RESOURCE MANAGEMENT FOR CONSTANT BIT RATE STREAMS IN COMPONENT-BASED DISTRIBUTED SYSTEMS

VOLKER FEIL
University of Stuttgart, Institute of Communication Networks and Computer Engineering,
Pfaffenwaldring 47, 70569 Stuttgart, Germany, feil@ind.uni-stuttgart.de

Abstract

In this paper we study an algorithm of a distributed resource management that is responsible for the admission of constant bit rate (CBR) streams. While a central resource management is aware of all resource states of the distributed system, the distributed managers are not omniscient. The considered management software allows the dynamic deployment of software components to the terminals that can be e. g. real-time capable java platforms. We focus on components which use and provide audio/video services that operate with CBR streams. They are meant to be deployed inside future vehicular on-board distributed systems of the telematics domain at system's run time. The resource management task is to dynamically decide on the admission of the CBR streams depending on the service's importance (priority). We map the admission problem to the well known multi-dimensional knapsack problem in order to get a suitable distributed admission control algorithm. Finally, we show that the distributed algorithm achieves a satisfying quality.

Keywords: *Distributed Real-Time Systems, Optimization, Algorithms, Component Software*

1. Introduction

Recent evolution of information and communications technology increasingly change our everyday life. The vehicular telematics domain is also affected by that tendency. Although the systems of that domain are originally static we want to enable the dynamic interchange of services by a special software management that allows the interaction of dynamic deployable software components. In particular, the management must allocate the resources for the components dynamically. The focus in this paper lies on resource management for constant bit rate (CBR) streams that regards special constraints caused by our component software paradigm as well as by expected evolutions in the vehicular telematics domain.

1.1 Vehicular On-Board Telematics Systems

Several physically distributed systems are even now part of the vehicles. Currently, such distributed systems can be roughly divided in an engine, a comfort, and a telematics domain. Candidates for future telematics bus systems are the powerful IEEE 1394 [9] or the optical MOST system (Media Oriented Systems Transport). They provide handling of audio and video (A/V) streams by allowing the establishment of communication channels with guaranteed bandwidths. There is a strict separation: communication channels of best effort services (e. g. WWW services) are realized by asynchronous bus services. However, communication channels of A/V services with real-time requirements are realized by synchronous bus services with deterministic time behaviour. Here, the term „real-time“ means that deadlines have to be regarded, but to miss them will not cause in a catastrophe. The real-time services operate with CBR streams, e. g. they operate with MPEG-2 CBR data [7] or raw CBR data. Bus systems like IEEE 1394 enable the easy, dynamic, and powerful linkage of a variety of A/V sources like CD players, DVD players, and videocameras with a variety of A/V sinks like displays, and speakers/headphones that can be installed per passenger seat.

In section 2 we show the structure of a possible future vehicular on-board distributed system more precisely.

1.2 Dynamic Deployment of Components

Information technology innovation cycles are mostly shorter than vehicle production cycles. There is much research effort today [8] to enable the interchange of services in originally static distributed systems as found in vehicles. A promising approach is to separate the distributed system into hardware that provides the resources, and software that provides the functionalities. Certainly, this is well known in the office world with its PCs and workstations hardware and applications software. However, in the telematics domain human administration should be reduced to a minimum.

In order to enable a user transparent interchange of software we need a special software management. The purpose is to manage dynamically exchangable software pieces (software components) that interwork together by their specified interfaces. The management must enable the dynamic deployment of components to the seat terminals of the distributed system in a user transparent way. The deployment depends on the service requirements of the passengers who currently use the seat computing environment.

In [2] we introduce an appropriate component software architecture DANA (Distributed Systems Architecture for Networks in Automotive Environments), and describe the management of components in a distributed system. In section 2.3 we present aspects of the architecture that are relevant for this paper. Our work is related to the work of Kon who presents a dynamic deployment management of CORBA components for the internet in [4]. Furthermore, there is a relationship to the work of Bates. In [1] he presents a concept for mobile code of multimedia applications. In opposite to ours, Kon's management architecture does not consider deterministic time behaviour. Furthermore, the focus of Bates' framework lies in a mobility concept for the multimedia applications, but not in the investigation of resource usage during their execution.

1.3 The Optimization Problem

In order to enable the functionalities of the deployed software components, there is the need to allocate resources (e. g. CPU times or bus bandwidths) dynamically. In this paper we consider the admission of CBR streams i. e. exchanged data of interworking components with real-time requirements. In section 3 we show that our admission problem can be mapped to the well known knapsack problem [5]. In section 4 results of our problem solving algorithm are shown and discussed.

1.4 Definiton of the Used Technical Terms

The following technical terms are relevant for this paper:

A **component** is a physical package of software with well-defined and published interfaces [3]. A component is deployable and configurable at system's run time. It uses **services** of other components and it provides services to other components as well as to the user. Therefore, there is an interaction between components and data is exchanged via logical directed **channels** as shown in Figure 1. Sections of logical channels outside the terminal can be mapped to synchronous physical channels of the bus system if there is to satisfy a real-time requirement. Furthermore, the sections of the logical channel inside a terminal are realized by control flows that copy data from the bus buffer to a communication endpoint of a component and vice versa. If necessesary the copying task requires real-time behaviour, too.

2. Future On-Board Telematics Systems

In the telematics domain we recognize an evolution from monolithic stand-alone computers in the past, via interacting end units of a distributed system by which the software and the hardware is not separable nowadays to interacting software components that are dynamic deployable to the end units (terminals) in future.

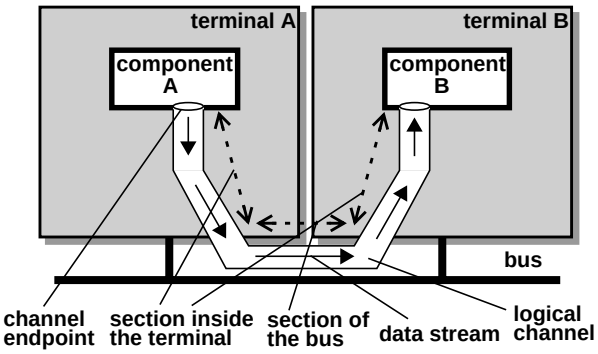

Figure 1: Interaction through a channel

2.1 Generic Component Platforms

Our intention is to enable service access for the vehicular passengers through special terminals per seat. Such terminals are both, equipped with periphery elements like displays (with e. g. MPEG decoders in hardware), headphones, and keyboards, and connected together by a broadband bus like IEEE 1394. The software on the terminals can be configured dynamically by loading software components. Therefore, only the actually needed software is located on the terminal.

The terminals must be generic, so that in general any component can be deployed. They must support best effort as well as real-time requirements of the components. Then it is possible to deploy e. g. a WWW browser component and a real-time audio component together on the same terminal.

A possible representator of such a generic terminal is the component platform realized by Jbed [6]. A Jbed compiler compiles java byte code to object code. Then a linker produces a binary code that is stored in a boot ROM. The possible existence of a compiler and linker inside the platform enables java class loading, and thereby component deployment at run time. Furthermore, beside the usual java priority scheduler there is an earliest deadline first scheduler for real-time java threads. Therefore, the usage of real-time components is possible. Jbed actually realizes an operating system for java programs.

2.2 Operating with CBR Streams

A component that operates with CBR streams can be an A/V source or an A/V sink. The components for a Jbed platform are written completly in java by using hardware-dependend Jbed libraries. For instance an audio sink component must get DMA access in order to control a speaker. There is a logical channel that connects the source and the sink component. It spans over the source terminal, the bus, and the sink terminal. All channel sections require real-time behaviour in order to exchange e. g. a CD audio data stream (1,41 Mbit/s). The channel section that is located within the terminal of the sink component copies

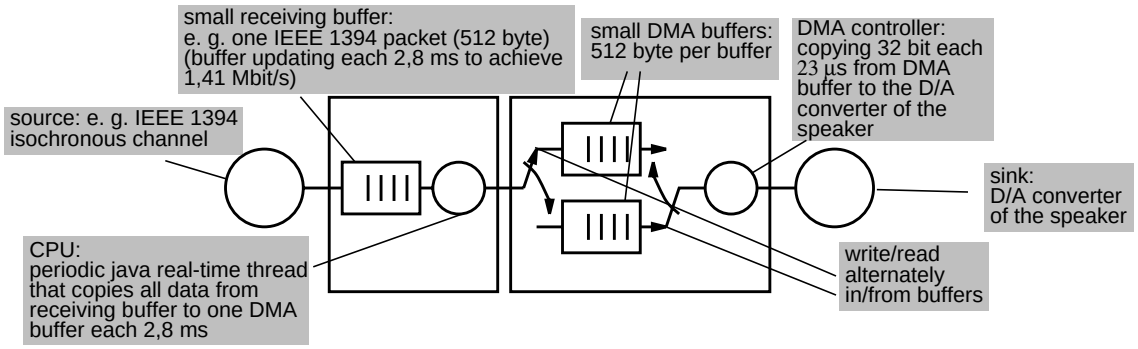

Figure 2: Physical view: Terminals channel section and the belonging sink component

the audio data from a physical isochronous IEEE 1394 channel endpoint to the sink component as depicted in Figure 2. Physically, this endpoint is a DMA buffer. The channel section is realized by a periodic java real-time thread. Finally, the D/A converter gets its data by DMA mechanisms that are managed by the sink component. The location of a component, where it is dynamically deployed, depends on the DMA controller location.

Components that realizes an A/V source or an A/V sink operate with CBR streams as well as with signalling data that allow the control of these CBR streams. For instance, there are start, stop, and pause functionalities that are enabled by further channels for exchanging signalling data among the source and sink components. The components should be kept as simple as possible. This means that complex control functionalities (e. g. controlling several CBR streams) should be splitted in several components. Then, the resource management is able to strike only one goal: It must keep deadlines for the CBR stream that ends on a source/sink component or it tries to minimize the response times of a component that realizes complex control functionalities. While the location of the components that operate with the CBR stream is fix due to the hardware dependencies the resource management can utilize the location of the control component as a degree of freedom in order to achieve the response time minimizing goal.

2.3 Distributed Resource Management for Dynamic Deployable Components

Software management of such a dynamic system has to deal with the following tasks:

Generally, the management has to deploy the components by knowing resource states and component dependencies in order to configure the system. The location is one degree of freedom of the component deployment process, and therefore there is to solve a global scheduling problem. However, in this paper we consider only CBR components with location constraints. Therefore we can neglect that aspect. Furthermore, there are other management tasks we do not consider here. For instance, there is to enable mutual discovery of the components by providing a service discovery mechanism (e. g. naming service).

The management task of our interest is the enabling of the component resource allocation. We introduce DANA in [2], that specifies a distributed resource management of dynamic deployable components. We came to the decision for a distributed and against a central management due to scalability and reliability benefits [2].

The allocation of bus and terminal resources of new inter-component channels is realized by the interaction of a set of DANA managers. The Local Channel Manager (LCM), and the Range Channel Manager (RCM) are responsible for the management of resources needed by channels. On each terminal an LCM is deployed. It appears in two roles – the requesting role and the requested role. The requesting LCM accepts an order to modify (e.g. establish or release) a channel from a component located on its terminal. By usage of a three-way-handshake protocol (exchange of the messages (1), (4), and (5)) the requesting LCM negotiates the channel modification with the requested LCM (see Figure 3). Within this interaction the requested LCM is responsible for the component of its terminal that is affected by the channel modification.

Furthermore, an RCM is involved by exchange of the messages (2) and (3) if the modified channel spans over different terminals. The RCM represents the bus system at the QoS negotiation. After establishing a new data channel, components can use it for data exchange regarding the required QoS. After releasing an existing data channel, the needed resources are deallocated.

3. Resource Management for Channels

The remaining part of the paper compares DANA's distributed resource management with its not omniscient managers with a central management that knows all resource states of the distributed system.

3.1 Resource Conflicts and Priorities

We make that comparison by studying the reactions of resource allocation conflicts. The following example shows their possible occurrence:

An existing CD audio channel with a sink component on a terminal (see also Figure 2) is the example's starting point. We assume that a DAB (digital audio broadcasting)

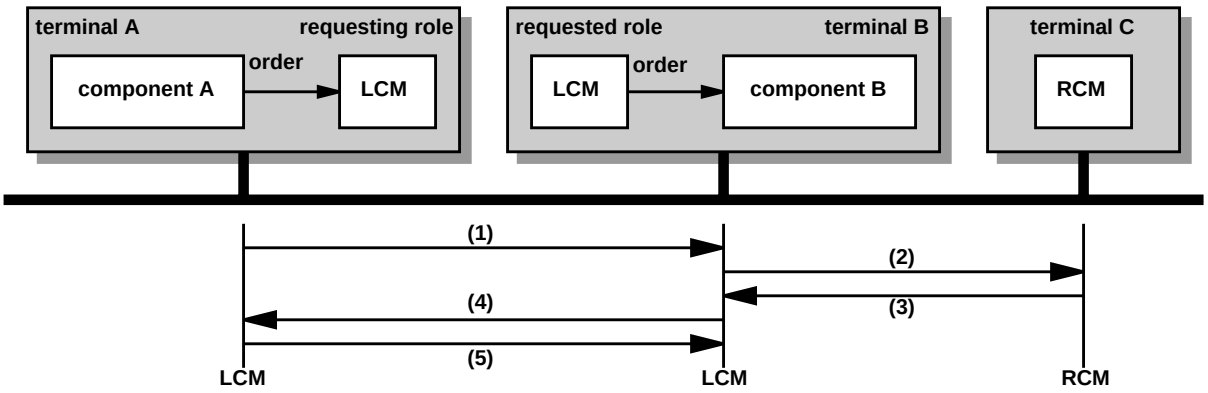

Figure 3: Interaction of the distributed managers

warning service forces the activation of another CBR stream while a warning text is spoken. Let the sink component for both streams be on the same terminal. If there are resource allocation conflicts, the „warning“ channel must get resources from the „CD audio“ channel. The „CD audio“ channel must be released in order to free the DMA resource and eventually the CPU resource. This happens, because the “warning” channel has a higher priority (importance) than the “CD audio” channel.

In general, any service and its enabling channels have a priority. Before a new channel can be established the following questions must be answered for any priority (beginning with the highest and ending with the lowest):

Are there resource allocation conflicts in the channels set with same priorities due to the possible establishment of a new channel with the same or a higher priority? How can these conflicts be solved?

3.2 The Knapsack Problem

The problem to find the optimal subset of channels within a channels set with same priorities can be mapped to a multidimensional knapsack problem. The task is to maximise the objective function

$$\sum_{j=1}^{n} p_j x_j$$

and to satisfy the constraints

$$\sum_{j=1}^{n} w_{ji} x_j \leq c_i$$

The binary variable x_j stands for a channel j, that can be established ($x_j = 1$) or not ($x_j = 0$). Let p_j be the profit of the channel j. For instance, the channel profit may depend on the number of end points (sink components), or on the duration since the channel is established. Furthermore, let w_{ji} be the weight of channel j regarding to a resource i with capacity c_i. Let w_{ji} be w_j for all i, so that channel j utilizes the resource i with a share of w_j/c_i. Here, resources are the required CPU time of the terminals, and the required bandwidth of the busses.

The one-dimensional knapsack problem is a NP-hard linear integer problem [5]. The multi-dimensional knapsack problem is also NP-hard, because with i = 1 it is reduced to the one-dimensional problem. A number of r channels leads to a state room with 2^r states. There are some approximative and heuristic algorithms beside the algorithms that get the exact solution [5].

3.3 Admission Control by Interaction of the Distributed DANA Managers

The distributed DANA managers interact together by the protocols that are depicted in Figure 3. In order to establish a new channel they cannot solve the global channels admission problem as multi-dimensional knapsack problem because each manager has only its view to a limited subset of the global resources: the LCMs are only able to see their CPU resource, and the RCM can see only the bandwidth resource of the bus. They order one another consecutively. By our approach each manager tries to solve its own one-dimensional knapsack problem. Because of the limitation of the number of channels due to the limited terminal resource capacities, the state room is expected to be small enough for computing an optimal solution in acceptable time for the one-dimensional knapsack problem. However if needed, solutions can be computed by approximative or heuristic algorithms. Figure 4 shows three possible strategies for the execution order of the admission control algorithms: terminal-terminal-bus (TTB) strategy, terminal-bus-terminal (TBT) strategy, and bus-terminal-terminal (BTT) strategy.

4. Results

The optimal solution is always achieved by solving the multi-dimensional knapsack problem by a central and omniscient management exactly. This section shows the quality of the solutions of the distributed approach in comparision with that central approach.

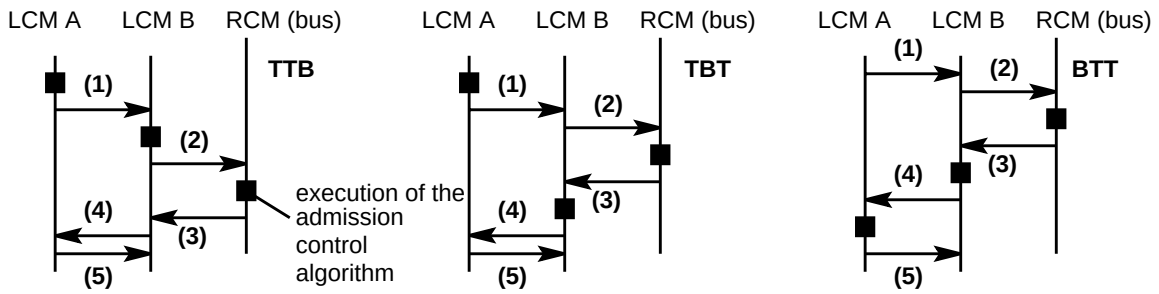

Figure 4: Three posible orders for the executions of the algorithms during the establishment process of a new channel

4.1 The Investigated Scenarios and their Parameters

Scenarios with fully exhausted resources. In these scenarios, all resources are totally allocated by channels with the same priority (channels set Ω). This may be channels of music and video services. Then, an event forces the freeing of resources that are assigned to Ω due to the allocation of resources for a new channel with a higher priority (HP channel). This may be a channel of a warning service. Accordingly, the releasing of a subset of Ω is necessary. The profit p_j and the weight w_j of any channel j are destinated randomly. We assume four seat terminals that are connected by a bus. The resource capacities of the locations are destinated by the channels of Ω that fully allocate all resources. In our scenarios we vary the size of Ω.

Scenarios with terminals as only bottleneck. We expect that in future only the terminals but not the bus system will be the location of resource bottlenecks. For instance, a IEEE 1394 bus will provide a total bandwidth up to 800 Mbit/s. Furthermore, 256 channels with deterministic time behavoiur can exist synchronously. Therefore, we consider also scenarios with totally allocated terminal resources again. However, now we assume an idealized infinite bus resource capacity.

4.2 Comparison between the Strategies for the Distributed Algorithms

In this section we study the managers that are involved by the HP channel establishment process. At any first admission control step an involved manager (LCM or RCM) must select at least one channel of Ω for releasing. The manager tries to select them by achieving its optimal local profit. However, there is the possibility that the decision does not result in the optimal global profit. There is the possibility for false selections that will have no positive effects for further admission control steps. In such cases the release of additional channels in one of the next admission control steps is necessary. The quality of the results of the distributed approach is shown in Figure 5 where the total profit of the modified Ω is compared between the distributed and the central and omniscient algorithm that gets the optimal solution.

Difference between BTT and TTB/TBT for scenarios with totally exhausted resources. For the strategy BTT there is the necessity for the RCM to select at least one channel from Ω for releasing at the first step. A special effect occurs due to the concentrator property of the bus (this means that all considered channels need the bus resource). The selection set of the first step is mostly larger by BTT than by TTB or TBT. Then, the probability to make a false decision is higher.

Equality between TTB and TBT for scenarios with totally exhausted resources. The strategies TTB and TBT lead to the same results. The reason is that the RCM needs never to release channels for both strategies. Therefore, only both involved terminals deals with the releasing of channels in any case. This is to reason by the size of the subset ς of Ω that is composed of channels that are already released during the first step. Because all channels span over the bus (due to its concentrator property) the deallocated resource weight $w_{S,T}$ of the first terminal is equal to the already deallocated resource weight of the bus $w_{S,B}$ for TBT before the RCM admission control is activated (due to $w_{ji} = w_j$ for all i). It can be even less for TTB. The weight $w_{S,T}$ is not less than the weight of the new HP channel w_{HP} because the terminal resources are totally exhausted. The RCM needs never to release channels in order to increase $w_{S,B}$ because $w_{S,B} \geq w_{HP}$ is already valid ($w_{S,B} \geq w_{S,T}$ and $w_{S,T} \geq w_{HP}$).

Equality between TTB, TBT and BTT for scenarios with terminals as only bottleneck. In the paragraph before, we have shown that the RCM does not need to release a further subset of the modified Ω. This is also valid for any strategy for scenarios with infinite bus resource capacity. Therefore, the strategies TTB, TBT and BTT for that scenarios lead to the same results. Furthermore, the results are equal to the results of the strategies TTB and TBT for the scenarios with totally exhausted resources.

Introduction of HTTB. We introduce a heuristic approach, called HTTB, additionally. The LCM that is responsible for the first admission control step knows the second involved LCM (and therefore the terminal) due to its knowledge about the HP channel. Therefore, the selection set can be limited, and the probability of a false selection with no positive effect for the next steps can be decreased. However, it is now possible to release one (or more) channel(s) with high profit(s) due to the limitation of the selection set. Therefore, TTB still works better than

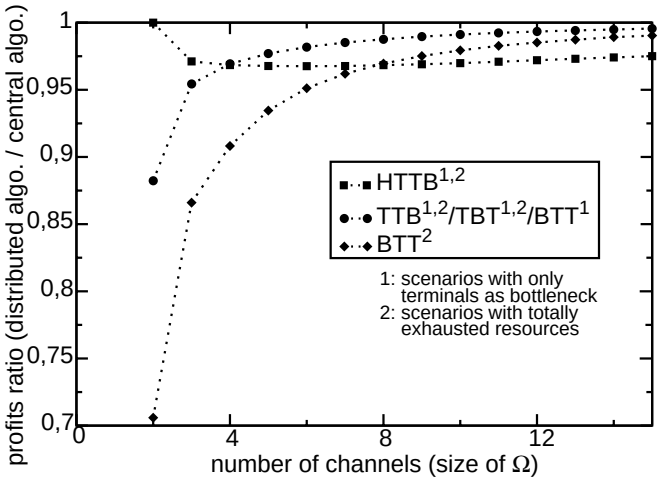

Figure 5: Profits ratio of HTTB, TTB/TBT, and BTT

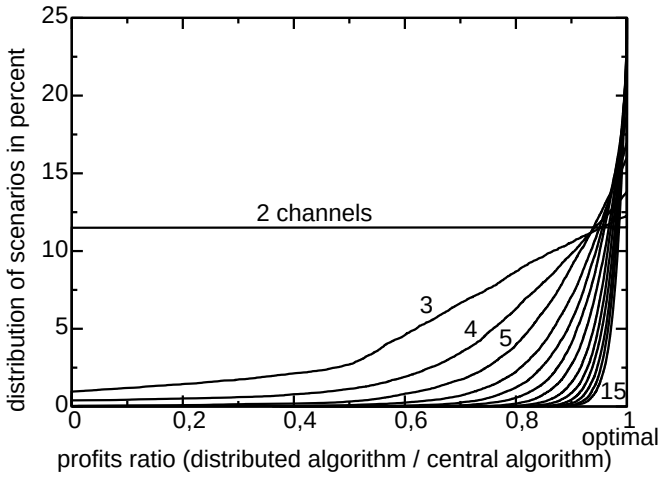

Figure 6: Distribution for TTB, TBT

HTTB even if TTB releases more channels however with less total profit. In general, HTTB works better than TTB only for small Ω (,and it works optimally if there are only 2 channels inside the distributed system).
In Figure 5 a discrete point represents a mean value of a distribution of the randomly generated scenarios. Figure 6 depicts the distributions for the TTB and TBT strategies. The distributions are also valid for the BTT strategy for scenarios with terminals as only bottleneck. The optimal distribution is a step at profits ratio equal 1 from 0 to 100 percent. The curves show a satisfying approximation of that optimal solution.

5. Conclusions

All investigated strategies for the distributed admission control algorithm lead to satisfying results. If there is a small number of channels in the distributed system the heuristic approach HTTB works better. However, the more the number of channels increases the less worthwhile is HTTB.

We expect, that only the terminals but not the bus will be the resource bottleneck in future vehicular on-board distributed systems. However, we cannot generally exclude the possibility that sometimes the bus will become a bottleneck, because the infinity of the bus resource capacity is an idealized assumption. Therefore, in any case TTB is to prefer to BTT and TBT.

The proposed approach can be part of a resource management that enables the dynamic deployment of software components to terminals inside distributed systems. The approach is mainly intresting for the management of CBR streams in future vehicular on-board distributed systems of the telematics domain. In future we want to extend the proposed admission control concept to a comprehensive resource management for dynamic deployable components. The goal is to realize a user transparent interchange of services in the vehicular telematics domain.

Acknowledgement

The author would like to thank Frank Krämer for his work with Jbed.

References

[1] J. Bates, D. Halls and J. Bacon, A Framework to Support Mobile Users of Multimedia Applications. *ACM Mobile Networks and Nomadic Applications*, no. 1, 1996.

[2] V. Feil, U. Gemkow and M. Stümpfle, A Vehicular Software Architecture Enabling Dynamic Alterability of Services Sets, *Proceedings of the 3th International IFIP/GI Working Conference, USM 2000, LNCS 1890*, Munich, Germany, 2000, 68-80.

[3] J. Hopkins, Component Primer, *Communications of the ACM*, vol. 43 no. 10, 2000, 27-30.

[4] F. Kon et al., Dynamic resource management and automatic configuration of distributed component systems, *Proceedings of the 6th USENIX Conference on Object-Oriented Technologies and Systems: COOTS 2001*, San Antonio, USA, 2001.

[5] S. Martello, P. Toth, *Knapsack Problems, Algorithms and Computer Implementations*. (Chichester: John Wiley & Sons Inc., 1990).

[6] M. Pilz, Earliest Deadline First Scheduling for Real-Time Java, *Embedded Systems Conference Europe*, Stuttgart, Germany, 2000.

[7] B. Sostawa et al., DSB-Based Transcoding of Digital Video Signals with MPEG-2 Format, *IEEE Transactions on Consumer Electronics*, vol. 46 no. 2, 2000, 358-362.

[8] M. Stümpfle et al., COSIMA - A Component System Information and Management Architecture, *IEEE Intelligent Vehicles Symposium 2000*, Dearborn, USA, 2000.

[9] IEEE, *Standard for a High Performance Serial Bus*, IEEE Std. 1394-1995, 1996.