A Micro-kernel for Isochronous Video-Data Transfer

Masaaki Iwasaki, Tadashi Takeuchi, Masahiko Nakahara, Shouji Nakamura, Takahiro Nakano, Kazuyoshi Serizawa, and Shihoko Taguchi
Systems Development Laboratory, Hitachi, Ltd.

Abstract. We developed a new micro-kernel (Tactix) that provides isochronous thread scheduling. Isochronous scheduling is designed to achieve cyclic executions of multiple threads with extremely low jitters (less than 1 msec). Furthermore, we made an experimental video-server application on top of Tactix and evaluated its performance and service quality.

The experimental VOD (video on demand) system consists of four client PCs and a video-server, and they are connected by a 10 Mbps Ethernet on which both the MPEG1 down-streams and command up-streams are transferred. Results show that the system achieves a packet loss ratio of less than 0.02 percent, without retransmission, when packet transmissions of the down-streams are controlled by the isochronous scheduler.

1 Introduction

To achieve smooth video and audio playback on a computer system, an operating system must provide accurate periodic scheduling of real-time threads (cyclic threads). Many scheduling algorithms have been proposed for this purpose; such as, the rate-monotonic scheduling [1], the deadline first algorithm [2], the priority-ceiling protocol [3], the deferrable server algorithm [4] or the processor capacity reservation [5][6][7]. However, actual implementations of these algorithms increase scheduling costs and they can not solve the priority inversion problem completely. Thus, in a multiple-stream environment under a heavy processor load, it is difficult to keep the deadline miss ratio low.

For example, if a software MPEG decoder and a 100 Mbps Ethernet driver are executed on a processor, it becomes difficult to keep the service qualities of both tasks high. The scheduling delay of the 100 Mbps Ethernet driver receiving the successive packets will induce packet loss due to receive buffer overflow if the software MPEG decoder is executed at the highest priority. This will be a fatal problem for the compressed video data playback. On the other hand, if the 100 Mbps Ethernet driver is executed at the highest priority, the software MPEG decoder will experience frequent deadline misses.

We are developing new micro-kernel Tactix from scratch to solve these problems. This paper presents details of Tactix's isochronous scheduling, and related kernel technologies that enable accurate cyclic executions of multiple real-time threads, and the evaluation results of a video-on-demand server on Tactix.
2 Isochronous Scheduling

The isochronous scheduler employs the following four schemes to achieve high-quality cyclic executions of multiple threads that are suitable for continuous media applications.

1. Static processing power reservation that satisfies the time constraints of all cyclic threads.
2. A thread switch synchronized with the timer interrupt according to the static processing power reservation information in (1) above.
3. Multiple-layered interrupt handlers that are separated into a higher priority part and a lower priority part.
4. Fine-granule preemption control that enables mutual exclusions of kernel resources without a lock mechanism.

In this section, we focus on these topics except fine-granule preemption that is described in Sect. 3.

2.1 Timer-Synchronized Scheduling

Before each cyclic thread begins its execution, it declares the time-constraint information: the execution period and the reserved execution time in each period. The isochronous scheduler creates a time-slot table based on this time-constraint information. The time-slot table is a time-reservation table where the time sequence is divided into time slots as shown in Fig. 1. The length of each time slot is equal to the timer interrupt interval. Each time slot is assigned to a cyclic thread that can be executed in the interval.

The scheduler is triggered by every timer interrupt, and it selects and dispatches the cyclic thread that must be executed in the next time slot at the highest priority. There is only one cyclic thread that can be in a raised state in each time slot. The other cyclic threads, except this thread in the raised state, are kept in the depressed state. The raised state thread is executed at the highest priority except interrupt handlers. The depressed state threads are never scheduled until their allocated time-slot arrives.

2.2 Time-Slot Allocation

The time-slot table is dynamically rebuilt when a cyclic thread is added to or deleted from the system. Each cyclic thread can specify the number of time slots as its execution period. To avoid deadline misses, it is better to keep an adequate margin when specifying the reserved execution time in a period. If the reserved execution time in a period is not consumed completely, the remained execution time can be freed. This extra processing power is used for executing non-cyclic threads.

A time slot (slots) for a thread with the shortest period is (are) allocated first as shown in Fig. 1. If the required execution period is too long and it is not possible to allocate successive time slots, the discrete time slots are allocated.