Optimization of a Low-Cost Truly Preemptive Multitasking PC Diagnostic Workstation

Srinka Ghosh, Katherine P. Andriole, David E. Avrin, and Ronald L. Arenson

The Windows 95/NT operating systems (Microsoft Corp, Redmond, WA) currently provide the only low-cost truly preemptive multitasking environment and as such become an attractive diagnostic workstation platform. The purpose of this project is to test and optimize display station graphical user interface (GUI) actions previously designed on the pseudomultitasking Macintosh (Apple Computer, Cupertino, CA) platform, and image data transmission using time slicing/dynamic prioritization assignment capabilities of the new Windows platform. A diagnostic workstation in the clinical environment must process two categories of events: user interaction with the GUI through keyboard/mouse input, and transmission of incoming data files. These processes contend for central processing units (CPU) time resulting in GUI "lockout" during image transmission or delay in transmission until GUI "quiet time." WinSockets and the Transmission Control Protocol/Internet Protocol (TCP/IP) communication protocol software (Microsoft) are implemented using dynamic priority timeslicing to ensure that GUI delays at the time of Digital Imaging and Communications in Medicine (DICOM) file transfer do not exceed $\frac{1}{10}$ second. Assignment of thread priority does not translate into an absolute fixed percentage of CPU time. Therefore, the relationship between dynamic priority assignment by the processor, and the GUI and communication application threads will be more fully investigated to optimize CPU resource allocation. These issues will be tested using 10 MB/sec Ethernet and 100 MB/sec fast and wide Ethernet transmission. Preliminary results of typical clinical files (10 to 30 MB) over Ethernet show no visually perceptible interruption of the GUI, suggesting that the new Windows PC platform may be a viable diagnostic workstation option.

Copyright © 1997 by W.B. Saunders Company

KEY WORDS: preemptive multitasking, diagnostic workstation, optimization, CPU resources.

THE WINDOWS95/NT (Microsoft Corp, Redmond, WA) operating systems currently provide the only low cost truly preemptive multitasking (PC) environment and as such become an attractive diagnostic workstation platform. The purpose of this workstation design project is to use the pre-emptive multitasking and dynamic prioritization assignment capabilities of the Windows 32-bit operating systems (OS) to investigate and

optimize allocation of central processing unit (CPU) resources to display station graphical user interface (GUI) actions and image data transmission.

The present approach is to develop an application, based on a multithreaded architecture, to resolve contention between GUI and network activities for CPU resources. The main task is segmented into two processes: (1) GUI, and (2) file transfer protocol (using Transmission Control Protocol/Internet Protocol [TCP/IP] and WinSockets [Microsoft]). Each process is assigned a priority and has multiple threads of execution with individual thread priorities. The coupling of the thread priority with the process priority establishes a timeslicing for allocation of CPU resources to the competing tasks. The primary motivation behind the use of multithreaded architecture is to be able to maintain easy access handles to the secondary tasks (tasks performed by the multiple threads), whose efficiency can be adjusted individually to optimize the overall performance of the GUI and network activities.

ISSUES

A diagnostic workstation in the clinical environment must process two categories of events: user interaction with the GUI via keyboard/mouse input and transmission of incoming and outgoing data files. These processes contend for CPU resources, resulting in GUI "lockout" during image transmission or delay in transmission until GUI "quiet time." Consequently, TCP/IP time-outs leading to interruption in network traffic or corruption of data files can have a detrimental effect on clinical operation.^{1,2} The ideal solution must include the following two elements: The user should not ob-

---

From the Laboratory for Radiological Informatics, Department of Radiology, University of California at San Francisco, San Francisco, California.

Address reprint requests to Srinka Ghosh, 200 Monte Vista, #1, Oakland, CA 94611.

*Copyright © 1997 by W.B. Saunders Company
0897-1889/97/1003-1055\$3.00/0*

serve perceptible GUI delays during the data communication process and network communication must be completed without loss of data or interruption. The multithreaded algorithm for CPU resource allocation is described in detail in the following section. The basic premise dictating the logic is that although the GUI is a foreground event and data communication is a background event, the latter when active has a higher priority.

METHODS & MATERIALS

The application was developed in Microsoft Visual C++ (Microsoft Corp) across platforms using the universal features of Windows95 and NT 3.51 and above. The client-server TCP/IP protocol was developed using the 32-bit WinSockets libraries WSOCK32.lib and WSOCK32.DLL (Microsoft Corp). Tests performed with 16-bit WinSockets libraries (results discussed in the following section) underscore the limits imposed by a lack of dynamic priority assignment capability, thus emphasizing the advantage of a 32-bit OS. The schematics of the multithreaded client-server file transfer protocol are presented in Fig 1.

Multithreading can be described as the ability of the application to multitask within itself. It provides a way to comply with the "1/10 second rule."³ This advises that a message queue thread should not spend more than 1/10 of a second processing a message; for anything longer, a new thread should be delegated to the job. This gives the impression of concurrently processing multiple tasks. Hypothetically, this provides for a mechanism to ensure that when the GUI is interrupted by a non-GUI thread, the "lockout" will last no longer than 1/10 of a second, making it imperceptible. This also ensures that a network activity will not have to wait any longer than a 1/10 of a second for a GUI "quiet time." The process priority can be set at high, low, or normal. For this application, normal is optimal. Threads are assigned priorities, ranging from real time to low. The coupling of thread and process priorities assigns a unique base priority to a particular secondary task. As the threads in the thread queue are processed, their respective base priorities are set in focus. When the application is viewed as a whole, the resulting impression is that of a dynamic priority assignment.

Unlike a 16-bit application with fixed timeslicing, in a 32-bit application, timeslicing of CPU resources can be manipulated in real time by adjusting the dynamic priority assignment. Therefore, CPU contention is resolved, by allocating its resources

among threads and processes, in dynamically assigned slices of time. It should be mentioned that these time slices are kept as short as possible. The detailed algorithm of the client-server protocol using WinSockets⁴ is described in Fig 2.

Network connections are initiated when a file SEND or GET is requested. In this protocol, the method of asynchronous notification via the Windows messaging system is used to wait for network events to complete. To minimize system holdup during thread activity, this choice was made over polling or blocking methods.⁴ The (1) FD_CONNECT(): request for connection from a client, (2) FD_ACCEPT(): acceptance by a server, (3) FD_READ: file read, (4) FD_WRITE: file write, and (5) FD_CLOSE: remote close, are handled synchronously by threads. For example, the dynamic name resolution (an element of client connect request) determines the server address and socket port. Its completion efficiency is increased manifold by setting the associated thread priority to high, while maintaining the process priority at normal. This example elucidates that an overall normal process priority coupled with a high priority thread is more effective than an overall high process priority setting. This also ensures that the communication protocol does not consume CPU resources when in a dormant state.

In the data communication protocol, it is critical to synchronize the file receives (Fig 3f,f~) from the network buffer with the disk writes (Fig 3g). This is done via dynamic priority allocation, shown in Fig 3, with overall process priority set at normal. To prevent GUI (Fig 3a) "lockouts," disk writes should not be queued up. Although the TCP/IP file transfer speed cannot be adjusted, and in fact to protect against network traffic interruption, transfers (Fig 3e) should be processed as a high-priority event; a new file packet (Fig 3f~) should be received only after the prior disk write has been completed. The effect of these enforced interrupts on TCP/IP time-outs currently is being investigated.

RESULTS

The client-server protocol can both be run on the same or on different machines. The server already must be listening (Fig 3b) at a socket when a connection is requested by the client (Fig 3c), else the dynamic name resolution fails. The 16-bit version of the protocol has been tested on the same machine. For non-continual transfers of typical

Fig 1. Schematic of a multithreaded architecture.

Fig 2. Algorithm for client-server file transfer protocol.

clinical files (10 MB), the WinSocket 16-bit client-server application did not cause any perceptible GUI “lockouts.” Continual file transfers, however, caused a queuing up of disk writes, resulting in 0.5- to 1-second GUI holdups. Dynamic thread priority allocation is not possible in a 16-bit application. This motivated the development of the 32-bit multithreaded architecture. It should be noted that the TCP/IP speed for data communication between two nodes on different machines will not be as fast as that of two nodes on the same machine. This in itself will support the synchronization of file receives and disk writes, but the dynamic priority

assignment will allow for optimizing synchronization.

The following data for the 32-bit protocol will be presented at SCAR 97 in Rochester, MN, June 21-24: Baseline timer data for TCP/IP communication with both active and inactive GUI will be presented to demonstrate the efficiency of the algorithm in resolving CPU contention. Also GUI response time during TCP/IP communication will be clocked. All the above tests will be conducted on three “testbeds”—the same system where TCP/IP communication speed is maximum and delay in Windows Messaging is minimum, and over 10 Mbits/sec and 100 Mbits/sec ethernet connections.

Fig 3. Dynamic priority allocation for managing contention between network and GUI actions.

DISCUSSION

Currently, Macintosh-based (Apple Computer, Cupertino, CA) diagnostic display workstations developed by several groups^{1,2,5,6} provide a user friendly GUI interface and cost containment that compares with UNIX-based workstations. As demonstrated earlier (by this group),⁵ to attain multitasking capabilities, an involved algorithm had to be developed. The Macintosh operates in a non-preemptive multitasking mode in which the TCP/IP activity flag is checked at designated GUI interrupt points. The pre-emptive multitasking capabilities of Windows95/NT on a PC platform provide for a robust mechanism to resolve CPU contention, and thereby optimize clinical workstation performance.

REFERENCES

1. Avrin DE, Ramaswamy M, Andriole KP, et al: Preclinical testing of a cost effective digital modality diagnostic workstation. Presented at SCAR 94, Winston-Salem, NC, June 12-14, 1994
2. Ramaswamy M, Avrin DE, Andriole KP, et al: Use of personal computer technology in supporting a radiological review workstation. SPIE Medical Imaging Proceedings 2165: 27-37, 1994
3. Petzold C: Programming Windows 95. Redmond, WA, Microsoft Press, 1996
4. Bonner P: Network Programming with Windows Sockets. Upper Saddle River, NJ, Prentice Hall, 1995
5. Avrin DE, Andriole KP, Arenson RL: Multi-tasking network communications and user interface in a Macintosh diagnostic workstation. Presented at SCAR 96, Denver, CO, June 6-9, 1996
6. Gillespy T III, Rowberg AH: Radiological images on personal computers: Introduction and fundamental principles of digital images. J Digit Imaging 6:81-87, 1993