Adaptive Resource Management in the ACTORS Framework –
A live DVB-T/Webcam Demo

Stefan Schorr, Anand Kotra, Gerhard Fohler
University of Kaiserslautern, Germany
{stschorr,meher,fohler}@eit.uni-kl.de

Johan Eker
Ericsson Research, Sweden
johan.eker@ericsson.com

Karl-Erik Årzen, Vanessa Romero
Lund University, Sweden
{karlerik, vanessa} @control.lth.se

Abstract—In this demonstration, we show the ACTORS framework managing multimedia video streaming to a client that runs an adaptive video player and has scarce and fluctuating resource availability. We employ only minimal buffering and perform the stream adaptation at run-time. As sources of the live MPEG-2/MPEG-4 SP video streams we use a DVB-T stick and an Axis webcam, respectively.

I. INTRODUCTION

Many applications have highly fluctuating and data-dependent resource demands. These applications compete for the available resources to provide services. Each service has a QoS and all services together contribute to the global QoS of the system. Static solutions to allocate resources to the applications based on worst case assumptions provide for highest QoS of individual applications at the expense of wasting resources most of the time. Static relaxation of these constraints provides for increased resource utilization at the expense of unnecessary QoS degradation under sufficient resource availability. Adaptive real-time systems adjust internal strategies upon scenario changes to provide for higher resource utilization without causing unnecessary QoS degradation.

Current RT embedded systems have very little capabilities for runtime adaptation, e.g. they lack explicit support for QoS management. A major challenge is to have a generic notion of QoS that is capable of expressing application level requirements to the underlying system. Another challenge is the increasing complexity of systems that requires reservation based resource management for ease of composability of subsystems [1]. With the shift towards multi-core systems, the need for resource management increases: Obtaining the full performance of these systems requires the efficient exploitation of parallelism and smart mapping of processes to the available computational resources.

In this demonstration, resource demands of video playback are subject to heavy fluctuations over time and are also data-dependent. Multimedia applications can adapt to the system’s varying resource availability by offering multiple service levels for playback with different properties such as e.g. frame rate, or quality of the displayed stream. These adaptive multimedia applications have different resource requirements and provide different QoS to the user depending on the chosen service level. Using an adaptive resource management framework increases the flexibility, maximizes the global QoS, and avoids wasting resources.

II. THE ACTORS FRAMEWORK

The ACTORS framework [2] automatizes the distribution of available resources to applications at run-time, based on application’s demands and the overall QoS of the system. The structure of the framework is as follows: At the very bottom, there is a physical platform which is capable of enforcing resource reservations for processes, the so called virtual processors (VPs): A virtual processor is a reservation of processor bandwidth that can be seen as an abstraction of a physical core. All tasks executing in a such a reservation are protected by temporal isolation against the influence of other tasks. Applications are split to a set of VPs, each executing a portion of an application. This way, resource allocation is performed independently of a particular physical platform.

There is a Resource Manager (RM) which makes the decisions about allocation of resources to applications according to their importances and requirements. The RM’s decision making is based on the proposition that frequent reactions to fluctuations of computational demands and resource availability would be too inefficient. Rather, resource management in ACTORS is inspired by the MATRIX resource management framework [3] where application demands are abstracted as a small set of service levels, each characterized by a QoS and resource requirements. The ACTORS RM ensures that only significant changes trigger a system reconfiguration. Furthermore, applications provide additional feedback to the RM – the so called happiness. The happiness value specifies whether the resources assigned to an application suffice to provide the QoS associated with the selected service level. The RM takes the happiness value into account to fine tune the resource allocation of the competing applications at run-time.

Another important characteristic of the ACTORS framework is its modularity: Components like, e.g. the underlying operating system that enforces the reservations or the logic of the RM, can easily be changed.

III. SETUP OF THE DEMONSTRATION

In the demonstration we show the ACTORS framework managing multimedia video streaming to a client that runs an adaptive video player and has scarce and...
fluctuating resource availability. We use an implementation of the ACTORS framework on a standard Linux kernel which utilizes a resource reservation mechanism called SCHED_EDF [2]. SCHED_EDF implements a hard reservation mechanism based on the Constant Bandwidth Server algorithm to guarantee a fraction of the CPU bandwidth to VPs.

The setup consists of two dual-core laptops connected over a wireless network, one acting as the video server, the other as the client. The server has a DVB-T stick as well as an Axis webcam attached to either provide an MPEG-2 or an MPEG-4 SP stream, respectively. In the case that there is no DVB-T signal, we will use a recorded DVB-T stream from the hard drive.

The client laptop contains the ACTORS framework and two adaptive video player applications: an adaptive MPEG-2 decoder that is based on a standard VLC player and an adaptive MPEG-4 SP decoder that has been implemented in the CAL dataflow language. Both decoders offer two distinct service levels for playback: The MPEG-2 decoder runs on a single dedicated VP which requires 30% CPU bandwidth in the highest service level and 5% in the lowest. The MPEG-4 SP decoder has been partitioned to two virtual processors both requiring 70% of CPU bandwidth in the highest service level and 50% in the lowest. Both decoders interface with the RM to inform on their resource requirements for the different service levels, and with the video server to adapt the bandwidth of the outgoing stream to match the decoders computational resources. In the case of the MPEG-2 stream, the server side adaptation consists of skipping of frames and DCT coefficients, while in MPEG-4 SP only DCT coefficients are skipped [4]. Additionally, the client runs a GUI that monitors at run-time the service level table, the resource allocation, and a history of the resource utilization.

IV. THE DEMONSTRATION

In the first part of the demonstration, the DVB-T stick provides an MPEG-2 stream which is processed by the server and then streamed to the client. Since there is only one application running on the client, the adaptive video player runs at the highest service level. Then we start a new application with higher importance than the player. This application consists of two VPs and only offers a single service level with a bandwidth requirement of 80% for each VP. In order to accommodate both applications, the RM reassigns resources to constrain the player’s VP to 5% bandwidth and tells the video player to operate on the lowest service level. The player then informs the server about this change and the server adapts the outgoing MPEG-2 stream. This results in a reduced but still acceptable quality of the video playback. When the second application terminates, the RM reassigns the resources and service level like in the beginning of this demonstration. The player informs the server about the change which finally results in highest quality video playback.

In the second part of the demonstration, the Axis webcam provides an MPEG-4 SP stream that the server processes and streams to the client. We again start with only the CAL player running and the RM tells it to operate on highest service level. Then we start up a more important CAL application which consists of two VPs and consumes approximately 40% of CPU bandwidth. The RM now reassigns the resources of the CAL player to obtain enough free resources to execute both applications and tells the CAL player to switch to the lowest service level. The result is that the CAL player gets sufficient resources so that video playback can continue at the cost of lower perceived video quality. When the second application terminates, the RM reassigns the resources and service level like in the beginning of this demonstration. Additionally, the CAL player informs its server about the change which finally results in highest quality video playback.

V. ACKNOWLEDGEMENTS

The authors would like to express their gratitude to Raphael Guerra for his support by which this work has benefited tremendously.

REFERENCES


