, USA
surendar@cse.nd.edu

ABSTRACT

In this work, we explore network traffic shaping mechanisms that deliver packets at pre-determined intervals; allowing the network interface to transition to a lower power consuming *sleep* state. We focus our efforts on commodity devices, IEEE 802.11b ad hoc mode and popular streaming formats. We argue that factors such as the lack of scheduling clock phase synchronization among the participants and scheduling delays introduced by back ground tasks affect the potential energy savings. Increasing the periodic transmission delays to transmit data infrequently can offset some of these effects at the expense of flooding the wireless channel for longer periods of time; potentially increasing the time to acquire the channel for non-multimedia traffic. Buffering mechanisms built into media browsers can mitigate the effects of these added delays from being mis-interpreted as network congestion. We show that practical implementations of such traffic shaping mechanisms can offer significant energy savings.

Keywords: MANET, Microsoft media, Real, Quicktime, WNIC energy conservation

1. INTRODUCTION

The proliferation of commodity mobile devices that can not only consume popular streaming formats such as Real,[1] Microsoft media[2] and Quicktime[3] but also create and service streams using wireless network technologies enables a number of emerging application scenarios. A necessary feature for mass acceptance of such a streaming multimedia system is acceptable battery life; especially for the ad hoc router nodes.

Even with tremendous advances in power efficient hardware technologies, WNICs that operate at the same range continue to consume a significant amount of power. Typical 2.4GHz IEEE 802.11b Wavelan cards[4] consume 177 mW in *sleep* state and about 1400 mW while in *idle* state as well as receiving data. For comparison, a fully operational HP iPAQ only consumes 929 mW.[5] The typical long duration of multimedia streams makes wireless network interface (WNIC) energy consumption a particularly acute problem. Future trends in battery technologies alone (along with the continual pressure for further device miniaturization) do not promise dramatic improvements that will make this issue disappear. It is important to look at techniques to reduce the energy consumed by the WNIC to forward multimedia data.

In this work, we exploit the power states of the WNIC to reduce the energy consumption of the ad hoc routers. Aggressively transitioning the WNIC to a lower power consuming *sleep* (instead of *idle*) state can be expected to provide significant energy savings. Note that, if a wireless network interface is in *sleep* mode during the start of packet transmission, the entire packet is not received at the client. So it is imperative that the network interface be ready for the packet. Mechanisms that make the data packets arrive at predictable intervals can facilitate such transitions to lower power states.

We argue that factors such as the lack of scheduling clock phase synchronization among the various participants and the scheduling delays from back ground tasks favors infrequent data transmission intervals. Media browsers adapt to such added delays as network congestion. Buffering mechanisms built into the browsers can mitigate some of these effects. Our traffic shaping proxy can indeed offer significant energy savings for the routing nodes.

This paper is organized as follows: Section 2 reviews prior work, while Section 3 describes the system architecture. We present our results in Section 4 with conclusions in Section 5.

2. RELATED WORK

There has been a considerable amount of work on power management for components of a mobile devices. Lorch et al.[6] presented a survey of the various software techniques for energy management. Havinga et al.[7] presented an overview of techniques for energy management of multimedia streams. Agrawal et al.[8] described techniques for processing video data for transmission under low battery power conditions. Sheu et al.[9] presented a bandwidth allocation mechanism and reservation mechanism for multimedia streams in 802.11 ad hoc networks. Rozovsky et al.[10] describe an efficient mechanism to schedule traffic by exchanging random seeds. Tseng et al.[11] explore power management techniques for mobile multi-hop networks. They prove the effectiveness of three power management protocols; dominating-awake-interval, periodically-fully-awake-interval and quorum-based protocols through extensive simulations. Krashinsky et al.[12] describe a mechanism to dynamically adapt the beacon intervals. Li et al.[13] explore the capacity limitations of ad hoc networks. Our work continues with exploring the energy efficiency of ad hoc networks for multimedia traffic using popular streaming formats.

Our work is similar in spirit to the work of Sivalingam et al.[14] and Shenoy et al..[15] Sivalingam's EC-MAC system explores an low-power medium access control mechanism for infrastructure wireless ATM networks. Similarly, Shenoy's work focuses on stream transformations and intelligent transmission in infra-structure networks. Our primary focus in this work is on ad hoc IEEE 802.11 networks.

Feeney et al.[16, 17] obtained detailed measurements of the energy consumption of an IEEE 802.11 wireless network interface operating in an ad hoc networking environment and showed that the energy consumption of an IEEE 802.11 wireless interface has a complex range of behavior and that the energy consumption was not synonymous with bandwidth utilization. They[18] also describe a mechanism to establish the sleep/wake cycles. Our work exploits the behavior of multimedia streams themselves without explicit synchronization of the underlying sleep/wake cycles.

In our earlier work,[19] we explored the energy implications of popular streaming formats (MS media,[2] Real media[1] and Quicktime[3]) for an IEEE 802.11 infra-structure network. We showed that MS media tended to transmit packets at regular intervals. Real stream packets tended to be sent closer to each other, especially at higher bandwidths. Quicktime packets sometimes arrive in quick succession; most likely an application level fragmentation mechanism. In this work, we extend similar techniques to ad hoc scenarios. The lack of a central coordinator node (i.e. access point) that can provide accurate physical layer clock synchronization along with the dynamic nature of ad hoc networks is expected to complicate similar solutions in the ad hoc scenario.

2.1. MAC level power saving modes

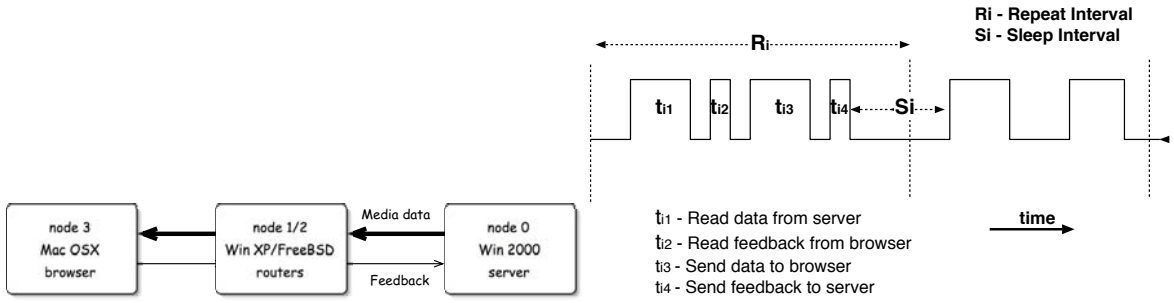
Wireless technologies such as IEEE 802.11b[20] use variations of scheduled rendezvous mechanism for energy saving wherein the wireless nodes switch to a low power *sleep* mode and periodically awaken to receive data from other nodes. The IEEE 802.11 ad hoc mode uses scheduled beacons along with an announcement traffic indication message (ATIM) packet notification mechanism. A station in power saving (PS) mode periodically wakes and listens for ATIM announcements for pending packets. Any ATIM notifications are acknowledged and the PS station waits for the entire beacon interval in higher energy *idle* state. This mode of operation can be expected to provide energy savings for "rare" data. As noted in,[21] this notify→ack→wait model of operation is not suited for pseudo regular multimedia streams; it is likely that there will be at least one pending datagram destined for a PS client during every beacon interval. The ATIM for nodes participating in the multimedia traffic will always indicate active traffic, forcing the clients to stay in high energy consuming *idle* states. A contract wherein the participants expect data at certain time intervals without explicit notifications are preferable. Note that the device drivers for typical cards did not allow us to initiate these ad hoc PS modes to gain practical experience.

3. SYSTEM ARCHITECTURE

In this section, we briefly describe our objectives, the system architecture used and the experimental setup. We will highlight our experiences in the next section.

3.1. Architecture

Our system consists of a multimedia server and a browser connected via a number of ad hoc routers (Figure 1(a)). We assume that a specific route was already identified using route discovery algorithms described in literature.[22] During a typical usage session, the browser node receives multimedia traffic from the multimedia server and sends feedback to the media server. The router nodes forward the traffic in either direction. The system utilizes a scheduled rendezvous



(a) Experiment setup

(b) Traffic shaping mechanism

Figure 1. System Architecture

mechanism to access the wireless channel. The participants of the multimedia transfer negotiate time intervals for sending (and hence receiving) data such that receiving nodes can expect data at pre-determined intervals and transition to a lower power consuming *sleep* states during periods of inactivity. The periodic nature of multimedia streams is expected to assist in such policies. The actual system might employ client and server side proxies which interact with the lower level device drivers to effect these power state transitions. Since no data transfers are expected during the *sleep* interval, no data is lost.

3.2. Scheduled Rendezvous Mechanism

We assume a single priority multimedia traffic that does not contend with other generic traffic. All the participants choose a particular stream bandwidth based on the resource constraints such as available bandwidth, battery capacity left etc. The schematic for one such traffic shaping scheme for the router nodes are illustrated in Figure 1(b). Within a single cycle, the router reads multimedia data from the upstream server (t_{i1}), reads feedback stream from the browser to the server (t_{i2}) and then forwards the multimedia stream to the client (t_{i3}) as well as forward the feedback stream to the server (t_{i4}). Each cycle can be defined by a repeat interval (R_i) or sleep interval (S_i). These intervals can vary with each cycle. These sleep intervals limit the available throughput of the system. The *server* end knows the exact amount of data to be transferred in a given interval R_i and hence can provide better guidance to the *client* end. In general, the amount of multimedia data transferred is much higher than the feedback stream. Hence, t_{i2} and t_{i4} can be 0 for certain R_i/S_i intervals. The specific choice for R_i/S_i and $t_{i(1...4)}$ can affect the potential energy savings. This paper is primarily concerned with choosing practical values for these intervals.

3.3. Objectives

Our primary objective was to explore the practical implications for energy conservation using general purpose multimedia devices and popular streaming formats. We believe that these widely popular formats are more likely to be deployed than custom streaming formats (that are specially optimized for lower energy consumption). Our primary goal was to consume energy proportional to the stream quality and not to reduce the overall energy consumption by always choosing a lower quality stream. The users choose a particular transcoded stream based on the available battery capacity and projected future usage requirements of all participating nodes. Our experiments were designed to answer the following questions:

- Can we realize energy savings for servicing multimedia streams in an ad hoc scenario? We focus our attention on intermediate routers that forward traffic on behalf of other partners.
- What are the practical implications of choosing various delay intervals for traffic shaping mechanisms? We analyze the effectiveness of different approaches in regulating the streams to transmit data packets at regular, predictable intervals.

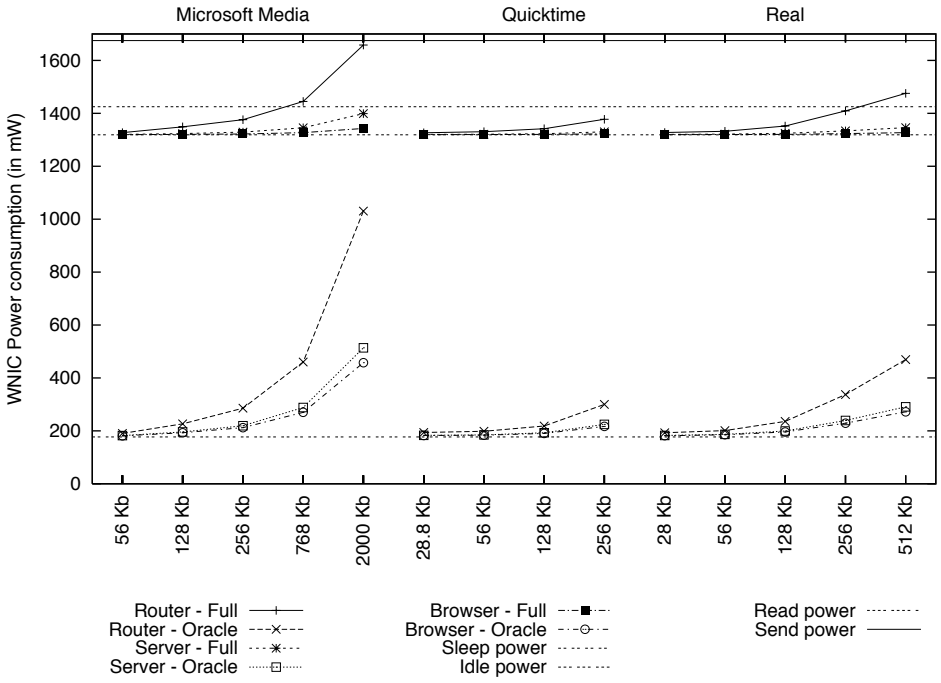


Figure 2. Average power consumed for the various streams

3.3.1. Experiment Setup

For our experiments, we utilized 2GHz Pentium laptops with 512 MB memory and 802.11b ad hoc wireless NICs (on dedicated channels) running Windows XP/FreeBSD 5.1 for the routers (nodes 1 and 2) and Powerbook G4 laptop running Mac OSX for the browsing station (node 0). A Windows 2000 Server (node 3) running Windows Media service, Realserver 8.01 and Apple Darwin Server 4.0 served our 1:59 minutes long Wall (movie) theatrical trailer (also used in[19]). The Wall trailer was digitized to a high quality stream and hence allowed us the flexibility of creating streams of varying formats and fidelities. We used tcpdump to capture packet traces at the browser as well as at the router and server hosts. These packet traces were fed to our energy simulator to analyze the system energy performance. For our simulations, we assumed a useful throughput of 6 Mbps for the wireless channel; utilizing published power parameters[4, 7] for a 2.4GHz IEEE 802.11b card. However, the model does not simulate lower level energy costs such as unsuccessful attempts to acquire the channel (media contention), or messages lost due to collision, bit error or loss of wireless connectivity.

4. RESULTS

First, we analyze the WNIC power consumed for the various multimedia streams by the browser, router and server nodes. We also analyze the maximum power savings possible with an *Oracle* policy that can accurately predict the packet arrival intervals. In the following sections we explore the potential loss in power savings from these ideal values because of inaccuracies introduced by unsynchronized clocks and scheduling delays between the various nodes. Based on these limitations, we design a traffic shaping mechanism that can offer energy savings on general purpose ad hoc routers.

4.1. Potential Energy Savings

For our experiments, we browsed the reference multimedia streams of varying network bandwidth requirements through one intervening router nodes and plotted the average power consumed for the various streams in Figure 2. We also plot the energy required for an *Oracle* policy that transitioned the WNIC card to a lower power consuming *sleep* state during any idle intervals. Note that this policy requires perfect future knowledge of packet arrival intervals. From Figure 2, we note that on average, WNIC cards operating without any power saving modes consumes power similar to the *idle* power; about 1300 mW. For high bandwidth streams such as a MS media stream at 2 Mbps, the power consumption of the nodes are higher as more data is sent/received. As expected, the power consumption of the router nodes are higher as the packets are

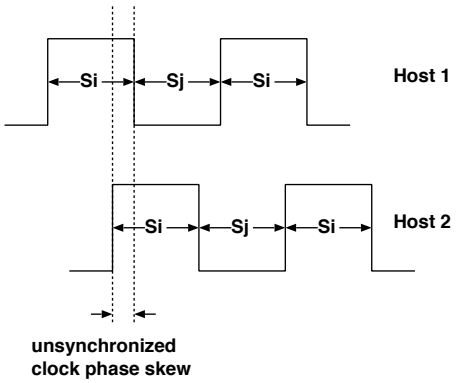


Figure 3. Implications of unsynchronized scheduling clocks

processed twice; received first and then forwarded. We note that the *Oracle* policy indeed shows tremendous opportunity for power savings by consuming power closer to the *sleep* power; about 200–300 mW. There is potential for significant power savings. We noticed similar results with two routers that were spaced such that the transmissions from the server cannot usually be heard by the browser (note that placing nodes any closer would be unrealistic as packets could be directly transmitted without intermediate routers).

4.2. Practical considerations for t_i, R_i/S_i

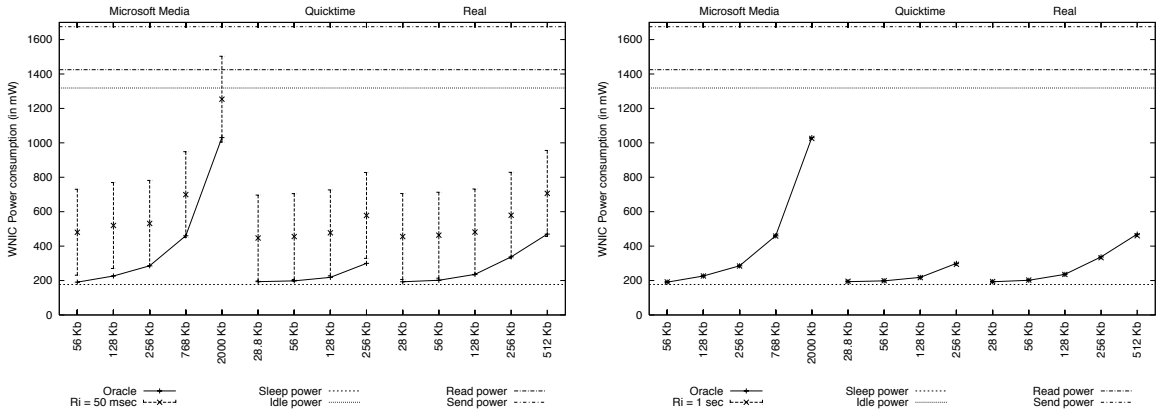
In the last section, we illustrated the possibility for significant power savings if the participants can send and receive data at precisely timed and synchronized intervals. Next, we explore the practical limitations imposed by general purpose devices. We investigate whether packets that were shaped and transmitted can be received at the commodity receiver at the predetermined intervals. Rendezvous mechanisms depend on this ability to transmit and receive packets at well timed intervals. Without this ability, shaping the traffic at the sender has no impact on energy savings because the client cannot be expected to leverage this shaping. In the following sections, we answer the following question:

- Can the receiver wake up to receive data the scheduled intervals?
- Once awake, can the receiver receive the packets at the shaped intervals?
- How robust is the transmission intervals for the streams?

4.2.1. Effects of out of phase scheduling clocks

Traffic scheduling mechanisms require globally synchronized clocks across the various participants. Wireless technologies such as IEEE 802.11b does not enforce synchronized clocks for ad hoc mode of operation. Even though it is feasible to use mechanisms such as NTP to synchronize the clocks across a cluster of ad hoc nodes, the interval timers used in each node to schedule system events can be out of phase. As nodes move and networks get partitioned, it is harder to maintain synchronous phase among various participating nodes (that may coexist in different ad hoc clusters).

General purpose operating systems used in our commodity devices achieve the required periodicity using high resolution interval timers. The timer intervals are usually rounded up to the next system clock tick (typically 10 msec[23]). One can reduce the phase skew by reducing the scheduling interval (say 1 msec) and use mechanisms outlined by Goel et al.[24] to support time sensitive applications. However, reducing the scheduling interval increases the total scheduler overhead. Hence, it is reasonable to assume that general purpose operating systems on these mobile devices can trigger future events at a resolution of 10 msec (rounded up). Thus, even in an ideal system where the schedules are not affected by issues such as clock drift and CPU contention with other runnable processes (i.e. both the hosts wake up exactly every S_j interval), the lack of clock phase synchronization means that their actual wakeup interval can differ by up to 10 msec. This scenario between two hosts Host1 and Host2 is illustrated in Figure 3. Both the hosts have pre-negotiated a periodic wakeup interval of S_i every S_j time units. In this system, the sending nodes can wake up from their sleep and immediately send data at the scheduled time interval. However, the receiving nodes have to account for the unsynchronized clocks and wake up at least



(a) R_i = 50 msec (b) R_i = 1000 msec

Figure 4. Increased power savings spread because of unsynchronized clocks (clock phase skews of 0 through 10 msec)

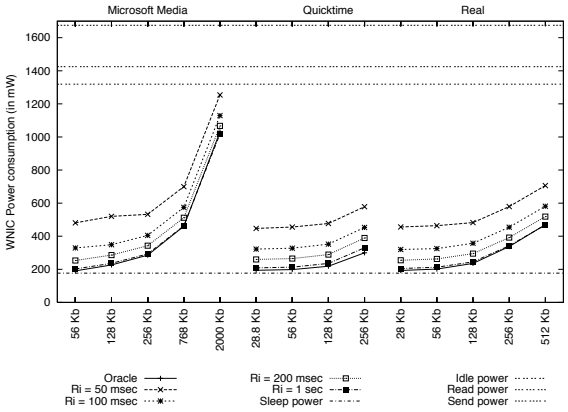


Figure 5. Reduced energy savings for various R_i because of unsynchronized clocks (clock phase skew = 5 msec)

10 msec before the scheduled interval so as not to miss the data. These extra delays are not desirable because they reduce the useful throughput of the wireless medium. The overall penalty depends on the frequency of scheduled wakes. CPU contention with other runnable processes can further exacerbate this clock phase skew overhead. In the next sections, we further analyze the implications of this 10 msec interval for receiving the multimedia streams at scheduled intervals.

Unsynchronized clocks and R_i/S_i interval The repeat interval R_i/S_i affects the interactivity of the multimedia streams. As the stream is forwarded through many routers, the stream may not start and be accessible for a long period. Reducing the repeat interval R_i offers several benefits: 1) *avoid client adaptation:* browser clients misinterpret these added delays as network congestion and adapt to a lower stream quality, 2) *less router buffering:* for resource constrained router nodes, this can be especially a bad problem, and 3) *detect node failures quicker:* alternate routers can be identified quicker.

However, as discussed in Section 4.2.1, unsynchronized clocks can add an extra *idle* time of up to 10 msec for the *clients*. This delay manifests itself as wasted power for the *clients*. To illustrate the effects of this added delay, we plot the spread in power savings by varying the clock phase skew from 0 through 10 msec and for various R_i's of 50 and 1000 msec in Figures 4(a) and 4(b), respectively. From Figure 4(a), we notice that for a R_i interval of 50 msec, the spread in power savings introduced by unsynchronized clocks can be quite pronounced; up to an additional 500 mW on average. As we reduce the amount of scheduled intervals, there is less data (needing less time) that needs to be transmitted to the *client*. Hence the effects of the clock phase skew dominate the power consumption. However, as we increase the R_i (to 1 second), we reduce the frequency of transmission intervals leading to minimal variance. Next, we plot the practical power savings

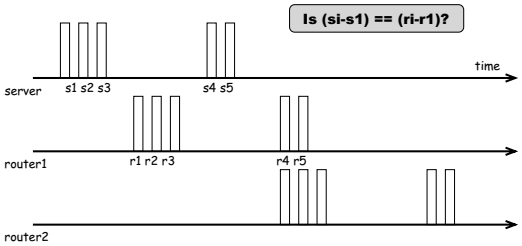
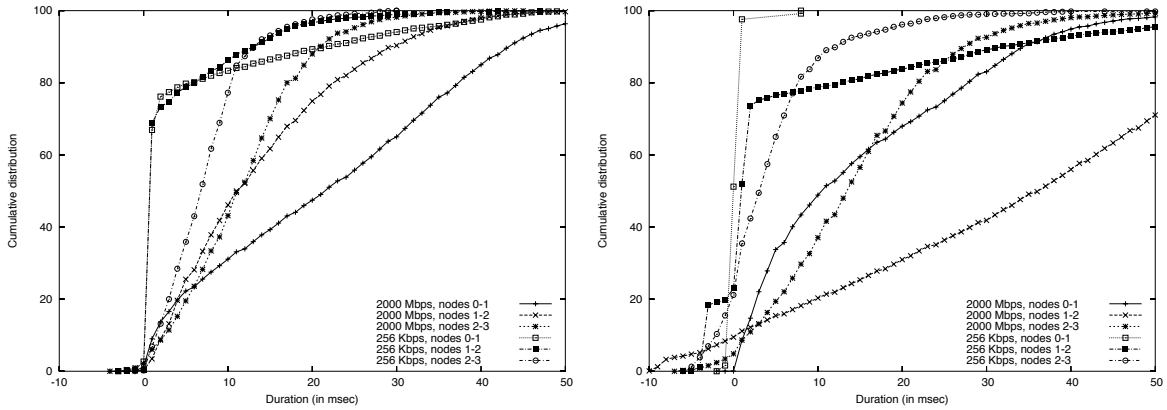
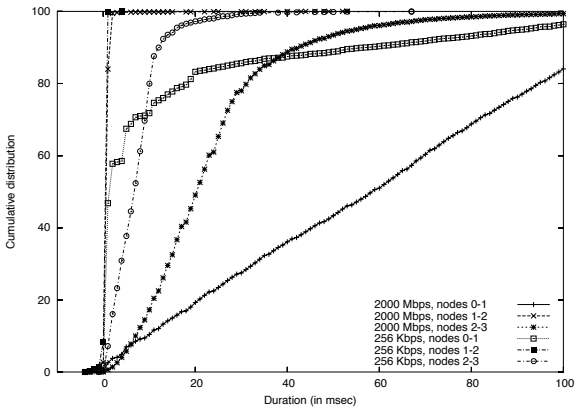


Figure 6. Ability to maintain schedules across wireless channel



(a) One WNIC in a FreeBSD router

(b) One WNIC in a Windows XP router



(c) Two WNIC (one each for forward and reverse traffic) in a FreeBSD router

Figure 7. Cumulative distribution of measured drift in reception $(r_i - r_1) - (s_i - s_1)$

achievable for various R_i intervals for our reference streams in Figure 5; assuming an average clock phase skew interval of 5 msec. We note that R_i should be high so as to achieve better and consistent power savings closer to the *Oracle* policy.

4.2.2. Ability to maintain a schedule across the wireless link

In the previous section, we investigated the implications of the inability of the receiving end to wake up at the required time. We showed the need to wake up at least 10 msec before the scheduled interval to offset this delay. In this section,

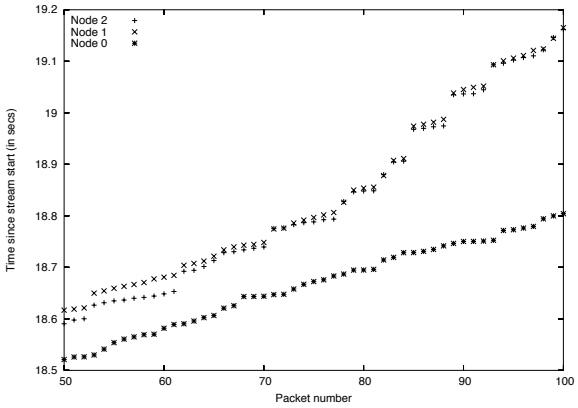


Figure 8. Effects of a context switch from routing operation (Windows XP routers)

we investigate whether packets that were shaped and transmitted can be received at the commodity receiver at the predetermined intervals. Using traffic shaping mechanisms (as illustrated in Figure 6) packets transmitted at time s_i are received at r_i (measured using local clocks which may not be synchronized). Our experiments are designed to verify whether $(s_i - s_1)$==$(r_i - r_1)$. Note that we are not concerned about the wireless transmission delay itself $(r_i - s_i)$ (as long as this delay is predictably constant).

For these experiments, we browsed the reference Microsoft media streams at 2000 kbps and 256 kbps and used tcpdump to capture the packet trace on all the intermediate routers as well as the sender and receiver. We repeated the experiments with FreeBSD 5.1 as well as Windows XP for the routing nodes. We repeated the experiments utilizing a single NIC for the routers as well as 2 WNICs, eliminating interference within the forwarding traffic. Each link chose a different non-overlapping 802.11b channel. Note that we could not install two WNICs on the Windows XP machine because of driver conflicts. We plot $(r_i - r_1) - (s_i - s_1)$ values as a cumulative distribution in Figure 7. Ideally, we expect the results to be zero for perfect reception of shaped data. However, we notice that there is a significant difference between these intervals.

From Figure 7(a), we note that for FreeBSD routers with a single NIC forwarding traffic for a high quality stream such as Microsoft media stream at 2000 Kbps, over 20% of the packets show a variation of more than 10 msec. The variation is higher for the browser to the last router link; the browser station shares its CPU resources between receiving the media stream and rendering them. However, for a 256 Kbps stream, over 60% of the packets are delayed by more than 10 msec with 10% of the packets delayed by more than 30 msec. We notice similar results using Windows XP for the routing stations (Figure 7(b)). We also notice a longer tail with wider variation for high bandwidth streams. For example, 50% of the packets are delayed by more than 40 msec between the Windows XP routers.

Similarly, in Figure 7(c), we illustrate the router nodes with two WNICs to eliminate interference between the forwarded traffic. For streaming mechanisms such as MS media that utilize network fragmentation to transmit large packets, the server side proxy would send fragments back to back, which the router node tries to forward back to back to the browser. The 802.11 DCF is used to avoid collisions; the node backs off and retries the transmission. In our ad hoc scenario, the steady stream of packets from the server side proxy and transmissions to the browser manifests itself in frequent back-offs and increased packet delays. In order to isolate the effects of this IEEE 802.11 collision avoidance mechanism, we installed two WNIC cards on the router nodes; each operating on a different channel and communicating with the server side proxy and browser respectively. We notice that these two WNIC does help the traffic between the FreeBSD router node themselves; the interval is almost zero. However, for the packets between the routers and the browser and server, the interval is much larger than with a single router. We are further investigating this strange effect.

The forwarding operation also competes for resources with other processes. The Windows XP routers share their CPU resources with other local GUI processes. Our FreeBSD based routers were not running any windowing system. The effect of these other processes on Windows XP are illustrated in Figure 8. Figure 8 plots the local time that a certain packet was received in the router node since the reception of the first packet in the streams. Notice the delay in packet receptions at Nodes 1 and 2 around 85th packet (possibly from a context switch on a routing node). The trend in mobile devices is more

towards GUI based mobile devices rather than a text based FreeBSD routers. Hence these added delays while the servers are servicing other clients can have a significant impact on real systems.

These results imply that using two WNICs and dedicated text based routers such as the two WNIC FreeBSD scenarios might provide us the ideal predictability demanded by rendezvous mechanism. However, in the more realistic one WNIC Windows XP scenario, the receiver may have to wake up much earlier (40-50 msec) before the expected rendezvous point to avoid missing packets, especially for high quality streams.

4.2.3. Robustness of t_{i1}, t_{i2}, t_{i3} and t_{i4} intervals

Constant bit rate (CBR) streams assist traffic scheduling mechanisms by allowing the system to renegotiate fewer transmission interval changes. For a VBR traffic, the schedules may have to be adjusted frequently. Without such renegotiations for VBR traffic, a smaller estimate could lead to extra delays and/or data loss (buffer overflow across scheduled intervals) while a larger estimate would lead to decreased throughput and wasted channel utilization. In this section, we analyze the various stream formats to measure the similarity of these popular streaming formats to a CBR traffic.

First we plot the projected time needed to transmit buffered data (calculated from the amount of data that needs to be transmitted) for R_i intervals of 100 msec and 500 msec in Figure 9. We chose high bandwidth streams for the plot as these streams carry much data. We expect the transmission times to be flat with little variance for a CBR traffic. From Figure 9, we note that the streams do not behave as a CBR traffic for any of the three formats explored (Microsoft media, Real and Quicktime). The streams stabilize somewhat for R_i interval of 500 mseconds. Hence, increasing the R_i parameter can have the side effect of reducing the t_i adaptation needed.

In summary, we noted that the scheduling clock phase skew and scheduling other processes can significantly affect the potential energy savings possible with a traffic shaping mechanism. In order to counteract the effects of clock skew, we need to **increase the R_i/S_i intervals** as much as possible. Increasing the repeat interval (R_i,S_i) not only reduces the relative penalty introduced by clock phase skew (up to 10 msec), but also increases the amount of data buffered, thereby smoothing the VBR media traffic as well as increasing the transmission intervals t_i (increasing the amount of useful work to counteract the delay introduced to transition the devices to the various power saving states). However, such increased interval can trigger client side adaptations (detected as network congestion) as well as tax the WNIC MAC collision avoidance mechanisms (WNIC buffering and increased delays because of collision back offs).

4.3. Experimental results for the traffic shaping mechanism

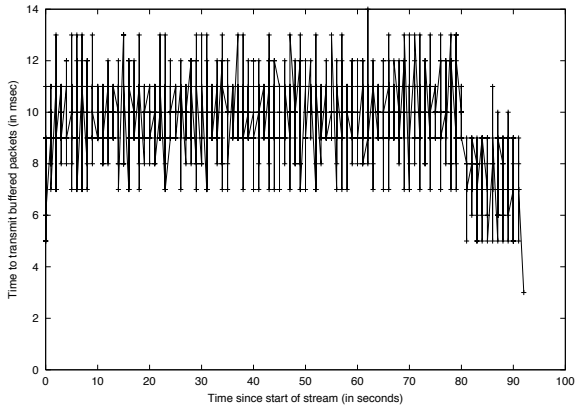
Based on the observations from the previous section, we built a traffic shaping mechanism. In this section, we describe the experiment setup and experimental results from our setup.

4.3.1. Experiment setup

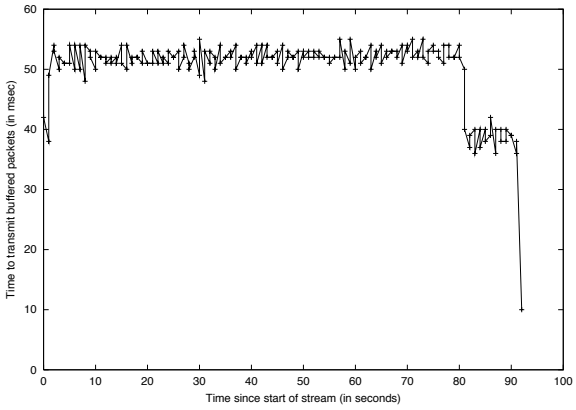
Our primary goal in building the traffic shaping mechanism was to keep the changes as simple and non-intrusive as possible. The multimedia server was connected to the server side proxy using a 100 Mbps full-duplex crossover cable (to leverage our experience with FreeBSD based mechanisms). The server side proxy, single router and browser clients were 2 GHz Pentium IV-M laptops with 512 MB main memory. The server side proxy and the router ran FreeBSD 5.0-Current, while the browser laptop ran Windows XP.

We implemented the traffic mechanism on the server side proxy using FreeBSD divert sockets interface. The kernel diverted all packets destined for the mobile client to the traffic shaper. The traffic shaper buffered the packets and periodically transmitted them. Note that the traffic shaping mechanism itself is not aware of the multimedia nature of the network packets. We implemented two variants of this traffic shaping mechanism; each of which maintained constant R_i and S_i. We used a timer based approach to maintain constant R_i (Figure 10) and a *usleep()* based mechanism to maintain constant S_i (Figure 11).

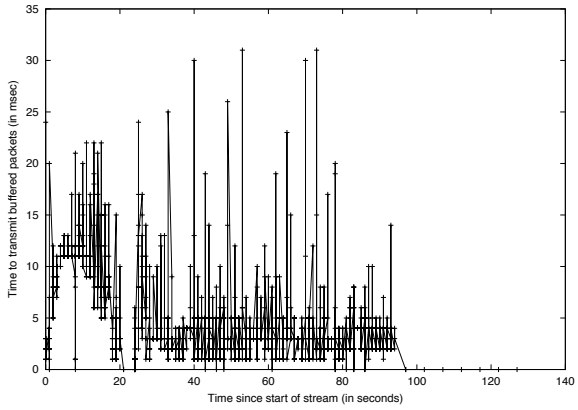
We imagine that the router nodes learn the traffic patterns and switch between receiving data from upstream and forwarding it to the downstream nodes. In order to effectively design such a mechanism, we wanted to measure the rate at which data packets were received in the router and the browser nodes. In our ad hoc scenario, the steady stream of packets from the server side proxy and transmissions to the browser manifests itself in frequent back-offs and increased packet delays. In our proposed traffic shaping mechanism, the router nodes can store the packets and transmit them separately, avoiding the dependency on MAC level implementations. In order to isolate the effects of this IEEE 802.11 collision



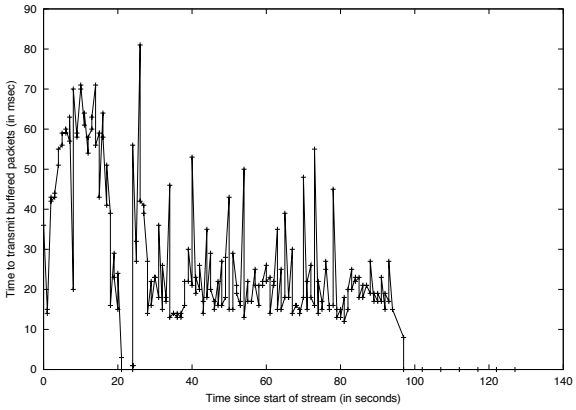
(a) Real 512 kbps ($R_i = 100msec$)



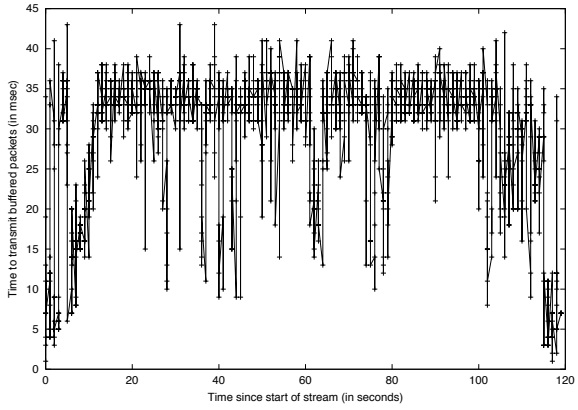
(b) Real 512 kbps ($R_i = 500msec$)



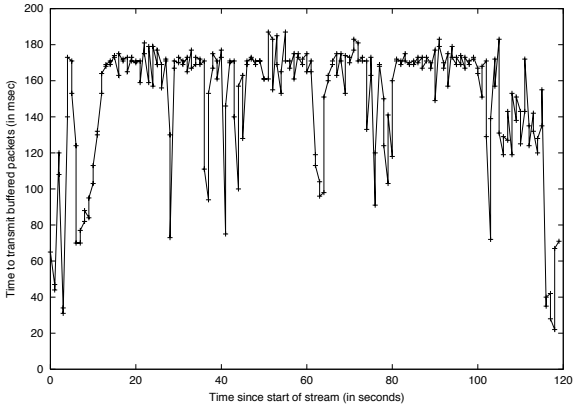
(c) Quicktime 256 kbps ($R_i = 100msec$)



(d) Quicktime 256 kbps ($R_i = 500msec$)



(e) MS Media 2000 kbps ($R_i = 100msec$)



(f) MS Media 2000 kbps ($R_i = 500msec$)

Figure 9. Time required to transmit multimedia data collected in the past R_i interval (high bandwidth streams)

Main thread:
M1. set interrupt timer for every 10 msec
M2. forever *read and buffer data*
Interrupt handler:
I1. Check to see if data received in the last 10 msec (and transmit all data)
I2. If no data received; transition WNIC to *sleep* state for calculated sleep interval (R_i-*time to receive useful data in the current burst*)

Figure 10. Server side shaper that maintains constant repeat interval R_i

Main thread:
M1. forever *read and buffer data*
Thread handler:
I1. Check to see if data received in the last 10 msec (and transmit all data)
I2. If no data received; transition WNIC to *sleep* state for calculated sleep interval (S_i)

Figure 11. Server side shaper that maintains constant sleep interval S_i

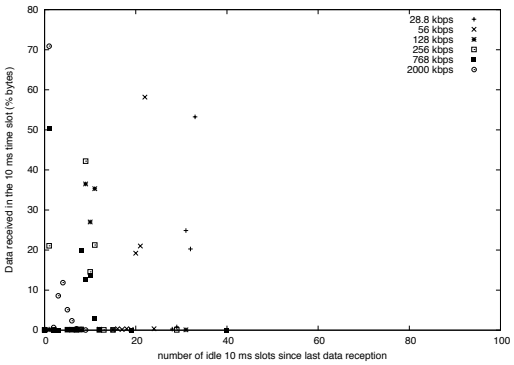
avoidance mechanism, we installed two WNIC cards on the router nodes; each operating on a different non-overlapping channel and communicating with the server side proxy and browser respectively. The packet traces were captured using tcpdump. In the following sections, we analyze the packet reception characteristics in the router and browser nodes for the different traffic shaping mechanisms.

4.3.2. Shaping mechanism that maintains constant repeat interval (R_i)

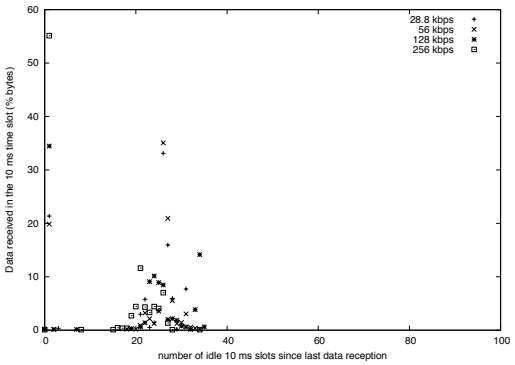
For these experiments, the traffic shaping server side proxy tried to maintain constant repeat interval R_i using timer mechanisms. However, other background tasks, network buffers and other mechanisms can delay packets and alter the times at which data is received in the clients. Since the energy saving mechanisms depend on being able to predict these intervals, we wanted to measure the times at which packets are received in the clients. We utilized tcpdump to capture the packet arrival times. We sorted data items by the number of idle 10 msec slots till next data transmission and plotted the amount of data transferred in each of these slots as a percentage for Microsoft media, Quicktime and Real streams in Figures 12, 12 and 13, respectively. Ideally, we expect the data to have two spikes, one around 1 (signifying that there is continuous transmission of data across a 10 msec slot) and n (where $n = delay/10$). In such an ideal case, the routers can wait for one extra 10 msec time slot after receiving data items; if no data items were received then the WNIC can be transitioned to the *sleep* state for $n - 2$ (one for the extra probing and one for lack of clock phase) 10 msec time slots without losing any data.

From Figure 12(a), we note that for the base case without any traffic shaping mechanisms, the delay between packets are varied and hovered around 100 msec. Figures 12(c) and 12(e) plot the results for a policy that maintains constant repeat interval (R_i) of 200 msec and 500 msec respectively. For a policy that maintains a delay of 200 msec, we note that most of the traffic are delivered back to back (for a delay of 1 in the graph) or 20 or 40 (corresponding to 200 or 400 msec). Lower quality streams send most of their data in 200 msec, while higher quality streams send significant amounts of data close together (70% for 2000 Kbps stream). For a delay of 500 msec, we notice a similar effect with most objects transmitted back to back for higher quality streams (65% for 2000 kbps streams). For these experiments, we configured the browser to automatically buffer as much network data as possible to offset the sporadic nature of the network traffic. However, for high quality streams, these mechanisms were not able to fully compensate for the delays; the browser adapted to a lower quality stream. We are investigating schemes to offset this adaptation.

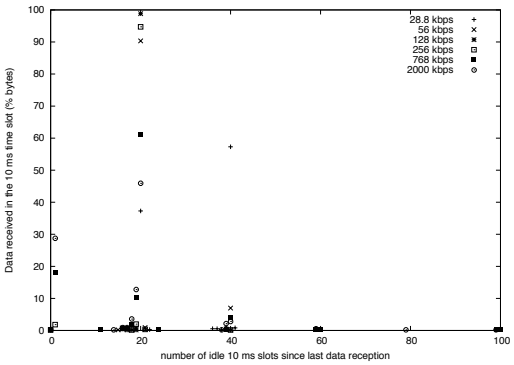
Similarly, we repeat the experiments for Quicktime (Figure 12) and Real (Figure 13). We noticed similar results as the Microsoft media. We noticed that higher quality streams tended to send more data back to back occupying contiguous time slots. Note that in all cases, we configured the respective browsers with recommended buffering to offset the effects of the added delays. We also measured the stream behavior on the browser node and verified that the stream shaping was still valid at the browser node without additional shaping at the router nodes. In general, the results were similar to the routers and are not illustrated for lack of space.



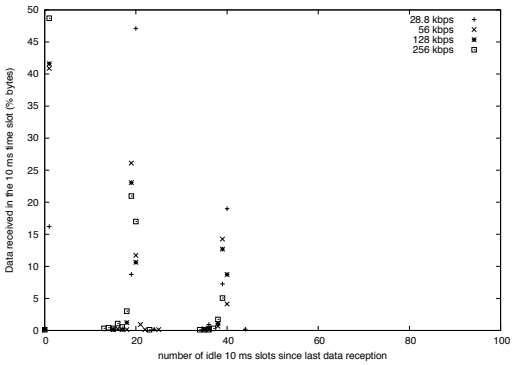
(a) baseline - no proxy (MS media, router node)



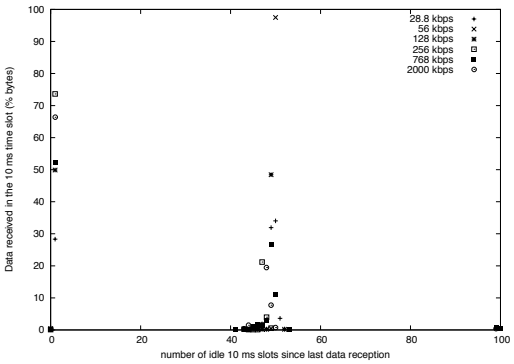
(b) baseline - no proxy (Quicktime, router node)



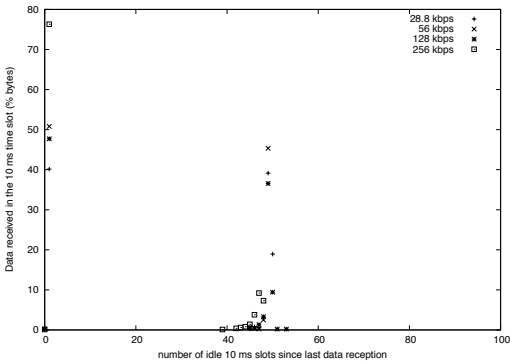
(c) $R_i = 200msec$ (MS media, router node)



(d) $R_i = 200msec$ (Quicktime, router node)

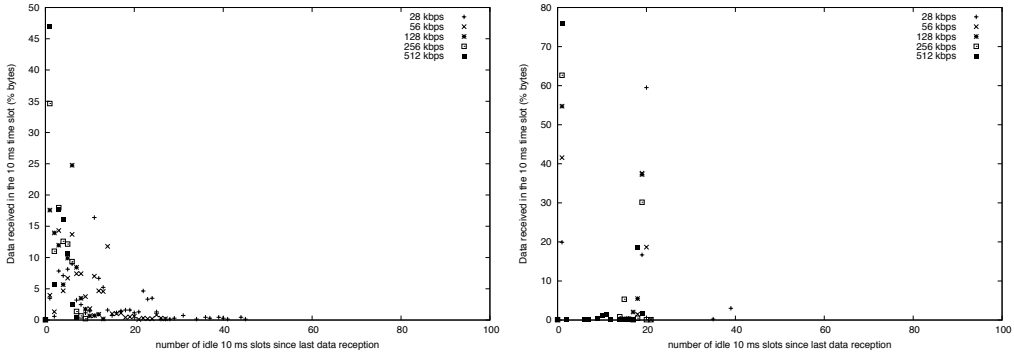


(e) $R_i = 500msec$ (MS media, router node)



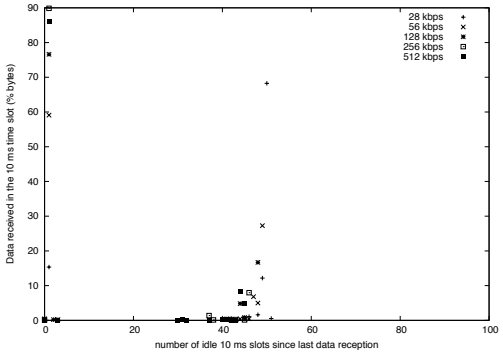
(f) $R_i = 500msec$ (Quicktime, router node)

Figure 12. Time required to transmit multimedia data collected in the past R_i interval (plotted as a histogram)



(a) baseline - no proxy

(b) $R_i = 200msec$



(c) $R_i = 500msec$

Figure 13. Time required to transmit multimedia data collected in the past R_i interval (plotted as a histogram) (Real, router node)

4.3.3. Shaping mechanism that maintains a constant sleep interval (S_i)

We also repeated the above experiments for a proxy that maintained constant sleep S_i intervals and measured the performance both at the router and browser nodes. In general, the results were similar and are not illustrated for lack of space.

4.4. Current Implementation status

We have built our FreeBSD based proxy mechanism and are continuing experimentation with realistic scenarios including cross traffic from other ad hoc clients. Also in the last section, we performed experiments on a FreeBSD based setup with two WNIC interfaces. However, we expect realistic systems to utilize a single WNIC interface as well as share the CPU resources for a graphical user interfaces. We are continuing experimentation with our proxy using a single WNIC interface. We are also exploring building shaping mechanisms for Windows XP and Linux so that we can experiment with iPAQs running Linux and Windows laptops.

Our proxy at present logs durations at which a WNIC power transition should occur. We have implemented a proxy mechanism that simulates transitions for the WNIC interface. Even though we did not have access to the device drivers to effect an energy transition, our proxy mechanism actually ran on the target machine (along with other production applications) and simulated the transition operation. Under such a realistic setup, the device transition predictions were less accurate, We are exploring ways of programmatically transitioning the network interface to a lower energy consuming *sleep* state so that we can measure the actual energy consumed with our proposed mechanism on a realistic network.

5. DISCUSSION

In this paper, we examined the practical limitations in achieving energy savings for commodity devices servicing multimedia streams using popular formats in an ad hoc network environment. We utilized a traffic shaping mechanism to schedule network data at predetermined intervals; avoiding unnecessary waits in higher energy consuming *idle* states. Our work makes the following contributions towards the design of such application specific energy-aware network traffic shaping mechanisms:

- We show that the lack of clock synchronization among the nodes favors infrequent data transmission intervals
- We showed that CPU contention with other competing processes favors infrequent data transmission intervals to amortize the possibility of waiting for prolonged intervals
- Using a large delay can lead to significant energy savings on the media browser as well as the intermediate routers
- Increased delays complicate the design of intermediate traffic shaping proxies that not only need to buffer more data packets, but also have to smooth the delivery intervals so as to avoid client adaptation of these added intervals as an indication of congested networks. Buffering mechanisms built into media browsers can tolerate delays of a few hundred mseconds.

In general, our traffic shaping mechanism expects the intermediate router nodes to place the highest importance to routing traffic. Typical operating systems do not offer mechanisms to express such requirements (global vs local policy decisions) nor provide support for effecting this changes. Without such facilities, traffic shaping mechanisms have to increase the scheduling delay in order to amortize the added costs. We are exploring OS mechanisms that can allow us to better express our routing requirements.

REFERENCES

1. Real Player. http://www.real.com/.
2. Microsoft Windows Media Technologies. http://www.microsoft.com/windowsmedia/.
3. Apple Quicktime. http://www.apple.com/quicktime/.
4. M. Stemm, P. Gauthier, D. Harada, and R. H. Katz, "Reducing power consumption of network interfaces in handheld devices," in *Proceedings of the 3rd International Workshop on Mobile Multimedia Communications (MoMuc-3)*, (Princeton, NJ), Sept. 1996.
5. S. Cho, "Power Management of iPAQ," Feb. 2001.
6. J. Lorch and A. J. Smith, "Software Strategies for Portable Computer Energy Management," *IEEE Personal Communications Magazine* **5**, pp. 60–73, June 1998.
7. P. J. M. Havinga, *Mobile Multimedia Systems*. PhD thesis, Univ. of Twente, Feb. 2000.
8. P. Agrawal, J.-C. Chen, S. Kishore, P. Ramanathan, and K. Sivalingam, "Battery power sensitive video processing in wireless networks," in *Proceedings IEEE PIMRC98*, (Boston), Sept. 1998.
9. S.-T. Sheu and T.-F. Sheu, "DBASE : A Distributed Bandwidth Allocation/Sharing/Extension Protocol for Multimedia over IEEE 802.11 Ad Hoc Wireless LAN," in *Proceedings of IEEE INFOCOM 2001*, **3**, pp. 1558–1567, (Anchorage, Alaska), Apr. 2001.
10. R. Rozovsky and P. R. Kumar, "SEEDEX: A MAC protocol for ad hoc networks," in *Proceedings of The ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, pp. 67–75, (Long Beach, CA), Oct. 2001.
11. Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh, "Power-saving protocols for ieee 802.11-based multi-hop ad hoc networks," in *Proceedings of IEEE INFOCOM 2002*, (New York), June 2002.
12. R. Krashinsky and H. Balakrishnan, "Minimizing energy for wireless web access with bounded slowdown," in *Proceedings of the 8th ACM International Conference on Mobile Computing and Networking (MobiCom '02)*, pp. 119–130, (Atlanta, GA), Sept. 2002.
13. J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris, "Capacity of ad hoc wireless networks," in *Proceedings of the 7th ACM International Conference on Mobile Computing and Networking (MobiCom '01)*, pp. 61–69, (Rome, Italy), July 2001.

14. K. M. Sivalingam, J.-C. Chen, P. Agrawal, and M. B. Srivastava, "Design and analysis of low-power access protocols for wireless and mobile ATM networks," *ACM/Baltzer Wireless Networks Journal* **6**, pp. 73–87, Feb. 2000.
15. P. Shenoy and P. Radkov, "Proxy-assisted power-friendly streaming to mobile devices," in *Proceedings of the 2003 Multimedia Computing and Networking (MMCN)*, (Santa Clara, CA), Jan. 2003.
16. L. M. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," in *Proceedings IEEE INFOCOM 2001*, **3**, pp. 1548–1557, (Anchorage, Alaska), Apr. 2001.
17. L. M. Feeney, "An energy-consumption model for performance analysis of routing protocols for mobile ad hoc networks," *Journal of Mobile Networks and Applications* **3**(6), 2001.
18. L. M. Feeney, "A qos aware power save protocol for wireless ad hoc networks," in *Med-Hoc-Net 2002*, (Sargegna, Italy), Sept. 2002.
19. S. Chandra, "Wireless network interface energy consumption implications of popular streaming formats," in *Multimedia Computing and Networking (MMCN'02)*, M. Kienzle and P. Shenoy, eds., **4673**, pp. 85–99, SPIE - The International Society of Optical Engineering, (San Jose, CA), Jan. 2002.
20. LAN/MAN Standards Committee of the IEEE Computer Society, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher-speed Physical Layer Extension in the 2.4 GHz Band*. IEEE, 3 Park Avenue, New York, NY 10016, 1999.
21. S. Chandra and A. Vahdat, "Application-specific network management for energy-aware streaming of popular multimedia formats," in *Proceedings of the USENIX Annual Technical Conference*, pp. 329–342, USENIX, (Monterey, CA), June 2002.
22. E. Royer and C.-K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *IEEE PERSONAL COMMUNICATIONS* , pp. 46–55, Apr. 1999.
23. M. B. Jones and J. Regehr, "The problems you're having may not be the problems you think you're having: Results from a latency study of windows nt," in *Proceedings of the 7th Workshop on Hot Topics in Operating Systems (HotOS VII)*, (Rio Rico, AZ), Mar. 1999.
24. A. Goel, L. Abeni, C. Krasic, J. Snow, and J. Walpole, "Supporting time-sensitive applications on general-purpose operating systems," in *Proceedings of the Fifth USENIX Symposium on Operating Systems Design and Implementation*, USENIX, (Boston, MA), Dec. 2002.