Towards Adaptive Resource Reservations for Component-Based Distributed Real-Time Systems

Nima Khalilzad[1], Mohammad Ashjaei[1], Luis Almeida[1,2], Moris Behnam[1], Thomas Nolte[1]
[1] MRTC/Mälardalen University, Sweden
[2] IT/DEEC/University of Porto, Portugal
nima.m.khalilzad@mdh.se

***Abstract*—In this paper we present our ongoing work on developing a framework supporting adaptive resource reservations targeting component-based distributed real-time systems. The components may be spread over different resources in a distributed system. The proposed framework utilizes a reservation-based scheduling technique in which the sizes of reservations are adjusted during run-time to deal with dynamic resource demands of the software components. We present our modeling approach, we describe design options made and we present corresponding challenges.**

I. Introduction

Complexity in the embedded software domain has been growing rapidly. Component-Based Software Engineering (CBSE) provides a modular approach for developing complex software systems. CBSE allows parallel development of a complex system by different development teams. In this approach, each team is responsible to develop a particular component. The components are integrated at the final stage of the development. In the context of real-time systems, resource reservation techniques have been widely used for developing component-based real-time systems [1], [2], [3]. In such an approach the capacity of a resource is divided into a number of reservations, and each reservation is assigned to a component. In doing so, the timing properties of the component can be studied in isolation. The integration techniques guarantee that the timing properties will be preserved after the integration.

In the context of hard real-time systems, the size of resource reservations is calculated at design time and kept fixed throughout the life time of the components. In the soft real-time systems domain, however, the reservation sizes may be adjusted during run-time to deal with the dynamic resource requirements of the components. Adaptive reservations have been studied for single resource systems (e.g. [4], [5]). However, in the context of distributed systems, adaptive reservations have not received much attention. In the distributed system context, the processor resources as well as the network resources have to be considered in an integrated fashion.

In this paper, we consider component models in which a component may be composed of multiple tasks spread over a distributed system. The tasks within the components may communicate through the network using messages. Therefore, the components, besides the tasks, may contain messages. We focus on dynamic resource reservations on the processor resources as well as the network resources. We also discuss the challenges associated with adaptation of the reservation of such resources for the components, and we propose a solution sketch.

The research leading to this paper has received funding from the Swedish Research Council (Vetenskapsrådet) under the project ARROWS and the Swedish foundation for strategic research (SSF) via the research project PRESS.

II. Related work

In the following, we review the reservation techniques inherent in three different areas, processor resources, network resources, and end-to-end resources in distributed systems. We also review two groups of works: (i) static reservations, and (ii) dynamic reservations. From a modeling perspective, several resource models have been proposed for modeling resource reservation techniques. For instance, the Periodic Resource (PR) model [1] uses a period and a budget for characterizing a reservation. A reservation is guaranteed to receive a specific budget during each time interval equal to the period. The budget is reduced while the resource is assigned to a particular component, and it is replenished at the start of the period.

Resource reservations on processors. A number of resource reservation models are realized on the processor resources. For instance, the Constant Bandwidth Servers (CBS) [6] are implemented in the Linux kernel [7], or the PR model is implemented in VxWorks [8]. When the tasks have dynamic resource demands, it is desirable to adapt the reservation parameters to deal with the resource demand changes. Adaptive CBS is promoted in the AQuoSA project [4] for dynamic tasks such as video decoders. The ACTORS projects [9] uses adaptive CBS on multiprocessor platforms. In [5], the budget of periodic servers are adapted tracking the processor demand of the components. In this work the model is hierarchical, i.e., the periodic servers may contain multiple tasks as well as multiple child periodic servers.

Resource reservations on network. The same modeling concepts as in processors have been applied on the network resources. A general category of the resource management in network is traffic shapers [10]. The purpose of these shapers is to limit the amount of traffic that a node submits to the network in a given time interval. Similar to the techniques used by processor servers, the traffic shapers use methods based on capacity which is replenished with different policies, e.g. credit-based shaping in Ethernet AVB. Moreover, some real-time Ethernet protocols enforce a cyclic-based transmission and reserve windows for different classes of traffic (e.g., Ethernet POWERLINK [11], FTT-SE [12] and HaRTES [13]). Also, a hierarchical server model [14] is proposed for the Ethernet switches in the context of the FTT-SE protocol to reserve a portion of bandwidth for different traffic types, hence providing temporal isolation among them. An online QoS management [15] is proposed in the context of a multimedia real-time application, which adapts the video compression parameters and the network bandwidth reservations to provide the best possible QoS to the streams. Our end-to-end reservation framework can use the above network technologies for reserving the network resources.

Registering resource reservations on network. In order to reserve resources for streams in the network several protocols have been proposed, where they use similar concepts. For instance, Stream Reservation Protocol (SRP) [16] defines a set of procedures to reserve network resources for the specific traffic streams, which are crossing through an Ethernet Audio Video Bridging (AVB) network. The SRP protocol forces the traffic to be registered on the AVB switches through its path, before being transmitted. Furthermore, a Resource ReSerVation Protocol (RSVP) [17] was proposed to reserve resources for a stream with a specific Quality of Service (QoS) requirement. This protocol operates using an admission control, which checks whether there are enough resources to supply the requested QoS requirement. In both protocols, the mechanism performs by sending a request through the network and checking in each node the availability of resources. In our framework, we use such protocols to communicate the resource adaptations (i.e. new reservations) throughout the distributed system.

Resource reservations in distributed dystems. Few authors have addressed the end-to-end reservation of resources for distributed systems, including processor and network resources. A distributed kernel framework with a resource manager in each node has been designed and implemented to provide an end-to-end timeliness guarantee [18]. Also, a resource management system, called D-RES [19], has been developed to handle shared resources among multiple applications in distributed systems. A very close work related to ours is the one presented in [20] in which a pipeline task is considered. Tasks may use one of the resources available in the system to carry on their computations. Adaptive CBS is used to track the resource demand of the tasks. In addition, a general model, called Q-RAM [21], has been developed to manage the resources shared among multiple applications. The applications in this framework have different operation levels with different qualities depending on the available resources. However, they have to satisfy their needs such as timeliness, reliability and data quality. The model allocates the resources to the applications considering that the overall system utility becomes maximum while the applications meet their minimum needs. This model has been extended in [22] for the systems with rapidly changing resource usage.

The main difference of our work with [20] and [21] is that we consider adaptation for components which may in turn be composed of multiple tasks. In our framework, performing adaptation of a reservation in one resource may imply adapting other reservations in that resource that are coupled with other reservations in other resources, via the distributed components, through the whole system. We also focus on the design of an adaptation protocol that can easily be integrated on top of the well established network technologies discussed in the previous paragraph. In the above outlined proposals the adaptation overhead was not thoroughly addressed. In our work, however, we will address the adaptive end-to-end resource reservation (both on processor and network resources) and we will focus on light-weight protocols in which adaptation can be enforced quickly with low overhead, i.e., with insignificant usage of the resources for adaptation.

III. Model

We assume distributed real-time systems in which n resources are available. We use r_h to denote the h^{th} resource available in the system. A number of real-time components are placed on the distributed system. The components may use a subset of all available resources. We consider two types of resources: (i) network resources; (ii) processor resources. The components use resource reservation polices for accessing the resources.

Component model. We assume that a real-time component $\mathcal{C}_i$ is composed of a set of tasks and messages: $\mathcal{C}_i = \{\tau_1^i, \tau_2^i, \ldots, \mathcal{M}_1^i, \mathcal{M}_2^i, \ldots\}$, where τ_j^i and $\mathcal{M}_k^i$ represent the j^{th} task and the k^{th} message of $\mathcal{C}_i$ respectively. The tasks are the processor consumers, while the messages consume the network resources. $\mathcal{C}_i$ is entitled to a share of the available resources with other components in the system through resource reservation techniques. In other words, fractions of different resources are reserved for each component. We use component interfaces for denoting the specifications of the resource reservations assigned to the components. The interface of $\mathcal{C}_i$ is denoted using: $\Gamma_i =< T_i, \{Q_i^1, Q_i^2, \ldots, Q_i^n\}, \zeta_i >$, where T_i is the period of the resource reservations, Q_i^h denotes the budget of r_h reserved for $\mathcal{C}_i$, and ζ_i represents the relative importance of the component with respect to the other components. The importance parameter is used when a resource is overloaded and some components have to be prioritized for receiving the resource.

Task model. We assume a sporadic task model in which a task τ_j^i is characterized with a minimum inter-arrival time p_j^i and an execution time C_j^i. We assume an implicit deadline task model, i.e., the deadlines of tasks are equal to their periods. We further assume that the tasks may operate in different modes. Each mode has its own execution time and period. We denote the task period and execution time associated with an operation mode m of τ_j^i using $p_{j,m}^i$ and $C_{j,m}^i$.

Message model. We assume a sporadic message model in which the k^{th} message of $\mathcal{C}_i$ is denoted using $\mathcal{M}_k^i$. The messages start from a processor resource, consume a number of network resources and end in a destination processor resource. The messages are produced by a producer task and consumed by a consumer task. The set of all resources consumed by $\mathcal{M}_k^i$ is denoted using $\mathcal{R}_k^i = \{r_s, \ldots, r_d\}$, where r_s and r_d represent resources where the producer task and the consumer task are located respectively. Similar to the task parameters, we use $p_{k,m}^i$ and $C_{k,m}^i$ to denote the minimum inter-arrival and the transmission time of $\mathcal{M}_k^i$ operating in the m^{th} mode.

Adaptation model. Since we assume that the tasks and the messages operate in different modes, the resource portions allocated to the components should be adapted to allow efficient resource utilizations. We intend to adapt the reservation budgets (Q_i^r) and keep the reservation periods (T_i) constant. We assume that a mode change in the producer task of a message leads to a mode change in the message itself as well as a mode change in the consumer task. We further assume that the table of mode associations are given prior to run-time. Therefore, without loss of generality, we assume that if the

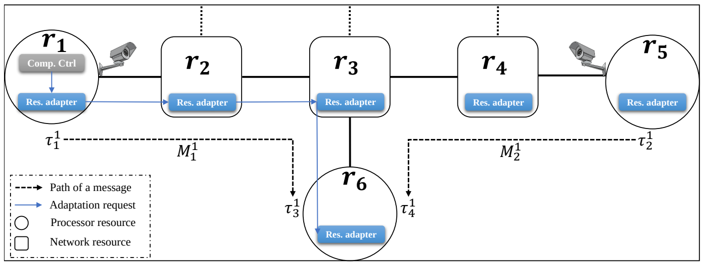

Fig. 1: Surveillance system example. r_1 and r_5 have two cameras attached. r_6 hosts the manager task. Component controller and resource adapter entities are described in Section IV.

producer task is operating in mode m, then the message and the consumer task are also operating in mode m. We perform the adaptations periodically. Therefore, at the time of deciding new budgets, the latest status of the resources and the components is available.

The QoS experienced by a component depends on the budgets allocated to the component. If a component receives the resource portions that it has requested, then the component is fully satisfied. However, components may receive lower budgets due to unavailability of the resources. The components experience partial QoS satisfactions when they receive lower budgets than their requested budgets. The QoS of the whole system, on the other hand, is a weighted aggregate of all components' QoS. When calculating the system QoS, the component importances are used as the weights.

Example. In the following we present an example for elaborating our framework. In our example we assume a distributed system consisting of six resources $\{r_1, \ldots, r_6\}$. r_1, r_5 and r_6 are processor resources while r_2, r_3 and r_4 are network resources. We assume that a surveillance component has been placed on this distributed system. Two cameras are attached to r_1 and r_5. The video frames are preprocessed in their source processors. Thereafter, the video frames are sent to r_6 which hosts the manager tasks. We model this surveillance system as a component placed on the distributed infrastructure: $\mathcal{C}_1 = \{\tau_1^1, \tau_2^1, \tau_3^1, \tau_4^1, \mathcal{M}_1^1, \mathcal{M}_2^1, \}$. τ_1^1 represent the video decoder task placed on r_1, while τ_2^1 is the video decoder task on r_5. Also, τ_3^1 and τ_4^1 are the consumer tasks of the video streams produced by τ_1^1 and τ_2^1. The video decoder tasks generate messages $\mathcal{M}_1^1$ and $\mathcal{M}_2^1$ that consume the following set of resources: $\mathcal{R}_1^1 = \{r_1, r_2, r_3, r_6\}$, $\mathcal{R}_2^1 = \{r_5, r_4, r_3, r_6\}$. The resource reservations of $\mathcal{C}_1$ is represented using the following interface: $\Gamma_1 =< T_1, \{Q_1^1, Q_1^2, Q_1^3, Q_1^4, Q_1^5, Q_1^6, \}, \zeta_1 >$. Note that $\mathcal{C}_1$ may be sharing the resources with other components.

IV. Framework

In this section we provide a high level description of our framework. The objective of our framework is to adapt the component budgets in response to the changes in the resource demands by the components. The adaptation mechanism has to be responsive while imposing insignificant overhead. We first need to sense the demand changes. Thereafter, we have to adapt accordingly. We consider two types of entities for performing the adaptations: (i) *component controllers* which communicate with the producer tasks to sense the resource requirement changes; (ii) *resource adapters* which adapt the reservation budgets responding to the adaptation requests sent by the component controllers.

Component controller. The change in the resource demand is initiated by the producer tasks. The producer tasks inform their corresponding component controller about the change in their operating mode. The component controller, then, makes note of the new resource requirements. The following two problems may be sensed: (i) resource excess, i.e., the resource is over-reserved for the component; (ii) resource deficiency, i.e., the amount of reserved resource for the component is less than the component's required resource. In the case of resource excess, the component controller only sends an adaptation request, if the wasted resource is more than a threshold value. However, for all resource deficiency problems an adaptation request is always sent. The adaptation requests are sent to the corresponding resource adapters. Returning to the example depicted in Figure 1, assume τ_1^1 initiates an adaptation. The controller of $\mathcal{C}_1$, that is placed in r_1 in this example, will send the request to the resource adapters of r_2, r_3 and r_6.

Resource adapter. The resource adapters are responsible to adapt the component budgets based on the requests received from the component controllers. In the case that the resource adapter receives multiple concurrent requests, then it sorts the requests based on the importance of the sender component. The resource adapter, then, executes requests starting from the most important sender. We define the following three responses for a request: (i) accept (ii) reject (iii) force accept. The resource adapter makes decisions based on a simple utilization test. If the sum of requested budgets is less than the schedulability threshold of the resource, then it accepts the request. On the other hand, when the new budgets cannot be allocated due to lack of free resource, then the request is rejected. In this case the sum of the budgets of all more important components than the current component $\mathcal{C}_i$ (i.e., $\forall \mathcal{C}_a \mid \zeta_i < \zeta_a$) and the new request is more than the schedulability threshold of the resource. A reject response will be sent back to the sender node while rolling back all adaptations caused by the request on its path. We also have a case in which the request can be accepted, if and only if, we force less important components to consume less resources. This case is in particular challenging because the changes may need to be propagated throughout the network. This is because if a component does not receive its desired resource portion on one resource due to a force accept response, then it may no longer require the same amount of other resources that are on its path. For instance, in Figure 1, if the component does not receive its desired portion of r_4 due to a force accept response, then it can reduce its required portions of r_3.

Design options. We divide the design problem into two main subproblems corresponding to the design of the component controllers and the resource adapters. In the following we discuss the design alternatives that we will take into consideration.

The component controllers are the generators of the adaptation requests, which are sent to the resource adaptors using request messages. Therefore, the periodicity of the adaptation is governed by the component controllers. The period of the adaptation affects the overhead of the adaptation due to the number of messages that the component controller may send. Thus, assigning an appropriate period to the component controllers becomes an important issue. Also, the threshold for sensing the resource excess plays a role in controlling the number of adaptation requests.

Furthermore, it is desirable to minimize the overhead due to the communications between the tasks and the component controllers. Thus, it is important to place the component controller on a processor that minimizes such overhead costs. For instance in Figure 1, it is better to place the component controller of $\mathcal{C}_1$ either on r_1 or r_5. This is because one of the producer tasks (either τ_1^1 or τ_2^1) will be able to communicate with the component controller without a need to send messages over the network.

The component controllers need to calculate the required budget for each resource based on their internal information about the operating modes of the tasks and messages. Depending on the implementation of the resource reservation, different budget calculation techniques may be used. For instance, we may use the periodic resource model [1]. In such a case, the analysis may also be used for budget calculations. However, such an analysis is pessimistic, and it will result in waste of the resources. It may also be possible to update the budget assignments based on a feedback loop similar to what is proposed in [5].

We have identified three types of messages which will be used for performing the adaptations. (i) Adaptation requests which are sent by the component controllers to the resource adapters communicating the budget changes; (ii) request responses which are sent by the resource adapters to the component controllers informing them about the result of their request; (iii) adaptation requests due to force accepts which are sent by the resource adapters to other affected resource adapters enforcing the new changes. Note that the type (iii) messages need to be sent before the type (ii) messages. This is because the result of type (iii) messages is needed for making decisions on how to respond to the adaptation requests. The type (iii) messages may further result in other type (iii) messages. Hence, it is important to define a stop criterion to avoid jamming the network with the adaptation requests. The proposed method can be applied on the available network resource reservation protocols. For instance, the type (i) and (iii) messages are similar to the advertise messages, and the type (ii) messages are similar to the ready messages used in SRP [16].

The resource adapter may utilize different approaches for deciding the force accept cases. For instance we may guarantee a minimum bandwidth (more than zero) of the resources for a component. In such a model, it will not be allowed to completely shut down a component. Moreover, the resource adapter may either be centralized or distributed (as depicted in Figure 1). In the centralized approach all adaptation requests are directed to a central resource adapter. The adapter, then, decides on how to respond to the requests. The type (iii) messages can be avoided using a centralized resource adapter. This is because this type of messages are sent from an adapter to other adapters which in this case are located on a single node. In the distributed approach, however, each resource has its own adapter. The adapter makes local decisions, i.e., it decides on how to respond to the adaptation requests sent to itself using its local information.

V. Conclusions and future work

In this paper we discussed the design of an adaptive framework for scheduling component-based distributed real-time systems. In the future we would like to develop a simulation tool for investigating different design options discussed in this paper. We will compare the performance of our framework in terms of adaptation overhead and components' QoS satisfactions with respect to the different design options.

References

[1] I. Shin and I. Lee, "Periodic resource model for compositional real-time guarantees," in *RTSS'03*, December 2003, pp. 2–13.

[2] G. Lipari and S. Baruah, "A hierarchical extension to the constant bandwidth server framework," in *RTAS'01*, May 2001, pp. 26–35.

[3] Z. Deng and J. W.-S. Liu, "Scheduling real-time applications in an open environment," in *RTSS'97*, December 1997, pp. 308–319.

[4] L. Palopoli, T. Cucinotta, L. Marzario, and G. Lipari, "AQuoSA-adaptive quality of service architecture," *Software: Practice and Experience*, vol. 39, no. 1, pp. 1–31, January 2009.

[5] N. Khalilzad, M. Behnam, and T. Nolte, "Multi-level adaptive hierarchical scheduling framework for composing real-time systems," in *RTCSA'13*, August 2013, pp. 320–329.

[6] L. Abeni and G. Buttazzo, "Integrating multimedia applications in hard real-time systems," in *RTSS'98*, December 1998, pp. 4–13.

[7] D. Faggioli, M. Trimarchi, F. Checconi, M. Bertogna, and A. Mancina, "An implementation of the earliest deadline first algorithm in Linux," in *SAC'09*, March 2009, pp. 1984–1989.

[8] M. Behnam, T. Nolte, I. Shin, M. Åsberg, and R. J. Bril, "Towards hierarchical scheduling on top of VxWorks," in *OSPERT'08*, July 2008, pp. 63–72.

[9] E. Bini, G. Buttazzo, J. Eker, S. Schorr, R. Guerra, G. Fohler, K.-E. Årzen, V. Romero, and C. Scordino, "Resource management on multicore systems: The ACTORS approach," *Micro, IEEE*, vol. 31, no. 3, pp. 72–81, May-June 2011.

[10] J. Loeser and H. Haertig, "Low-latency hard real-time communication over switched ethernet," in *ECRTS'04*, June 2004.

[11] *EPSG Draft Standard 301 Ethernet POWERLINK Communication Profile Specification Version 1.2.0*, Ethernet POWERLINK Standardisation Group, 2013.

[12] M. Ashjaei, M. Behnam, L. Almeida, and T. Nolte, "Performance analysis of master-slave multi-hop switched ethernet networks," in *SIES'13*, June 2013.

[13] R. Santos, A. Vieira, P. Pedreiras, A. Oliveira, L. Almeida, R. Marau, and T. Nolte, "Flexible, efficient and robust real-time communication with server-based Ethernet switching," in *WFCS'10*, May 2010.

[14] R. Santos, M. Behnam, T. Nolte, P. Pedreiras, and L. Almeida, "Multi-level hierarchical scheduling in ethernet switches," in *EMSOFT'11*, October 2011.

[15] J. Silvestre-Blanes, L. Almeida, R. Marau, and P. Pedreiras, "Online qos management for multimedia real-time transmission in industrial networks," *IEEE Transaction on Industrial Electronics*, vol. 58, no. 3, Mar 2011.

[16] "IEEE 802.1Qat, draft standard for local and metropolitan area networks virtual bridged local area networks amendment 9: Stream reservation protocol (SRP)."

[17] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "Rsvp: a new resource reservation protocol," *IEEE Communications Magazine*, vol. 40, no. 5, pp. 116–127, May 2002.

[18] K. Lakshmanan and R. Rajkumar, "Distributed resource kernels: Os support for end-to-end resource isolation," in *RTAS'08*, April 2008.

[19] A. Oliveira, A. Azim, S. Fischmeister, R. Marau, and L. Almeida, "D-RES: Correct transitive distributed service sharing," in *ETFA'14*, 2014.

[20] T. Cucinotta and L. Palopoli, "QoS control for pipelines of tasks using multiple resources," *IEEE Transactions on Computers*, vol. 59, no. 3, pp. 416–430, March 2010.

[21] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek, "A resource allocation model for QoS management," in *RTSS'97*, December 1997.

[22] S. Ghosh, J. Hansen, R. Rajkumar, and J. Lehoczky, "Integrated resource management and scheduling with multi-resource constraints," in *RTSS'04*, December 2004.