

# Predictable Scheduling for a Soft Modem

**Stefan Saroiu**

Department of Computer Science & Engineering  
University of Washington  
Seattle, WA 98195-2350, USA  
tzoompy@cs.washington.edu  
<http://www.cs.washington.edu/homes/tzoompy>

**Michael B. Jones**

Microsoft Research, Microsoft Corporation  
One Microsoft Way, Building 112/2156  
Redmond, WA 98052, USA  
mbj@microsoft.com  
<http://research.microsoft.com/~mbj/>

## Abstract

*Soft Modems* use the main processor to execute modem functions traditionally performed by hardware on the modem card. To function correctly, soft modems require that ongoing signal processing computations be performed on the host CPU in a timely manner. Thus, signal processing is a commonly occurring background real-time application—one running on systems that were not designed to support predictable real-time execution. This paper presents a detailed study of the performance characteristics and resource requirements of a popular soft modem. Understanding these requirements should inform the efforts of those designing and building operating systems needing to support soft modems. Furthermore, we believe that the conclusions of this study also apply to other existing and upcoming soft devices, such as soft Digital Subscriber Line (DSL) cards. We conclude that (1) signal processing in an interrupt handler is not only unnecessary but also detrimental to the predictability of other computations in the system and (2) a real-time scheduler can provide predictability for the soft modem while minimizing its impact on other computations in the system.

## 1. Introduction

*Soft Modems* use the main processor to execute modem functions traditionally performed by the digital signal processor (DSP) on the modem card. While soft modem hardware retains A/D and D/A functions, CPU-intensive computations, such as signal processing, are performed on the host CPU. We measured a 14.7% sustained CPU load on a Pentium II 450 MHz machine. Because soft modems do periodic real-time computations on the host CPU in order to maintain line connection and transmit data, a mechanism ensuring periodic computations is essential.

We analyzed the driver of a popular soft modem<sup>1</sup> in order to better understand its real-time characteristics. Currently, the vendor version performs its digital signal processing work in interrupt context. We modified this driver to produce three additional versions by executing the signal processing code in the context of a Deferred Procedure Call (DPC), a thread scheduled by the Windows NT scheduler and, finally, a thread scheduled using the CPU Reservation abstraction of the real-time Rialto/NT scheduler [Jones & Regehr 99], respectively.

We conclude that signal processing in interrupt context is not only unnecessary but also detrimental to the predictability of other computations in the system. While DPCs

and time-based scheduling have milder side effects upon the rest of the system, nevertheless, they suffer from some of the same drawbacks as the original version. Real-time scheduling for commodity systems provides a good answer to the observed predictability problems.

Finally, we believe many of the lessons learned from studying soft modems are applicable to a wider class of problems. Other soft devices are already in use, with more coming soon. For instance, software-based *Digital Subscriber Line* (Soft DSL) [Tramontano 00] devices have been announced.

## 2. Soft Modem Driver Versions

The soft modem uses Direct Memory Access (DMA) to transfer the data between the A/D and D/A, and the physical memory. There are four different buffers—two output buffers (one for data and one for voice samples) and two input buffers. Each buffer has a size of 512 16-bit samples, for a total of 1024 bytes. Our experiments traced the data buffers only.

We compared and contrasted four versions of the soft modem driver:

- *Vendor version (INT)* – in this initial version of the driver, when the card interrupts the CPU, the driver software performs the signal processing inside the interrupt handler (a.k.a. Interrupt Service Routine or ISR). Both outgoing and incoming samples are processed.
- *DPC version (DPC)* – this version executes the signal processing code in the context of a DPC. When the modem card raises an interrupt, the ISR queues a DPC.
- *Thread version (THR)* – in this version, the signal processing routines are executed in the context of a regular thread with a given priority and scheduled by the Windows NT scheduler.
- *Rialto/NT version (RES)* – in the final version of the driver, the signal processing thread is scheduled using the Rialto/NT scheduler using a *CPU Reservation*.

Threads make CPU Reservations to ensure a minimum guaranteed execution rate and granularity. CPU Reservation requests are of the form *reserve X units of time out of every Y units for thread A*. The current implementation of Rialto/NT has two restrictions: (1) CPU reservations must have values that are integer multiples of milliseconds since they are driven off the periodic Windows NT clock and (2) the period of a reservation must be a power-of-two multiple of a millisecond due a choice of algorithms within Rialto/NT [Jones & Regehr 99].

<sup>1</sup> Our agreement with the manufacturer prevents us from identifying this soft modem.

### 3. Results

The measurements presented in this section use a test scenario of a dial-up connection with a normal priority spinning competitor thread. We use the term *elapsed time* to measure the time spent by the signal processing routines. The elapsed time incorporates the time spent in other contexts that preempted the signal processing routines.

#### 3.1 Soft Modem Interrupt Rate

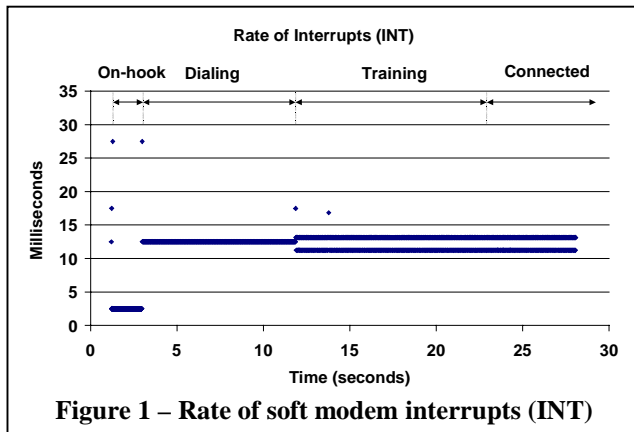


Figure 1 – Rate of soft modem interrupts (INT)

Figure 1 shows the interrupt rate for the INT version. Before dialing, the interrupts occur very frequently (every 2.5ms) for about 2 seconds. The modem is on-hook performing ring detection. For the next 10 seconds, the interrupt rate is 12.5ms while the modem is dialing and waiting for the other end to start the connection. Whenever there is a change in DMA frequency or in the size of the samples buffers, the modem requests an interrupt frequency change. This request causes a short delay in the interrupt rate that corresponds to the six scattered points in the graph. Once the V.90 protocol has been established, the interrupt occurs every 13.125ms or 11.25ms, with an average period of 12.5ms. These rates correspond to the PC 99 [Intel & Microsoft 98] recommended interrupt rates of 3 to 16 milliseconds. This interrupt rate behavior is virtually unchanged in all the other driver versions.

#### 3.2 Elapsed Times in ISR in INT Version

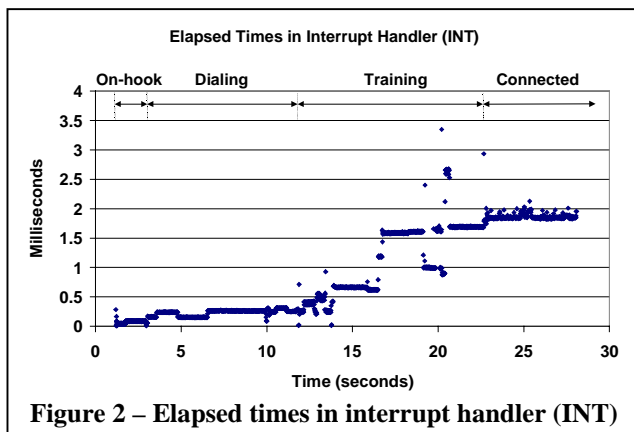


Figure 2 – Elapsed times in interrupt handler (INT)

The PC 99 specification recommends that the maximum time during which a driver-based modem disables interrupts should not exceed 100µs [Intel & Microsoft 98].

Figure 2 shows that the execution of the interrupt handler lasts 1.8ms with a repeatable worst case of 3.3ms, a factor of 33 worse than the specs recommend.

#### 3.3 CPU Utilization

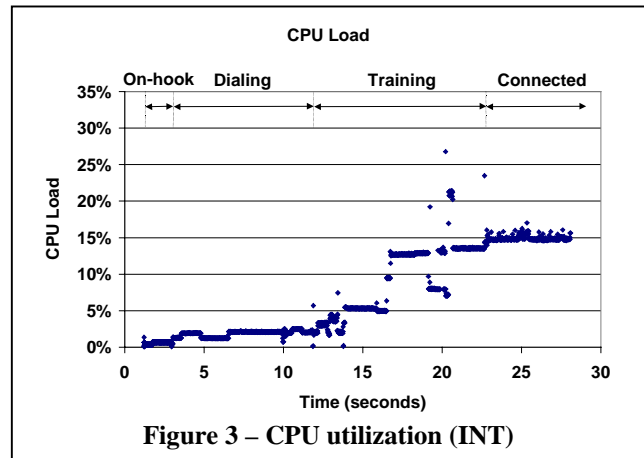


Figure 3 – CPU utilization (INT)

Figure 3 shows the CPU utilization of the soft modem. To compute the CPU load we decided to use 12.5ms (the actual interrupt rate) as the interval step. That is, we combined the time spent inside the interrupt handler executing signal processing for each 12.5ms interval throughout the trace for the vendor version. As Figure 3 illustrates, while connected, the soft modem uses a sustained 14.7% share of the CPU.

#### 3.4 Times Spent by the DPC Version

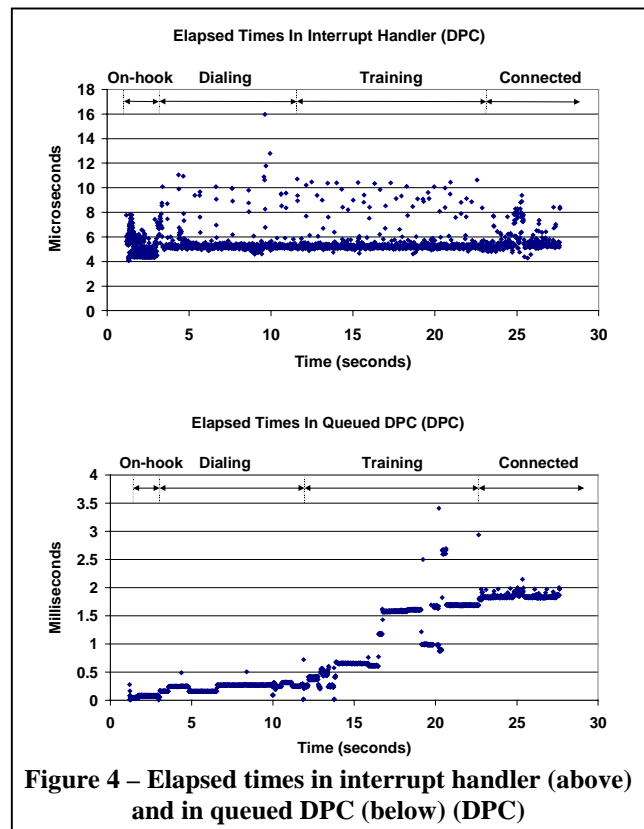


Figure 4 – Elapsed times in interrupt handler (above) and in queued DPC (below) (DPC)

Without the signal processing code, the interrupt handler does very little work. For the DPC version, the signal

processing is done in a DPC. Figure 4 shows the times spent inside of the interrupt handler and the queued DPC. In the THR and RES versions, the DPC execution time is replaced by the scheduled thread execution time.

While the ISR execution time has been reduced from milliseconds to a few microseconds, the time spent inside the DPC is still too large. The *PC 99* specifications suggest that at any instant in time, the total execution time required for all DPCs that have been queued by a driver-based modem, but have not been executed should not exceed 500 $\mu$ s.

### 3.5 Interference with Other Applications

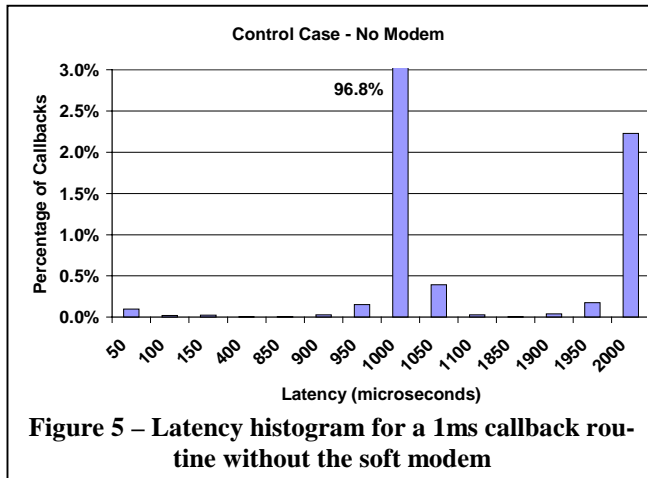


Figure 5 – Latency histogram for a 1ms callback routine without the soft modem

In order to understand the effects of long running ISRs and DPCs, we measured the observed latencies of a callback routine that uses the Windows Multimedia Timers. The timers have been set to fire every millisecond and the routine is called with priority 31, the highest priority for a real-time thread.

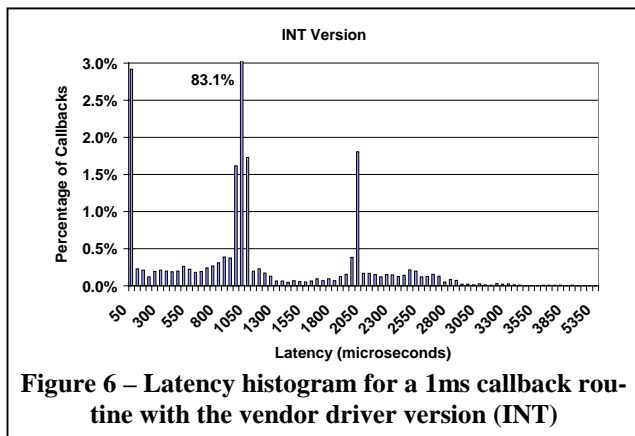


Figure 6 – Latency histogram for a 1ms callback routine with the vendor driver version (INT)

Figures 5 and 6 show histograms of the measured latencies when the soft modem is not running and for the INT version. The experiment captured 30,000 wakeups over a 30-second period. Samples are accumulated into 50 $\mu$ s buckets. While not ideal, the control case (Figure 5) shows the latencies that are achievable on an idle system.

The damage the soft modem's long-running ISR or queued DPC causes to the coexisting callback routine is evident. Notice, in particular, that the tails of the distributions increase from 2ms to over 5ms.

Figure 7 illustrates the callback latencies for the THR driver version. As before, the THR uses a priority 24 real-time thread.

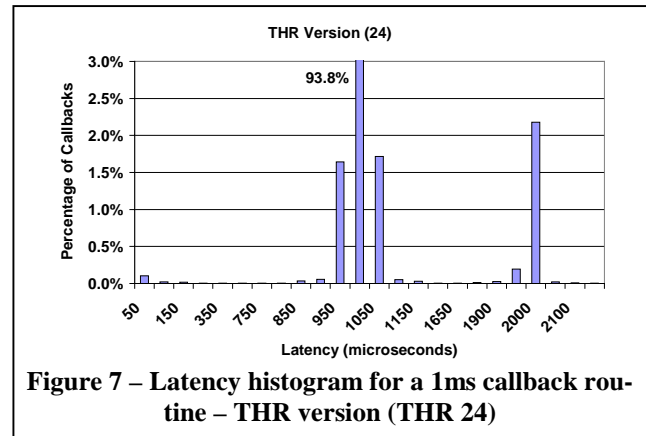


Figure 7 – Latency histogram for a 1ms callback routine – THR version (THR 24)

### 3.6 Reflections Upon the Vendor Choice

Given that thread-based signal processing works well and causes less disruption of coexisting system activities, why then might a vendor still chose to perform signal processing in DPCs or in interrupt handlers? Vendors face a problem common to all priority-based open systems (ones in which independently authored applications and/or drivers may be executed): for any chosen priority, there is a potentially unbounded delay until a thread is scheduled to run. These delays can be caused by other applications running for arbitrary periods of time at the chosen or higher priority. Thus, *no timing guarantees can be made*.

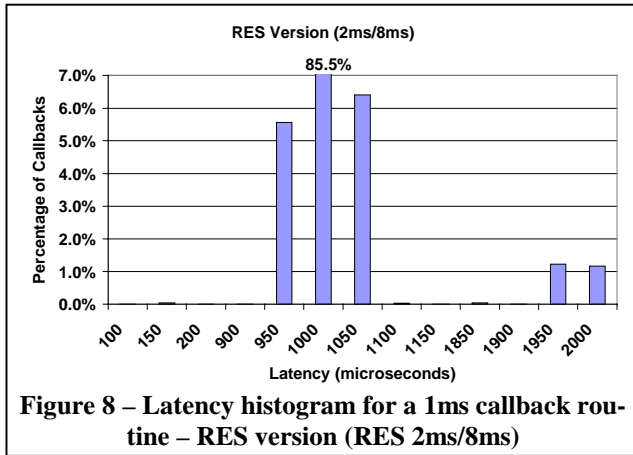
For systems with a fixed priority preemptive scheduler, like Windows NT, it is possible to use Rate Monotonic Analysis (RMA) to determine when each of the system threads should be scheduled to run according to their deadlines. Unfortunately, RMA cannot be employed due to the following reasons:

- RMA assumes cooperation between the threads, which is unrealistic given the existence of different drivers written by different vendors
- RMA assumes constant timing requirements for all the coexisting threads. More precisely, whenever a thread changes its CPU requirements, it potentially affects all the other coexisting threads.

We believe that a better alternative to RMA would be a real-time scheduler, such as Rialto/NT. The coexisting threads could then reserve ongoing portions of the CPU, using the *CPU Reservation* abstraction.

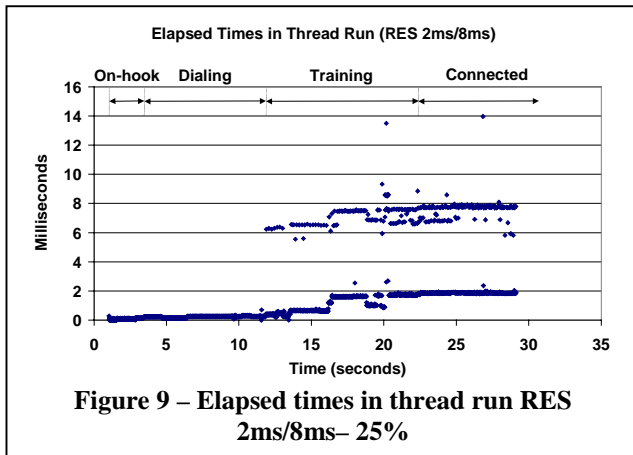
### 3.7 Coexistent Thread Latencies in RES Version

After trying different values for the *CPU Reservation* and due to the coarse-grained accuracy of the Rialto/NT prototype, we decided to use a 2ms/8ms reservation for the soft modem. Figure 8 shows the callback latencies for the RES version with a 2ms/8ms *CPU Reservation*. The predictability of the callback routine improves substantially over the INT and DPC versions. Compared to the THR version shown in Figure 7, there are two times more callbacks occurring within 50 $\mu$ s from 1ms.



### 3.8 Elapsed Times per Wakeup in RES Version

Figure 9 presents the work done by the signal processing thread when scheduled with a 2ms/8ms (25%) CPU Reservation.



While larger as a percentage than the actual modem requirements, a 2ms/8ms reservation is not an ideal match for the soft modem processing routines. The period of 8ms causes the signal processing thread to be scheduled to execute at different times than the occurrences of the interrupt. Whenever scheduled, the thread will cede its reservation to the normal spinning competitor if it is not in a ready state. This explains the long processing times of single thread runs, since they also incorporate the times spent in contexts that preempted the measured thread. However, despite the period mismatch, this reservation does allow the modem to operate perfectly, as the results in the next section show.

### 3.9 End-to-End Modem Download Throughput

We also measured the time required to transfer a file of 200,000 bytes. For each version of the driver, 10 copies of this file were transferred. Table 1 contains statistics about the transfer times recorded in seconds. For all the tests except for the THR priority 8, a normal priority spinning thread was executing in parallel with the file transfer.

The 2ms/8ms and 1ms/4ms reservations (25% CPU) behaved almost identical to the INT and the DPC version, whereas the 1ms/8ms (12.5%), 3ms/16ms (18.75%) and 2ms/16ms (12.5%) either needed a longer amount of time

for transfers or were accompanied by occasional modem disconnections.

	Min	Max	Mean	Std Dev	Succ
INT	37.914	37.258	37.222	0.019	10
DPC	37.151	37.303	37.229	0.051	10
THR Pri 8	59.899	60.658	60.219	0.228	10
THR Pri 24	37.147	40.648	37.560	1.086	10
RES 2ms/16ms	156.632	240.932	204.146	35.447	10
RES 3ms/16ms	37.864	122.042	76.840	30.775	10
RES 1ms/8ms	43.741	83.336	56.237	10.73	9
RES 2ms/8ms	37.086	37.242	37.175	0.053	10
RES 1ms/4ms	37.118	38.823	37.354	0.518	10

**Table 1 – File transfer time (seconds) of 200,000 bytes**

## 4. Conclusions

First, signal processing in interrupt context is not only unnecessary, but also detrimental to the predictability of any coexisting activity.

Second, the DPC version has some of the same predictability drawbacks as the vendor version. Both the vendor and the DPC versions do not correspond to the PC 99 set of recommendations for the Windows NT driver writers [Intel & Microsoft 98].

Third, the NT scheduled thread version helps alleviate some of these problems. We found that the soft modem driver functions well when the signal processing thread has high real-time priority and no competition.

Fourth, we conclude that other threads are less interfered with when the modem is scheduled using the real-time CPU Reservations abstraction. In particular, this abstraction allows us to control the amounts of time that the modem interferes with other time-sensitive computations while still meeting its needs.

In summary, this study will make the detailed performance characteristics of a popular soft modem available to the industry. We believe that this data should prove useful for informing ongoing work on providing predictable execution on consumer and general-purpose operating systems.

## References

- [Jones & Regehr 99] Michael B. Jones and John Regehr. CPU Reservations and Time Constraints: Implementation Experience on Windows NT. In *Proceedings Third USENIX Windows NT Symposium*, Seattle, WA, pages 93-102, July 1999.
- [Intel & Microsoft 98] *PC 99 System Design Guide – A Technical Reference for Designing PCs and Peripherals for the Microsoft Windows Family of Operating Systems, Chapter 19 – Modems*. Intel Corporation and Microsoft Corporation, 1998. [http://download.intel.com/design/pc98/pc99/Pc\\_99\\_1.pdf](http://download.intel.com/design/pc98/pc99/Pc_99_1.pdf).
- [Tramontano 00] Mike Tramontano. *The DSL Market is Going Soft*. Inter@ctive Week Online, Ziff Davis, July 17, 2000. <http://www.zdnet.com/intweek/stories/news/0,4164,2604854,00.html>.