ABSTRACT
This paper describes the distributed real-time monitoring tool JewelNT that has been developed and first applied in the context of the EIVIS video server, a European ESPRIT Project. The paper motivates the need for a fine-grained, trace-based monitoring tool in order to support development of soft real-time applications on Windows NT, like the EIVIS interactive Video-on-Demand server. It also sketches the design of JewelNT that hooks in the NT kernel and provides full information about NT’s thread-scheduling at system and application-level.

1 Introduction
The European ESPRIT-project EIVIS currently develops an "Embedded Interactive Video Server", a video-on-demand server for aircrafts and trains. The server architecture is described in more detail in section 3. The server is designed to run on Windows NT as the operating system. Windows NT has been chosen because of its availability and its excellent development support at a reasonable price. Obviously, streaming video-data imposes real-time requirements on the underlying system. Although video-on-demand is often considered as a pure soft real-time application, it becomes "harder" in an aircraft where the carriers have special QoS requirements on in-flight entertainment systems, e.g. video must not flicker because passengers otherwise might fear a sever damage of the aircraft.

As stated in [MS95] „Microsoft® Windows NT™ Workstation is not a hard real-time operating system. Rather, it is a general-purpose operating system that has the capability to provide very fast response times, but is not as deterministic as a hard real-time system". But still it comes with many valuable mechanisms for soft real-time applications, like e.g. multimedia applications or on-line stock-trading. It provides preemptive multi-threading with special real-time priorities, a memory-management that offers page-locking, and a kernel that is optimized for minimizing interrupt latency by the use of deferred procedure calls [Cus93].

But sometimes the design of Windows NT (that is optimized for responsiveness in user-interaction) still makes it difficult to understand exactly how and especially when things are happening. NT, unlike many dedicated hard real-time kernels, has no means to cope e.g. with priority inheritance, as it is not relevant in normal time-sharing operation. Therefore, it becomes necessary for the developer of a real-time application to get to know of these problems and to work around them.

In general, often during the development of a real-time application it highly desirable to get deep insight into the dynamics of the operating system and its scheduling decisions in order to understand timing and synchronization problems that cause anomalies in the application’s behavior. The standard performance tools provided by NT are not sufficient in these cases. They usually rely on the performance counter API that is designed to provide average values but no information on individual events. Detailed traces of system related events with accurate time-stamps are more appropriate to understand e.g. why the synchronization of two threads fails in certain cases. In the context of the European ESPRIT-project EIVIS (Embedded Interactive Video Server, a video-on-demand server for aircrafts and trains) GMD has developed the JewelNT tool that addresses this lack of real-time tools.

2 The JewelNT Tool
The idea of a monitoring tool that can visualize real-time applications and the scheduling decisions of a real-time operating system is not new in itself. The ARTS kernel implemented such a tool in 1988 [Tok88] and today the TORNADO development environment for the real-time kernel vxWorks comes with the WindView tool (http://www.wrs.com/windview.html) that provides a similar functionality for stand-alone system. What is novel in JewelNT’s approach is the fact that it is applied to the NT kernel that is not designed to support such a
tool and that it is integrated in the larger JEWEL monitoring environment. JEWEL [LKG92, GLK92] provides the infrastructure to observe a distributed system under test and to present the collected data on-line on a centralized experiment control workplace. In contrast to existing monitoring tools JewelNT allows to observe the activity of a number of distributed machines simultaneously and to relate events between the interacting machines. This feature is important for the evaluation of the EIVIS server as its functionality of delivering striped video contents relies on the timely coupled interaction of different nodes.

The JewelNT tool monitors NT threads. It extracts on-line detailed traces of kernel-level events (thread switches, interrupts, and user-defined events in application specific drivers) combined with user-defined application events in user-level code. It allows analyzing the event traces on a remote machine with a graphical presentation interface. For generating the kernel-level events JewelNT hooks into the Windows NT kernel with its own driver that intercepts the according kernel routines.

JewelNT allows instrumenting the application’s (C++) code with arbitrary user-events. During monitoring an experimenter can interactively select the set of events (and even the application objects) he/she is interested in. Also the graphical display can be customized on-line. With the addition of the kernel-level events by JewelNT it can provided a view on the system that combines application semantics and information on its implementation by the system. A typical observation that can be made with JewelNT would be e.g. the overall time of an application’s I/O operation and its breakdown into application specific activities (e.g. class-library calls), kernel activities (Win32 and system server threads), interrupt processing, waiting time, and pre-emption by other threads.

![Figure 1: Components of the JewelNT tool](image)

JewelNT consists mainly of three components (depicted in figure 1): a Windows NT kernel-level driver, an ONC RPC-based remote communication infrastructure, and a graphical user interface. The kernel driver instruments the standard NT kernel in order to intercept all thread switches and interrupt handlers. It generates event-descriptions about these system events and stores them with a high-accuracy time-stamp (either provided by the PerformanceCounter API (resolution 0.8 µs) or by the Pentium internal tick-counter (resolution < 10 ns)) in a global shared buffer. Event generation itself introduces an overhead of about 17 µs/event (on a 90 MHz Pentium machine). By calling a library provided with JewelNT a developer of kernel- or user-mode NT real-time software can also instrument his/her own software in order to add also user-defined events into the same event-buffer. The remote communication infrastructure (in the figure the „external sensor“ component) allows to collect the local event-streams from multiple nodes of a distributed system and to forward them on-line to one experiment control node. On this node the experimenter can manage the experiments remotely only using one graphical user interface. This interface allows selecting the interesting events, to start and stop measurements on a selected set of nodes, to initiate on-line data transfer to the presentation node and to analyze the measured data in detail. The analysis view displays either as text or as graphical display (shown in figure 2) with variable zooming facility all thread switches, interrupt and user-defined events with their exact timing. This is a very fine-grained view of the real-time behavior but still very intuitive for a software developer.

![Figure 2: Graphical display of the measured task-switches and interrupts on one Windows NT node](image)

3 Measurements in the EIVIS Server

JewelNT has been first applied to the prototype of the EIVIS video-on-demand server, an embedded video server for aircrafts and trains. The main challenge for any video-on-demand server is the enormous bandwidth of random access video data that is required to provide any client a full VCR-like high-quality video service [LOP 94, Bol96]. To achieve this guaranteed bandwidth with the EIVIS video server the video-data is striped across a cluster of disks connected to a cluster of nodes...
of nodes connected via a high-speed interconnection network. The general architecture is depicted in figure 3. It shows the disk server nodes (called "data pumps") with the connected disks and the "Cabine Network Interface"-node responsible for re-assembly of the scattered video stripes from the data pumps and for injection of the video data into the cabin network. In the first phase of the project one server installation consists of at least four Pentium-based Windows NT 3.51 nodes, connected by a proprietary high-speed interconnect, the HyperNet, developed by our EIVIS partner Hyperparallel Technologies Inc. (HPT), France. The application has typical soft real-time constraints as it has to server more than 60 MPEG1 streams with a guaranteed data rate of 1.5 Mb/s. As in most video-servers the technical challenge is the re-assembly of the scattered video stripes from the disks on all nodes into one output-stream. In the first EIVIS prototype this de-multiplexing is done via the Hypernet that currently transfers packets at 30 Mb/s per node link. In order to optimize server performance and to minimize the required buffer-space the synchronization and the scheduling of the involved server threads has to be fine-tuned. In order to achieve that, measurements with JewelNT were recently started to understand NT’s scheduling of these tasks and to measure the influence of the Hypernet controller interrupts on the overall performance.

Figure 3: General architecture of the EIVIS video server

4 Summary and Outlook

While NT is no dedicated real-time operating system it still is an attractive basis for soft real-time applications. Due to the complexity of NT we stated a need for a fine-grained trace-based monitoring tool in order to support development of soft real-time applications. The paper sketched the design of JewelNT, a tool that hooks in the NT kernel and provides full information about NT’s scheduling combined with application-level timing information. We already adapted the tool to symmetrical multiprocessor machines with NT, where synchronization problems are even more an issue than on single processor. JewelNT is currently running on Windows NT 3.51 and 4.0 on Intel platforms (with the graphical interface also available on Solaris). A beta-version is available for evaluation from GMD. For the details and the latest developments please also have a look at the JewelNT Web-page http://set.gmd.de/~mfg/JewelNT/.

In the next future we will integrate this tool with our automatic instrumentation tool for C++ [Ger94] with the final goal to get an tool that can provide precise application specific timing information from the measurement of a real-time program running on NT without further user intervention.

5 References