# A Planning Method for Component Placement in Smart Item Environments Using Heuristic Search

Jürgen Anke[1,2], Bernhard Wolf[1], Gregor Hackenbroich[1], and Klaus Kabitzsch[2]

[1] SAP Research CEC Dresden, Germany
[2] Dresden University of Technology, Institute of Applied Computer Science, Germany
{juergen.anke, gregor.hackenbroich, b.wolf}@sap.com,
kk10@inf.tu-dresden.de

**Abstract.** Smart item environments consist of networked nodes with heterogeneous hardware equipment and intermittent network connections. Using a common component technology allows for flexible distribution of components for processing of smart item data. Finding a good deployment plan for a new set of components in an infrastructure is called Component Placement Problem. We propose an approach for finding suitable deployment plans for components with special regard to the characteristics of smart item environments. Our method evaluates deployment plans in terms of both resource consumption and availability. From the analysis of the solution space we conclude that the number of network link uses is an important criterion for the quality of a deployment plan regarding both cost and availability. Based on this finding, we have derived a heuristic that creates deployment plans, which have a low number of link uses and are thus more likely of high quality.[1]

## 1 Introduction

Smart items are physical products that include product embedded information devices (PEIDs), e.g. embedded systems, or RFID tags. For application domains such as Product Lifecycle Management (PLM), enterprise applications benefit from accessing data on smart items. Error-prone manual data input can be replaced with automatic data acquisition to support business decisions, e.g. for maintenance planning, effective recycling, and product design improvements. As it is not reasonable to integrate mechanisms for accessing PEIDs into business applications, this functionality is provided by a middleware. The middleware and the PEIDs form the *smart item environment*, which can be distributed in a network over various nodes. A key characteristic of smart item environments is a high degree of heterogeneity in terms of hardware resources. Typically, there is a powerful middleware server which is contacted by client applications to request smart item data. The requests are then forwarded to other nodes in the field, which translate the requests into a PEID specific protocol. Finally, the PEIDs are embedded systems, which contain the data sources and have very limited resources. In general, available resources are decreasing towards the edge of the network.

[1] Parts of this work are based on the PROMISE project (www.promise.no), which is funded by the European Union IST 6th Framework program, project no 507100.

J. Indulska and K. Raymond (Eds.): DAIS 2007, LNCS 4531, pp. 309–322, 2007.
© IFIP International Federation for Information Processing 2007

All nodes in the smart item environment can contain a standardised execution environment, such as OSGi [1] or Jini [2]. This turns the smart item environment into a distributed execution environment, allowing for flexible placement of components that format, analyse, filter, or pre-process the data flows between backend applications and smart items [3]. If a new set of components (a *component composition*) has to be deployed onto the smart item environment, each component has to be assigned to a host. The assignment of a component to a host is called *component placement*, and hence finding a set of good assignments is the *component placement problem* (CPP) [4]. A *deployment plan* is a set of component placements for a given component composition. Previously, we have proposed a method to identify good deployment plans in smart item environments based on the *cost of demanded resources* [5]. The cost of demanded resources is an important evaluation criterion as resources on the various hosts and network links are differently valued in heterogeneous environments.

However, considering the cost of resource demands alone is not sufficient to identify suitable deployment plans. Most products are mobile and therefore communicate with middleware over wireless connections, which influences the *availability* of PEID data. Availability is defined as the degree to which a system or component is operational and accessible when it is required for use [6]. In a distributed component-based system, the availability depends on the placement of components [7, 8, 9] and hence availability is also a relevant evaluation criterion for deployment plans. Intuitively, the availability increases when the amount of data to be transferred over unreliable connections decreases. This can be achieved by placing components on the PEIDs to perform data analysis locally and transmit only the analysis results. However, this competes with the goal of minimising cost of resource demands as resources on the PEIDs are much more expensive than on other nodes. This trade-off has not been investigated in the context of component deployment planning. Instead, existing approaches use a single evaluation criterion to determine the quality of deployment plans (see section 3).

In this paper, we propose a component deployment planning method, which is applicable for smart item environments. More specifically, we propose an extended system model, evaluation functions for both cost and availability, and methods to determine model parameters. Using a practical application, we found that the number of network link uses is an important driver for the quality of a deployment plan. Based on that finding, we propose a heuristic for creating deployment plans with few network link uses. We show that applying this heuristic leads to good results in very short time.

The remainder of the paper is structured as follows: First, the characteristics of smart item environments are discussed and a set of requirements for deployment planning are derived. Afterwards, we review related work and point out their shortcomings with regard to our problem. In section 4 our solution is presented in an overview. Section 5 contains the core model for the CPP including evaluation functions and constraints. The extension of this model for smart item infrastructures is shown in section 6, where we propose methods to determine availability and resource demands. Finally, in section 7 it is investigated whether the two dimensions cost and availability are competitive, i.e. form a trade-off. Furthermore, we show that the number of network link uses is a major driver for the quality of deployment plans and present an algorithm, which creates deployment plans based on this heuristic.

## 2 Problem Analysis

Smart item environments have some characteristics which place special requirements on deployment planning methods. Here, these characteristics are briefly discussed to derive requirements from them. The requirements provide a rationale for the deployment planning method we propose and serve as basis for identifying weaknesses in related works. The main characteristics of smart item environments are:

- **Heterogeneity of infrastructure:** Infrastructure nodes in the smart item environment can range from resource-constraint embedded system, to conventional personal computers and middleware servers with vast resources. Network links connecting these nodes do also have different capacities.
- **Intermittent connections:** PEIDs are typically connected to the middleware using wireless connections that are not permanently available. This is either due to restrictions in the technology, e.g. mobile phone networks do not have full coverage, or a result of application specifics. For instance, if a PEID in a truck connects to a middleware access point in a depot using wireless LAN, it is unavailable during the time when the truck is not in connection range.
- **Distributed data sources:** Smart item environments are mainly employed to collect and analyse product data, e.g. static product information, the product structure, the operational status of the product, as well as historical records of owners, users, maintenance operations, etc. This data can be provided from local memory of the PEID or read from sensors that are integrated in the product. Other examples for data sources are rule repositories used for data analysis and thresholds, which might be stored on a middleware server. These data sources have a certain location in the infrastructure and send a response of a certain size when they are queried.

A deployment planning method for smart item environments has to take all these characteristics into consideration by fulfilling the following requirements:

1. **Consider cost of demanded resources:** The method shall consider the cost of resource demands at different hosts in the infrastructure. Although there are various resources, the method should at least take CPU, memory, and bitrate into account. These are the resources that are particularly scarce at the edge of the network, e.g. an embedded system has only a small memory, a very limited CPU power, and might only have low-bitrate connectivity, such as GPRS[2] or IEEE 802.15.4[3].
2. **Evaluate the effect of intermittent connections:** Intermittent connections influence the availability of data in the smart item environment. Component deployment plans can lead to better or worse availability. Therefore the method shall evaluate the effects of intermittent connections on the availability.
3. **Explicit modelling of distributed data sources:** Resource consumption depends on the data amounts that have to be transferred between the components and thus between their hosts in the infrastructure. As the traffic originates from the distributed data sources, the method shall provide means to explicitly model the location and message sizes of data sources.

[2] General Packet Radio Service, a packet-oriented communication protocol on GSM mobile phone networks.

[3] An IEEE standard for low-rate Wireless Personal Area Network (WPAN) connectivity.

## 3 Related Work

There is currently no component deployment planning method, which specifically addresses the domain of smart item environments. Hence, we review existing methods from related areas, e.g. mobile applications, grids, and computing clusters.

Mikic-Rakic *et al* present (re-)deployment planning for components in the context of PRISM [10], a middleware for distributed and mobile applications. In this environment, hosts are resource-constrained devices connected with intermittent wireless links. Component redeployment aims to improve the availability of a system. Special focus is put on the evaluation of planning algorithms, e.g. an approximative algorithm based on ordered lists of hosts and components [8], a decentralised algorithm with an auctioning mechanism [9], greedy and clustering algorithms [11]. The input model allows specifying memory constraints, and evaluation of bandwidth constraints through frequency of message exchanges between components and the average message size. The approach is very comprehensive, however, it does neither support the modelling of data sources nor evaluation of CPU utilisation. It also does not consider different costs of resources.

Another approach [4] was proposed for resource-aware deployment planning for component in grid environments based on Artificial Intelligence (AI) methods. For each component the required CPU and bandwidth must be defined to compute a resource-optimal deployment plan, which fulfils a deployment goal specified by the user (such as component *c1* should run on host *h1*). Additional components for encryption, caching, compression etc. may be added to the deployment plan to adapt the resource demands to the infrastructure's capacities. Although the presented approach is sophisticated, it is not suitable for our purposes, as it does not support heterogeneous infrastructures and modelling of data sources. Also, the effect of intermittent connections is not considered.

Steward *et al* propose automatic deployment for components of a J2EE application running on a cluster of computers [12]. This method aims to find a deployment that maximises the throughput of the distributed application but does not evaluate resource consumption. The method is not applicable for smart item environments as it assumes homogeneity of nodes. Deployment plans are evaluated only in terms of throughput but not for cost or availability. Finally, modelling of data sources is not possible.

Dynamic networks with intermittent connections play a key role in the deployment method for hierarchical components in a heterogeneous distributed system [13]. Unlike other approaches, the deployment plan is not calculated in advance but determined dynamically during the deployment process in a propagative manner. In the context of our elaborated requirements the approach is not applicable, as it only seeks a satisfying solution rather than evaluating different valid deployment plans. Furthermore, it does not support modelling of data sources and different costs for resources in the network.

An allocation algorithm for the placement of complex CORBA components is presented by Wu *et al* [14]. The method supports modelling of resource demands and constraints as well as global weighting of the resources memory, CPU and bandwidth according to their importance in the respective situation. Components are placed in order of their allocation priority, which is derived from the weighted ratio of resource demand and sum of available resource across all containers. However, the method neither considers intermittent connections, nor supports modelling of data sources. It does not allow for assigning different costs for these resources on each host. Finally, the

modelling used in this approach is very complex, which is appropriate for CORBA components but less applicable for the simple data processing components in our case.

In summary, existing component deployment methods do not model distributed data sources and are all based on a single evaluation criterion. Resource constraints are considered in some approaches but no cost-based evaluation of resource demands is performed, i.e. the resources are valued the same on all hosts. It can be assumed that the degree of heterogeneity in scenarios addressed by existing methods was low, which made cost-based evaluation unnecessary.

## 4 Proposed Solution

We propose a solution to the component placement problem that addresses the requirements stated above. Its overall approach is to create and rank deployment plan candidates by evaluating their cost of resource demands and their availability (Figure 1). Expected resource demands are determined, and added as annotation to the composition model. On the basis of a given deployment plan, these demands are mapped to the infrastructure model to relate the demands with the respective cost and capacities. If no resource constraints are violated, the availability of the system and the cost of utilised resources is calculated and compared to the best plans found so far. When all deployment plan candidates have been evaluated or the maximum number of plans to evaluate have been reached, the best ranking deployment plans are presented to the user.

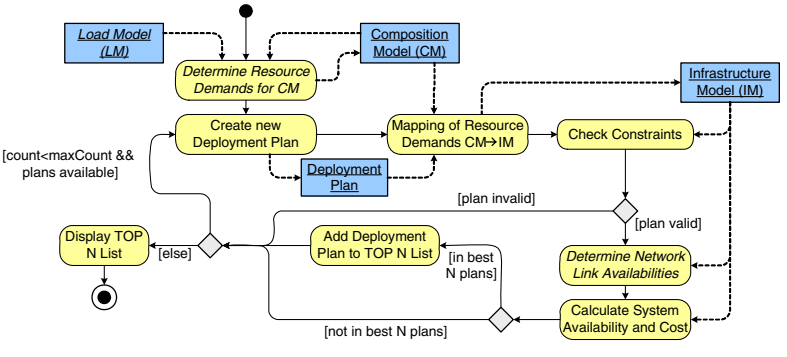


**Fig. 1.** Solution overview

Details of our solution are presented in the next two sections as follows:

*Core Model.* The basis of our solution is the core model of the component placement problem, which consists of the following elements:

- *Composition Model (CM)*, which specifies the composition of components, their dependencies, and resource demands. It contains the data sink and data sources.
- *Infrastructure Model (IM)* describing the structure of the network, the resource capacities of each host and network link, and cost per unit for these resources.

- *Mapping function* describing the assignment of components to hosts for mapping resource demands in the CM to resource capacities and costs in the IM.
- *Constraints* to validate deployment plans.
- *Evaluation functions* to calculate quality measures (availability and cost of demanded resources) of valid deployment plans.

*Determining Model Parameters for Smart Item Environments.* The model requires a number of parameters, which have to be supplied when the model is applied for component deployment in smart item environments. We go beyond estimating these parameters by proposing methods to determine availability as well as the demands for bitrate and CPU based on a load model. As these methods for determining parameters are decoupled from the CPP model, they can easily replaced by other ones when appropriate. Activities related to determining parameters are highlighted with italics in Figure 1.

## 5 Core Model of the CPP

**Composition Model.** The composition model is represented as a connected, directed composition graph $G$, consisting of a set of nodes $\mathcal{C}$ and set of dependencies (edges) $\mathcal{D} \subseteq \mathcal{C} \times \mathcal{C}$. The set $\mathcal{C}$ consists of a set of nodes $\mathcal{C}_R$ that can be relocated and a set $\mathcal{C}_F$ of nodes which are fixed to a specific host. The number of relocatable components $C$ and dependencies $D$ is the cardinality of the respective set: $C = |\mathcal{C}_R|$ and $D = |\mathcal{D}|$.

- *Data Sources and Data Sink* It is characteristic for each component that it receives an input and produces an output of data. Therefore, each component depends on one or more other components. Besides components there are two other node types in the composition graph: First, there can be one or more data sources that only provide output of data. Second, there must be exactly one data sink, which only receives data input. Data sink and data sources represent endpoints in the composition graph and belong to the set $\mathcal{C}_F$ as they are fixed to a specific host.
- *Resource Demands* For all components $c \in \mathcal{C}$ a resource demand $R_z(c)$, where $z = \{mem, cpu\}$ depends on memory and CPU power. Similarly, for all dependencies $d \in \mathcal{D}$ in the composition graph we assign the required bitrate $R_{br}(d)$ for the communication between the respective two components.

**Infrastructure Model.** The infrastructure onto which components are to be deployed, is modelled using a connected, undirected infrastructure graph $I$. It consists of a set of hosts $\mathcal{H}$ and a set of network links $\mathcal{L} \subseteq \mathcal{H} \times \mathcal{H}$.

- *Resource Capacities* For each host $h \in \mathcal{H}$ the available capacity $S_z(h)$ of memory and CPU are stored, $z = \{mem, cpu\}$. The same applies to network links, each of which holds a value $S_{br}(l)$ describing its available bitrate of each link $l \in \mathcal{L}$.
- *Cost of Resource Units* As mentioned before, we use a cost-based evaluation of resource demands to address the heterogeneity of hosts and network links in the infrastructure. Thus, we assign the costs $W_z(h)$, $W_{br}(l)$ for a unit of memory and CPU power consumption, and for a required unit of bandwidth, respectively.
- *Network Availability* Each network link $l$ in the infrastructure is assigned a value $0 \leq a(l) \leq 1$ describing the availability of that link. This measure is important for evaluating the system's availability of a given deployment plan later on.

**Assignment of Components to Hosts.** For deployment planning, every component $c_j$ is assigned to a host $h_i$. Such an assignment is called a *component placement*:

$$c_j \rightarrow v(c_j) = h_i.$$

A *deployment plan* $v : \mathcal{C} \rightarrow \mathcal{H}$ is a set of component placements, such that each component of $\mathcal{C}$ is assigned exactly to one host of $\mathcal{H}$. On the opposite, every host can have assigned $0..C$ relocatable components. The *set of all deployment plans* is denoted by $\mathcal{V}$ and has the cardinality $V = |\mathcal{V}| = H^C$.

**Constraints**

*Static Assignments.* The subset of nodes $C_F$ in the composition graph are statically assigned to hosts, i.e. these assignments are the same in all deployment plans. *Static assignments* are primarily used for data sources and the data sink as they can not be relocated. Additionally, user-defined static assignments are possible, if a component has to be placed on a specific host.

*Resource Constraints.* Besides static assignments, we have the requirement that the demand for resources does not exceed the capacity of infrastructure elements. For the hosts this requirement implies that the resource demand does not exceed the capacity

$$\sum_{j, v(c_j)=h_i} R_z(c_j) \leq S_z(h_i)\,.$$

Likewise it is necessary to formulate the constraint for the maximum bitrate demand on network links. This is more complicated as the communication between any pair of components can affect multiple network links in the infrastructure, if the two components are deployed to hosts which are not directly connected with each other. To formulate this constraint, we consider the communication path $\mathcal{P}$ between two components $c_i$ and $c_j$ within the infrastructure at a given deployment plan $v$. This path is a set of network links connecting the hosts $v(c_i)$ and $v(c_j)$ on which the components reside.

Now the constraint for the maximum bitrate demand on network link $l$, requires that the sum of all communication between neighbouring components that use this network link to be less than the capacity of this link:

$$\sum_{<i,j>} Q_l(\mathcal{P}(c_i, c_j)) \cdot R_{br}(d(c_i, c_j)) \leq S_{br}(l)\,.$$

Here, we introduced the projection:

$$Q_l(\mathcal{P}(c_i, c_j)) = \begin{cases} 1 & \text{, if link } l \text{ belongs to the path } \mathcal{P}, \\ 0 & \text{, else.} \end{cases}$$

**Evaluation functions.** If a valid deployment plan was found, both its cost of resource demands and its availability is evaluated. Although both measures can be used independently for evaluating deployment plans, it may be assumed that high availability implies high cost of resource demands.

*Cost of demanded resources.* The cost of resource demands for a given deployment plan $v$ is the total cost of resource demands, cumulated over all hosts and network links.

$$K(v) = \sum_{i=1}^{H} \sum_{z} Res_z(i) \cdot W_z(i) + \sum_{j=1}^{L} Res_{br}(j) \cdot W_{br}(j) \tag{1}$$

Here, $L$ is the number of network links, $H$ is the number of hosts in the infrastructure and $Res_z(i)$ is the total demand for resource $z$ on host $i$. Similarly, $Res_{br}(j)$ denotes the total bitrate demand on network link $k$.

*Availability.* For the evaluation of a deployment plan's availability, the availabilities $a(l)$ of all individual network links have to be aggregated. Availabilities can be considered as probabilities of success for communication between pairs of components over network links $l$. The availability of the deployment plan is determined by the product

$$A(v) = \prod_{l=1}^{L} a(l). \tag{2}$$

We note, that the determination of the link availability $a(l)$ is not trivial. We explain the method we have used in section 6.2.

## 6 Determining Model Parameters for Smart Item Environments

To use the presented core model for deployment planning, it has to be instantiated with actual values for the input parameters. In this section, we explain methods on how resource demands and the availability can be determined. As these methods are decoupled from the core model, it allows for any other way to determine the input parameters.

### 6.1 Determining Resource Demands

Some resource demands depend on other inputs and have to be calculated before a deployment plan can be evaluated. In principle, we follow an approach proposed by Steward *et al* [12]. It estimates resource demand for components based on "resource profiles", which are created "off-line" by measurements under different workloads.

**Load Model.** To calculate component-level resource demands, except memory, the load placed on the composition has to be known. Load refers to the number of requests a user issues over a period of time. Generally, the requests over time are POISSON-distributed. As our method only considers static deployment planning, the mean value of this distribution ($\lambda$-parameter) is sufficient to characterise the load. This parameter is named $iph$ (invocations per hour) and does logically belong to the data sink.

Besides the number of invocations, also the message sizes to be transferred have to be defined in the load model. As the data originates from the data sources, the message sizes are logically assigned to them. Therefore, for each data source the size of the message returned when it is queried has to be specified in the load model.

**Bitrate.** Bitrate demand $R_{br}(d)$ for the communication between components depends on the message sizes to process and the load. By multiplying the size of the message to process with the invocation per hour $iph$, we get the incoming bitrate. At each invocation, the incoming data is processed into outgoing data, whereby the size of outgoing data can be different. One approach to model this for simple functional blocks in building automation is used by Plönnigs *et al*. They use an amplification factor ($gain$) to describe the relation of inputs to outputs in processing devices [15]. We extend this by using a linear function $o_c$ to describe the input/output-relation for each component

$$IORel : o_c(i_c) = e_c \cdot i_c + f_c \,.$$

Here, $o_c$ is the output of component $c$, which depends on the input $i_c$, the amplification factor $e_c$ and the bias $f_c$. In our model, the input $i_c$ is the sum of all incoming bitrates for a component. Note that $e_c$ and $f_c$ are constant during the calculation.

**CPU Power.** The CPU demand $R_{cpu}(c)$ is calculated with a method proposed by Steward *et al.* [12], who used it to plan component distribution in a server cluster for maximum throughput. They describe the CPU demand as linear function, whereby load is the independent variable. The coefficient $a_c$ and the constant $g_c$ were gained by linear regression on a series of CPU utilisation measurements under different loads.

$$R_{cpu} : p_c(i_c) = a_c \cdot i_c + g_c$$

We adopt this method and use the amount of data to be processed by the respective component as load $i_c$. For each component, such a linear function has to be determined with different data amounts rather than with requests per second. A major difference between our work and the work by Steward *et al* is the heterogeneity in the infrastructure. While a server cluster consists of identical machines, the CPU power in a smart item environment is diverse. Therefore, we propose to compute the CPU demand function on a reference system, and adjust CPU capacities on each host to reflect its CPU power in relation to the reference system. For example, if an embedded system has only 5% of the CPU power of the reference system, its CPU capacity is set to 5. We recognise that this method allows only for a rough estimation of CPU demands. However, in our view it is a good balance between model complexity and accuracy for our purpose.

### 6.2 Determining Availability

For the evaluation of the system's availability (Equation (2) in section 5) the availability of all network links in the infrastructure is needed. To characterise intermittent network links, we introduce two parameters: (a) Mean connection duration $d_C$, and (b) Mean pause duration $d_P$ (see Figure 2). We present three different methods for calculating the availability of a network link, which is understood as probability of success for:

1. Network link availability
2. Immediate successful execution of a request
3. Successful execution of a request within a given time frame

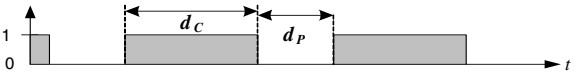


**Fig. 2.** Parameters to describe an intermittent connection

For simplicity we will denote availability as $a(l)$ for these probabilities and the specific context clarifies the meaning in each case.

**(1)** The probability of network link availability is the ratio between connection duration $d_C$ and the duration between two connection establishments ($d_C + d_P$)

$$a = \frac{d_C}{d_C + d_P} \tag{3}$$

**(2)** The probability of immediate successful execution of a request considers the time required to transfer the requested data amount. Based on the data amount $msg$ and the capacity of the network link $S_{br}$, the required transfer time $d_T$ can be determined by $d_T(l) = \frac{msg}{S_{br}(l)}$. The transfer of the requested data amount is successful, if both the connection is available and the transfer was started before the connection is terminated.

$$a = \frac{d_C}{d_C + d_P} \cdot \frac{d_C - d_T}{d_C} = \frac{d_C - d_T}{d_C + d_P} \tag{4}$$

This only is meaningful if the $d_T < d_C$, otherwise the request will not be successful and the availability of the whole system is set to $0$.

**(3)** For the probability of successful execution of a request within a given time frame, a maximum time $d_{max}$ has to be specified. The calculation is based on probability of the $n$-fold repetition of the complementary event ("transmission unsuccessful"):

$$a = 1 - (1 - \frac{d_C - d_T}{d_C + d_P})^n \text{ , whereby } n = \frac{d_{max}}{d_C + d_P} \mid n \geq 1\,. \tag{5}$$

*Multiple Uses of Links.* All equations defined in this section determine the availability of a network link for a single transmission. As every dependency in the composition graph represents a service invocation (request) and its result (response), the network link is used twice for each dependency mapped to it. Moreover, several dependencies can be assigned to a network link. If a link is used multiple times, its availability is the product of all availabilities for each individual use. For each use, the required transmission time $d_T$ might be different. Therefore, we consider the transmission time $d_{Ti}$ for transmission $i$ to determine availability using Equation (4) for multiple uses by

$$a = \prod_i \left( \frac{d_C - d_{Ti}}{d_C + d_P} \right) .$$

Similarly, the availability can be calculated with Equation (3) and Equation (5). In each case, it can be seen that the availability decreases when the number of network link uses increases.

## 7 Analysis of the Solution Space

We validate our proposed method by analysing the results of a practical application, which deals with maintenance planning for trucks and was adapted and extended from an earlier publication [5]. It consists of 11 components, which are to be deployed onto an infrastructure with 3 hosts. We show the solution space for all valid deployment plans and identify the location of the best deployment plans in it. Furthermore, the influence of the number of network uses on both cost and availability is analysed.

### 7.1 Analysis of Competition

The complete solution space for the base scenario is depicted in Figure 3.1. It shows the cost and availability of all valid component deployment plans. The best deployment plans (low cost and high availability) are located in the upper left corner of the diagram. Data points representing the best deployment plans are highlighted with circles and bounded by a rectangle, which marks the area between the two extremal points of lowest cost and highest availability. All deployment plans which are not highlighted can be discarded as they are definitely worse than the highlighted ones. There are 35 deployment plans which were identified as ”best”. For clarity, we name these ”deployment plan candidates”.

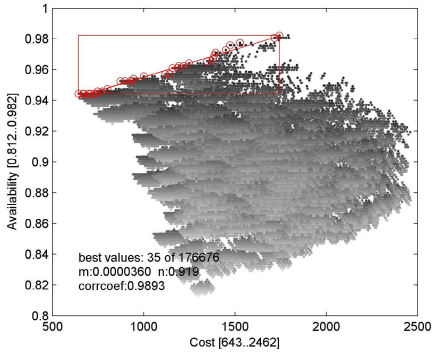


**Fig. 3.1** Complete solution space

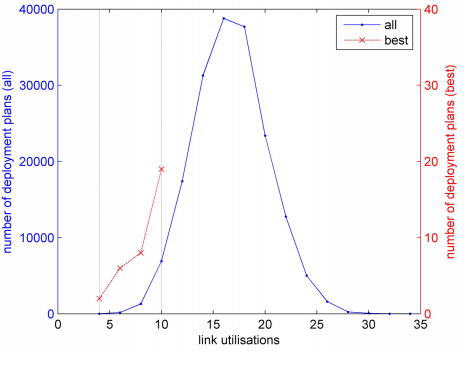


**Fig. 3.2** Distribution of network link uses

### 7.2 Number of Network Link Uses

We have analysed the effect of the number of network link uses on the quality of deployment plans. As briefly discussed in section 6.2, the availability depends on how many times a network link is used. Furthermore, the cost is also influenced by this measure as the cost for transmission depends on how much bitrate demand from dependencies is mapped to network links. Therefore, it can be assumed that the number of network link uses is an important factor for the quality of a deployment plan.

To verify this hypothesis with our example, the number of network link uses is represented by the colour of data points in Figure 3.1, whereby darker points represent

deployment plans with fewer link uses. As it can be seen, there is a tendency that good deployment plans utilise network links fewer times. To further investigate this, we have analysed the position of the best deployment plans in the distribution of link uses.

As Figure 3.2 shows, the deployment plan candidates utilise network links 10 times or less in this example. This is an important finding, which helps to design a heuristic search for good deployment plans in the solution space without complete evaluation of all possible combinations.

## 8 A Heuristic Algorithm for Finding Deployment Plans

From our findings, we have derived a heuristic algorithm that finds good deployment plans without scanning the whole solution space.

*Heuristic Generation of Deployment Plans.* As depicted in Figure 1, our method creates a number of deployment plans for evaluation and stores the best found ones. Using the heuristic that a low number of network uses are a characteristic of good deployment plans, our algorithm places neighbouring components only on the same hosts or on neighbouring hosts. Therefore each dependency of the composition model is mapped only to either 0 or 1 network links. The recursive algorithm is initialially invoked with the data sink as argument (see Algorithm 1).

**Algorithm 1.** placeDependentComp($start$)

1: find host $h$ on which $start$ is placed
2: find all hosts $H_n$, which are direct neighbours of $h$
3: find all components $K_n$, on which $start$ depends
4: **for all** $k_j$ in $K_n$ **do**
5: randomly select host $h_i$ from $(H_n \cup h)$
6: place $k_j$ on $h_i$
7: **placeDependentComp**($k_j$)
8: **end for**

*Evaluation.* To evaluate the quality of the heuristic, we have compared it to another method, which creates deployment plans based on random placements of components to hosts. Both the heuristic and random assignments were used to evaluate various percentages of all combinatoric possible deployment plans. The quality criterion used is the mean euclidean distance of found deployment plan candidates to the nearest plan found by an exact algorithm, i.e. the optimum.

The results in Figure 4 show that the deployment plan candidates found by the heuristic algorithm are closer to the optimum than almost all random component placements. Furthermore, it shows that good results can achieved without evaluating a large number of deployment plans. However, it can also be seen that the optimum is not reached. The reason for this is that the highest availability is achieved in this scenario by placing all components on the embedded system. This means that there is more than one network link between the data source and the first dependent component. This is prevented by the heuristic algorithm which only allows a maximum distance of one host between any pair of neighbouring components.

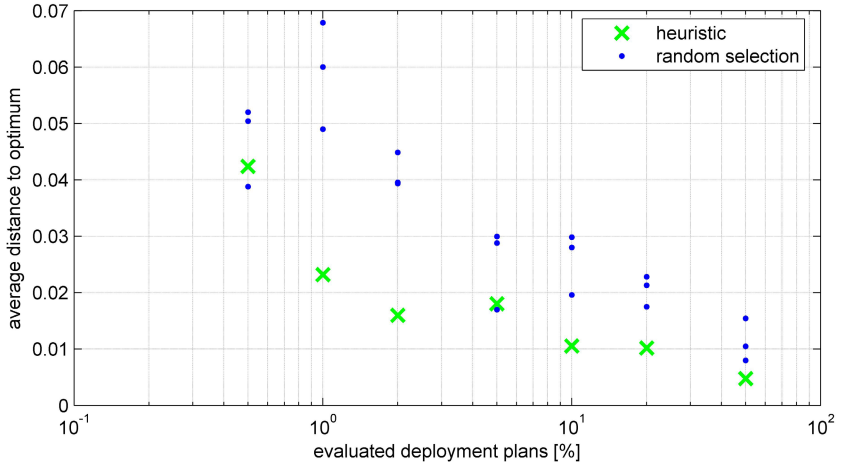


**Fig. 4.** Evaluation of Heuristic Accuracy

## 9 Conclusion and Outlook

We have presented a deployment planning method for components that addresses specifically distributed components in smart item environments. These networks are characterised by a high degree of heterogeneity in terms of available hardware resources. The main contribution of this paper is a concept for evaluating deployment plans both in terms of availability and cost of demanded resource. We have shown that these two criteria compete with each other among the deployment plan candidates in the solution space. Furthermore, we have presented a comprehensive model for component deployment, which might serve as basis for other research questions in the domain of smart item environments. Additionally, we have identified the number of network link uses as a key driver for the quality of a deployment plan and derived a heuristic from this finding. As the evaluation showed, the application of this heuristic helps to find very good deployment plans after testing only a small fraction of all possible plans.

In the future, our work will focus on improved heuristic algorithms for creating deployment plans which are likely of high quality. For that, additional characteristics of good deployment plans, such as the average distance of components to the data sinks, are investigated and integrated into the algorithms.

## Acknowledgements

The authors would like to thank Mario Neugebauer and Eric Neuber for their valuable comments, and Jürgen Zimmermann for his support with the implementation.

## References

1. OSGi Alliance: Open Services Gateway Initiative (2006)
2. SUN Microsystems: Jini Network Technology (2006)

3. Anke, J., Neugebauer, M.: Early data processing in smart item environments using mobile services. In: Proceedings of the 12th IFAC Symposium on Information Control Problems in Manufacturing (INCOM 06) St. Etienne, France (2006)
4. Kichkaylo, T., Karamcheti, V.: Optimal Resource-Aware Deployment Planning for Component-Based Distributed Applications. In: Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC'04), pp. 150–159. IEEE Computer Society, Los Alamitos (2004)
5. Anke, J., Kabitzsch, K.: Cost-based Deployment Planning for Components in Smart Item Environments. 11th IEEE International Conference on Emerging Technologies and Factory Automation, Prague, Czech Republic (2006)
6. Institute of Electrical and Electronics Engineers: IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries, New York (1990)
7. Wegdam, M.: Dynamic reconfiguration and load distribution in component middleware. PhD thesis, University of Twente, Enschede (2003)
8. Mikic-Rakic, M., Malek, S., Medvidovic, N.: Improving availability in large, distributed component-based systems via redeployment. Third International Working Conference on Component Deployment, Grenoble, France (2005)
9. Malek, S., Mikic-Rakic, M., Medvidovic, N.: A decentralized redeployment algorithm for improving the availability of distributed systems. Third International Working Conference on Component Deployment (2005)
10. Malek, S., Mikic-Rakic, M.: A style-aware architectural middleware for resource-constrained, distributed systems. IEEE Trans. Softw. Eng. 31(3), 256–272 (2005)
11. Mikic-Rakic, M., Malek, S., Beckman, N., Medvidovic, N.: A tailorable environment for assessing the quality of deployment architectures in highly distributed settings. Second International Working Conference on Component Deployment, Edinburgh, UK (2004)
12. Stewart, C., Shen, K., Dwarkadas, S., Scott, M.L., Yin, J.: Profile-driven component placement for cluster-based online services. IEEE Distributed Systems Online 5(10), 1 (2004)
13. Hoareau, D., Mahéo, Y.: Constraint-Based Deployment of Distributed Components in a Dynamic Network. In: Grass, W., Sick, B., Waldschmidt, K. (eds.) ARCS 2006. LNCS, vol. 3894, pp. 450–464. Springer, Heidelberg (2006)
14. Wu, Q., Wu, Z.: Adaptive component allocation in scudware middleware for ubiquitous computing. In: Yang, L.T., Amamiya, M., Liu, Z., Guo, M., Rammig, F.J. (eds.) EUC 2005. LNCS, vol. 3824, pp. 1155–1164. Springer, Heidelberg (2005)
15. Plönnigs, J., Neugebauer, M., Kabitzsch, K.: A traffic model for networked devices in the building automation. In: Proceedings of the 5th IEEE International Workshop on Factory Communication Systems (WFCS 2004) Vienna, Austria, pp. 137–145 (2004)