A Real-time Upcall Facility for Protocol Processing with QoS Guarantees

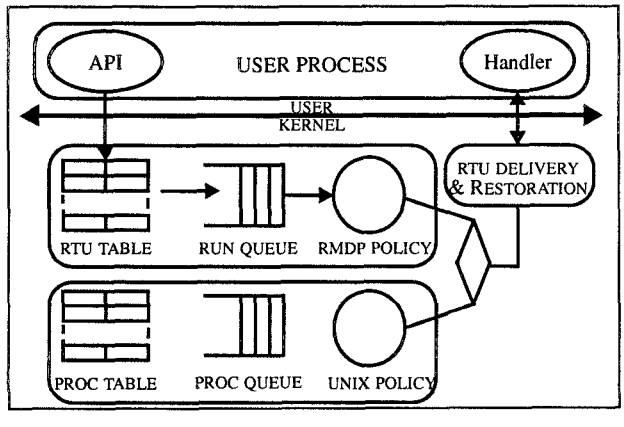
R. Gopalakrishnan
gopal@dworkin.wustl.edu

Guru M. Parulkar
guru@flora.wustl.edu

Department of Computer Science
Washington University, St. Louis, MO 63130

1. Introduction. There is a growing need to provide guarantees for protocol processing within the host operating system to support multimedia applications that have quality of service (QoS) requirements. The QoS parameters for a multimedia stream translate to protocol processing requirements that can be expressed in terms of a time period T, and the number of protocol data units (PDUs) B_p in a batch to be processed per period[1]. This report is an extended abstract of the work in [4] that details the design and implementation of a real-time upcall (RTU) facility to support protocol implementations that require periodic processing with guarantees. RTUs are an alternative to real-time periodic threads and have advantages such as low implementation complexity, portability, and efficiency. The RTU mechanism is an enhancement to the upcall mechanism that has been used to structure layered protocol code. RTUs are invoked periodically in real-time in a user process and implement PDU processing code[2]. The RTU mechanism can be easily extended to protocols implemented in the kernel. The RTU facility uses a scheduling policy called rate monotonic with delayed pre-emption (RMDP)[3]. Other types of media and bulk data processing with QoS can also benefit from the RTU facility since it falls into the periodic processing model[1]. We describe the RTU implementation below.

2. RTU Implementation. An RTU is created by a process as part of connection setup and has attributes that include a procedure (or handler) in the user program, values of T and B_p, and a data structure called a protocol control block (PCB). The RTU facility is layered on top of the UNIX scheduler as shown in the figure. The UNIX scheduler itself is unchanged–the RMDP policy overrides the decision of the UNIX scheduler based on the status of the RTU run queue.



The RMDP policy gets control whenever an RTU becomes runnable in the clock interrupt. It then switches in the correct process, and upcalls the RTU handler with its PCB as argument. The PCB serves to isolate RTUs in the same process and encapsulates the long term state of the protocol session. When the handler returns either due to a yield request or on completion, the kernel restores the system state so that normal processing can resume. A system call is provided to allow a process to synchronously wait for RTU invocations. We briefly outline the RMDP scheme in Section 3.

3. The RMDP Scheme. RMDP exploits the iterative nature of protocol processing to increase scheduling efficiency. The key component of the RMDP scheme is the use of a shared memory data structure between each RTU and the kernel scheduler. When a higher priority RTU becomes runnable, the scheduler does not pre-empt the running handler. Instead it sets the *yield_request* flag in the shared data structure. The running handler checks this flag after completing each iteration and returns if it is set. If at this time the handler has not processed all of its B_p PDUs, the remaining number (stored in shared memory) get processed when the RTU is resumed. To account for the pre-emption delay, we have modified the existing schedulability test for the RM policy that assumes instantaneous pre-emption.

The obvious advantage of RMDP is that the number of context switches are reduced because at least one iteration is completed every time a handler is invoked. It also avoids the need to preserve the activation record and register context of a handler across pre-emptions. Another advantage due to the non-preemptive nature of RMDP scheduling is that shared variables need not be locked before access. This saves on system calls to manipulate locks and reduces handler code complexity.

4. Experimental Results. The first implementation of the RTU facility was on a 25 Mhz Sparc-2 platform. We were able to port it to a 100 Mhz Pentium platform in a matter of a few days. Our experimental results show that RMDP reduces the number of context switches by 35% giving a 10% increase in useful utilization when the total RTU utilization is 70%.

5. References

[1] Gopalakrishnan, R., Parulkar, G.M., "A Framework for QoS Guarantees for Multimedia Applications within an Endsystem," Swiss German Computer Science Society Conf., 1995.

[2] Gopalakrishnan, R., Parulkar, G.M., "Application Level Protocol Implementations to Provide QoS Guarantees at Endsystems," *Proc. of the Ninth IEEE Workshop on Computer Communications,* Duck Key, 1994.

[3] Gopalakrishnan, R., Parulkar, G.M., "RMDP-A Real-time CPU scheduling Algorithm to Provide QoS Guarantees for Protocol Processing," *IEEE Real-time Technology and Applications Symposium,* (Poster), May 1995.

[4] Gopalakrishnan, R., Parulkar, G.M., "Real time Upcalls," Tech. Rep. WUCS-95-06, Washington Univ. Mar 95.

SIGOPS '95 12/95 CO, USA
© 1995 ACM 0-89791-715-4/95/0012...$3.50