# DECLARATIVE RESOURCE DISCOVERY IN DISTRIBUTED AUTOMATION SYSTEMS

Christoph Gerdes[1], Christian Kleegrewe[1], Jörg P. Müller[2]
[1]Siemens AG, Corporate Technology, Information & Communications, Germany, [2] Clausthal University of Technology, Germany

Corresponding author: Christoph Gerdes
Siemens AG, Corporate Technology
Information & Communications
Otto-Hahn-Ring 6, 81739 Munich, Germany
Email: c.gerdes@siemens.com

**Abstract.** Engineering of complex automation systems is a major cost factor. One reason for the complexity of engineering processes is the tight coupling of automated functions and the actual automation device. The paper introduces a method and architecture to de-couple functions from devices. As foundation Peer-to-peer and Grid computing technologies are applied to provide a flexible framework for automated functions. A key issue is the applicability in actual industrial systems which is accounted for by designing an algorithm which can be implemented cost efficiently on resource constraint devices. The approach is evaluated through extensive simulations which demonstrate benefits and applicability of the proposed architecture.

## 1 Introduction

Engineering is the major cost factor in the construction of state of the art industrial systems. Automation equipment features a multitude of parameters that need to be configured and parameterized during commissioning of a plant. Additionally, the environment is highly heterogeneous with multiple equipment vendors and product versions. Finally the recent demand from customers to build more flexible, easier to customise and fully integrated manufacturing systems adds to the complexity.

Several approaches have been proposed to cut engineering costs. One way to achieve more flexible automation system is to shift to decentralised architectures [14] [11]. However, these approaches mark a major paradigm shift in automation which is currently not supported by neither equipment vendors nor system integrators. The key problem, however, the so called point to point dilemma, i.e., the tight coupling of automated functions and resources remains unaddressed. Today systems are designed statically by directly specifying end-to-end communication. For example, sensor S delivers data to motor M. If M is replaced, or other units are interested in data generated by S, each new connection needs to be configured end-to-end. Thus an enormous conglomeration of static communication links is the result. Besides being expensive to configure, the static interweaving of automation equipment prevents flexible execution of control software.

In this paper, we present an architecture capable of de-coupling individual automation equipment. We propose a robust and self-organising system to discover resources at runtime in a networked automation system using declarative resource discovery. We use Peer-to-Peer (P2P) and Grid computing technology originally designed for the Internet and advance them such that they can be implemented on even resource constraint equipment. P2P systems such as Chord [18] and CAN [15] provide the foundation for loosely coupled systems in the Internet. However, both their code complexity as well as stabilisation effort, i.e. the intensity of required communication, suggest cost intensive realisations and hence those approaches are rather unlikely to emerge in industrial products. We therefore investigate how a lightweight algorithm can provide similar features albeit being less capable in an Internet scale, i.e., millions of users, scenario.

The structure of this article is as follows. First we describe the related art in P2P computing and its application in automation systems. After that a system overview introduces architecture and core concepts like queries, stabilisation and grid functionality. Providing a simple application example the discovery process is illustrated. Sections with simulations and the discussion of our results conclude the paper.

## 2 Related Work

In recent years structured P2P protocols have been extensively studied. Ratnasamy et al. suggest in [15] a multidimensional hash table for distributed data management whereas in most of the related work like [18], [2] different routing algorithms in one dimensional distributed hash tables (DHT) are suggested. Since DHTs are not suitable for complex queries such as range queries, there is considerable research effort in enhancing such distributed data structures in a way that range queries can be managed at low costs. One major approach in this field of research is the application of prefix hash trees (PHTs) as proposed in [17]. This approach suggests that data is stored in a tree structure that grows and shrinks dynamically with the amount of data being stored. Data is sorted in the tree leaves according to a prefix matching algorithm. The prefixes of the tree nodes are then stored in a DHT. For range queries the first node that contains data is evaluated by consistent hashing and after finding this node only the subtree beyond this tree node must be searched for data. A similar approach is suggested in [12]. A different method for range queries in DHT structures is based on space filling curves, first described by Guiseppe Peano in 1890. Such search algorithms are described in [1], [8] and [21]. In [1] a range query method is introduced that is optimised for information search in grid information services. Spanning Trees to enhance routing capabilities in P2P networks are evaluated in [6] and [10]. In [10] a self-stabilising spanning tree algorithm is suggested that is capable of handling churn events like common DHT protocols for Internet scale applications. The algorithm additionally provides optimum results in networks where no global network view can be provided. In [6] an algorithm is described that allows placing of processes in a self-healing and ordered spanning tree in which distributed object queries are routed. In the context of industrial automation systems, concrete P2P mechanisms are proposed in Drinjakovic et al. [5] who suggest lookup methods in a process control system using P2P networks. Thereby information resources are grouped according to a structured naming scheme and search methods provide deterministic access at runtime. The described query method, however, is one dimensional and thus limits queries to keyword searches.

## 3 Overview of our Approach

We address the point to point dilemma by designing automation systems as loosely coupled orchestrations of automation assets. Similar approaches are taken in service oriented architectures (SOA) such as sensor Grids or similar Grid computing [7] applications. In SOA, assets are described by the service, i.e. the function, they provide. Services encapsulate resources which execute the functions provided by the service via a service interface which in turn provides a consistent view on the resource. Besides the service interface, a service description provides information about the interface as well as qualities of operation and management. In order to use a service, a service consumer states his interest in a service query and issues the query at a service registry which maintains all available service descriptions. Once found, the service consumer binds to the service first and then executes the desired functions (Figure 1). A key advantage of the service oriented approach are the different temporal options for service binding. Early binding is referred to at design time when requirements are mapped to service descriptions. Late binding, in contrast, renders a system more flexible as the service consumer binds the service at runtime. Ultra-late binding takes the concept one step further as applications are created by composing services dynamically for each invocation and removing services from the application after they are no longer needed.

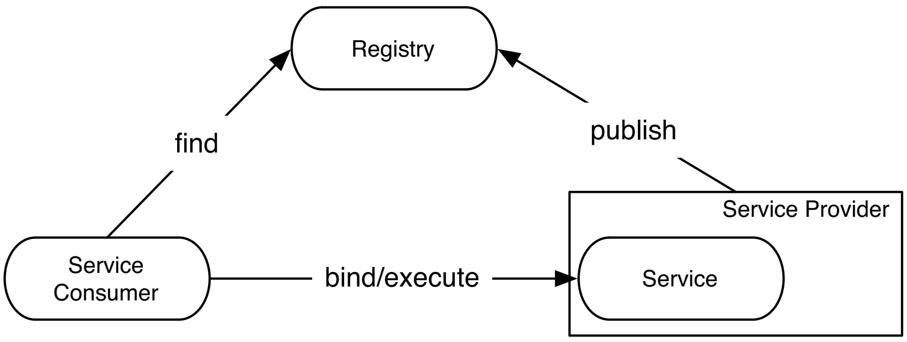


**Figure 1:** Actors in service oriented architectures: provider, consumer and registry

In order to support late and ultra-late binding paradigm, the service registry must provide efficient means to process the service query and match it to service descriptions accordingly. Solutions for this service discovery process range from simple keyword searches to evaluation of fuzzy logic and semantic reasoning. While the expressiveness of keyword searches is often not sufficient for complex service descriptions, the resource demand for semantic matching prohibits cost efficient solutions. A flexible compromise is provided by complex declarative queries, common in most database systems. Query languages like the standard query language (SQL [4]) provide rich semantics to express complex interests yet they can be implemented efficiently.

While centralised registries can be implemented efficiently, they are additional infrastructure components that need to be integrated and maintained. Distributed registries, however, use already existing resources of the networked assets. They are further more robust and scale dynamically with the number of networked assets. In the following paragraphs we describe a generic service oriented architecture for loosely coupled automation devices. Being a key component, we emphasis in the description of a distributed service registry with support for declarative queries.

Figure 2 illustrates the multi layer architecture of a service oriented automation system. As foundation, the distributed registry collects service descriptions and provides query interfaces. Networked assets are interlinked via a self-organising overlay network. The overlay acts as network virtualisation by abstracting physical network topologies and providing content based addressing schemes. A query engine parses incoming queries and generates a query execution plan. A data management component organises service descriptions and handles their storage on respective assets. Building on the abstract service descriptions, a resource virtualisation layer coordinates resources allocation. It provides a level of abstraction to establish location transparent resource access. Further it allows to aggregate several atomic resources to higher level concepts. For example a boiler is a high level concept being composed of several assets, e.g., sensors, pumps and heaters. Finally the service level combines atomic and composite resources to services and service workflows like automated functions or monitoring and analysis functions.

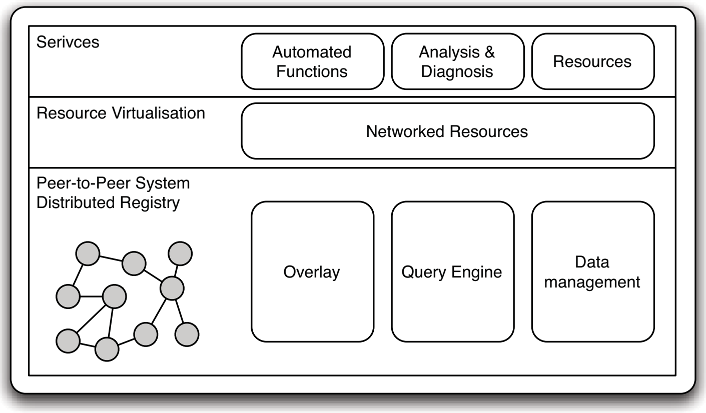


**Figure 2:** Automation Grid Architecture

Before we can describe the resource discovery process we need to introduce a simplified resource model. The model is based on Chen's Entity-Relationship model [3]. It is rather generic and can be mapped to more advanced approaches like the WS-* standards. Based on this model, a domain specific query language is introduced that enables declarative resource discovery.

## 3.1 Resource Model

The concept of a *resource* denotes any asset in a networked automation system such as sensors, actuators, as well as information entities like device states and condition. Each resource $R$ is modelled through a finite and ordered list of features $f_1..f_n$. Features have a class $L$ and a value $V$ as well as an informal description. The set of all features of all resources span a vector space $F$. A resource relationship is an association among resources. For instance, a resource temperature sensor is associated to a resource controller as it delivers required measurements. A correlation of resources describes a temporary semantic similarity of resources. Correlations can be quantified via a distance metric $M$. Hence correlated resources can be organised in a cluster structure. Put formally, resource $R_i$ belongs to group with centroid $G_i$ iff $\|R_i - G_i\|_M \leq r$ where $r$ is an application specific radius. Whereas the centroid is a numerical representation of the correlation.

## 3.2 Peer-to-Peer Subsystem

As peer we understand a software component that is hosted by networked assets. Resources are managed by peers and all peers are interconnected in a P2P network. Resource correlations emerge dynamically during the lifetime of a system. There are computed continuously at individual peers by exchanging information on hosted an associated resources and applying a correlation function.

The topology of the overlay network is determined by resource relationship and correlations. Peers hosting correlated resources are more likely to be neighbours while peers with non correlated resources are unlikely to be directly linked.

## 3.3 Resource Discovery

Supporting existing programming models, automated services can be defined using common standards, e.g., IEC 61131, augmented with a declarative semantics for resource discovery. Hence resources are addressed not directly using physical memory or network addresses but rather declaratively, namely using queries that describe what features of the resource are required. To support query composition and execution, we developed a query language to express resource interest in an abstract form, hiding unnecessary details. A complete description of the language is beyond the scope of this paper. In the following we introduce the key features of the language by discussing a simple query statement.

**Listing 1:** Example of a continuous query

```
SELECT * FROM RESOURCES
       WHERE resource
       HAS FEATURE(r=0, 575, 35345, 767854, *)
       TOP 1
       WINDOW(0, FOREVER, 1s)
```

The example provided in Listing 1 illustrates a simple resource query stated in our language. The language is based on the well known and widely adopted standard query language (SQL [4]) but augmented with additional statements for resource selection. While the SELECT ... WHERE clause is standard, the HAS FEATURE(TUPLE), specifying the requested resource, is a unique extension in our system. A tuple consists of a specification of the maximum distance $r$ between query and resource whereby $r = 0$ limits the result to only exact matches. The rest of the elements specify the features of the resource being searched for. The wildcard character '*' causes the particular feature to be excluded from the evaluation. The TOP $n$ operator reduces the result set to the n resources closest to the feature specification. Queries can run either once as snapshot query or continuously over a period of time. If run once, the query returns a single result set while if running continuously, the query initiates a data stream of result sets. The WINDOW operator specifies the activation interval of the data stream. The first two parameters set start and end time while the third parameter specifies the interval of execution, e.g., every 10 seconds. Once injected into the network, queries are compiled to binary form and loaded in the local query engine. An execution plan (Figure 3) is generated and scheduled for processing. The plan lists all actions necessary to deliver the requested result set. The plan depicted in Figure 3 first checks locally if the query can be evaluated with local information alone. If so query execution is complete. Otherwise the query is optimised and rewritten for distributed execution. Query optimisation is a complex procedure where the query is restructured to reflect the overlay topology as well as states of other concurrent queries scheduled in the same query engine. Afterwards a resolving action is triggered which determines which peer might provide the requested information. This procedure in discussed in detail in section 4. Subsequently the query as a whole or in part is assigned to the peer accordingly. Intermediate results are stored in the local storage. The process continues until all requested information is contained in the local storage. In case the maximum number of retries is reached or a timeout occurred an exception is raised and the execution ends.

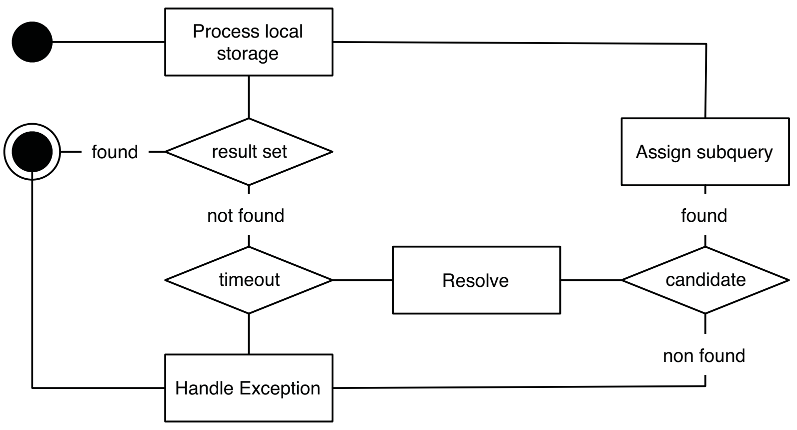


**Figure 3:** Query execution plan

Continuous queries are constantly evaluated, and hence can adapt to network reconfigurations and resource dynamics. Therefor, each peer collects information on resource states and conditions from its neighbors and re-optimises the query to reflect the new network condition.

## 4 Routing and Stabilisation

Besides query optimisation, resolving of candidates for query processing is a key element for every execution plan. Each peer stores service descriptions of the resources it hosts in its local storage. Descriptions of composite resources are hosted by all peers having associated resources. Hence the problem to address is to find a mapping from the content requested via the query to a respective peer that can provide the content. In distributed hash tables a hash function is applied which maps the query, i.e., the keyword to an identifier space which is partitioned over all peers participating the DHT. For the distributed registry as described in section 3 a multidimensional lookup mechanism is required which locates a resource based on a set of features specified in the query. Using a DHT, this would require to maintain multiple overlay structures, e.g., a Chord ring for each feature causing considerable compute and network overhead. Multidimensional approaches such as CAN seem more suitable but are restricted by their enormous code complexity which yet let to only small experimental deployments in artificial environments. Additionally, using a uniform hash function, peers in standard DHTs are treated as equal independent of their resource capacities. Data is stored as determined by the hash function, thereby not considering the specifics of individual peers. Hence, resource constraint devices might get easily overloaded with popular keywords, e.g., 'temperature' whereas other devices might have unused capacities. On the other hand, unstructured approaches like Gnutella v2, are able to assign different roles to peers depending on their network capabilities. However, due to their non deterministic behaviour, i.e., stored items are not guaranteed to be found, they have not been considered in technical systems. Playing on the peculiarities of automation systems features of both DHTs and unstructured approaches can be combined to support complex query execution.

Automation systems are built for reliable operation for long periods of time. Thus, in comparison to the Internet [19], from which most P2P algorithms originate, we can assume the environment to be rather static i.e. device failure is rare compared to the time required to stabilise routing tables. Having to deal with limited dynamics an iterative approach based on *K-means* clustering, e.g., [20] is chosen to stabilise the overlay and route information between peers.

Routing and stabilisation evolves through two phases. In the first phase the system initialises, i.e., peers exchange information about their resources, features and physical network addresses. Based on this information exchange, peers cluster around certain centroids. In each cluster peers maintain a set of links to their neighbours. Based on their neighbour links local routing tables are built which are used to map features and combination of features to peers. The first phase ends when clusters and therefore routing tables stabilised leaving the system in a ready to operate state. It is important to note that any further communication between entities in the system is based on the routing information initialised during this phase. Only in the event of unit failure or reconfiguration, routing information needs to be updated. This ensures that communication can be realised very fast by conducting only lookup in local routing tables. In the 2nd phase peers monitor their resources as well as the resources of their neighbours in the cluster. If a peer departs from a cluster it drops its routing tables and enters phase 1 to locate an alternative cluster. Communication in phase 2 is gradually reduced to a minimum. If a peer failure is detected, a notice is propagated within the cluster causing all connected peers to revaluate their connections within the cluster.

Having provided a high level overview of the routing and stabilisation procedure we now examine the cluster formation and inter-cluster communication in further detail. At iteration $k = 0$, the network is initialised with a random set of centroids $\{C_{j,i,0} : 1 \leq j \leq K\}$ distributed over all peers. Each peer counts the number of resources associated to a centroid $\|C_{j,k}\|$. In each iteration, the $i$th peer $P_i$ picks a selection $C$ of $n$ peers at random from the set of all peers and exchanges its local set of centroids and the resource counts with the selected peers. Having received the centroids each peer begins to update, i.e. compute the mean, its local centroids thereby using resource counts as weights. Peers in $C$ that have resources associated to the same centroids as $P_i$ are stored locally in a neighbour table. This process is continued until the change of centroids between iterations is smaller than a predefined threshold. Once the change of centroids of all peers drop below the threshold, the initialisation phase terminates and each peer has its features associated to at least one centroid.

While clusters are highly connected, further links for inter-cluster communication need to be established. Therefore, for each cluster, bridge peers are determined that establish links to other clusters. Initially in the process each peer assumes it is the bridge to other clusters and forwards this information along with its distance to the target cluster to all of its neighbours. Peers process this information and select the neighbour closest to the target cluster as bridge peer. Acting as bridge, respective peers collect additionally to cluster neighbours also peers from clusters they bridge to. Consequently inter-cluster links are established as the algorithm iterates.

## 5 Application Example

To further illustrate the workings and benefits of the declarative resource discovery mechanism, consider the temperature control system depicted in Figure 4 based on [13]. It consists of an oven, e.g., to cure products made of epoxy resin. Attached to the oven is a temperature sensor and a heater unit. We assume that both heater and sensor are intelligent units that provide some form of communication capabilites as well as a compute component to handle query processing.

A controller unit monitors the temperature readings of the sensor and sends control commands to the heater in order to maintain the required temperature in the oven. It applies a simple PID controller to adjust the heater intensity. This simple control loop, where measurements are acquired, processed by control logic and finally control actions are executed is exemplary for a whole range of industrial control problems. Listing 2 shows the control program.

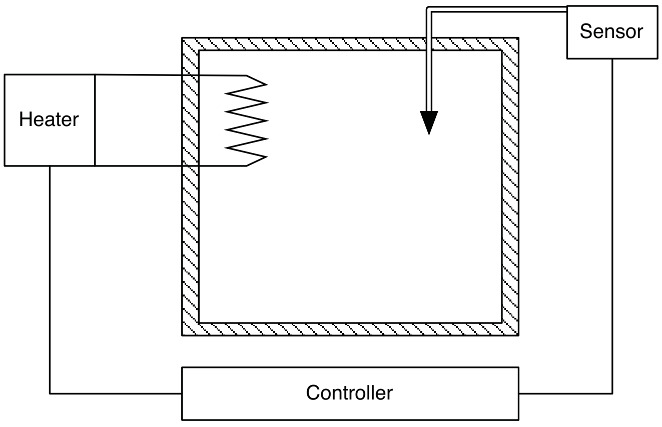


**Figure 4:** Simple control loop

**Listing 2:** Simple control program

```
 1 DO
 2         heaters -> SELECT * FROM RESOURCES
 3                 WHERE resource
 4                 HAS FEATURE(r=0, 'heater', 'oven')
 5
 6         sensors -> SELECT * FROM RESOURCES
 7                 WHERE resource
 8                 HAS FEATURE(r=0, 'temperature', 'oven')
 9
10         heater.temperature = PID(sensor.temperature, 180)
11 LOOP
```

The system is modelled with heater and sensor as atomic resources while the oven is a composite resource comprised of heater, sensor and controller (Figure 5). The oven contains two associations namely heaters and sensors which are defined as resource queries. For this simple example we assume that respective devices can be uniquely identified by the specification of only two features in the query. The relations heaters and sensors contain all sensors and heaters respectively that are currently installed. The feature valued 'oven', causes all entities to group in a single cluster. Upon query execution, the resolver iterates through the list of neighbours matching the query against the features of the neighbour peers. Subsequently full service descriptions are retrieved from the matching peers and cached locally.

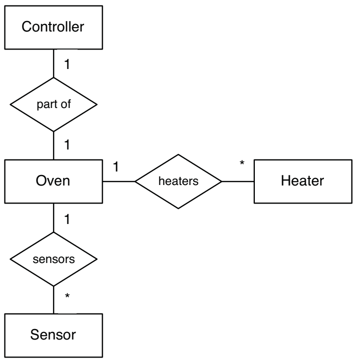


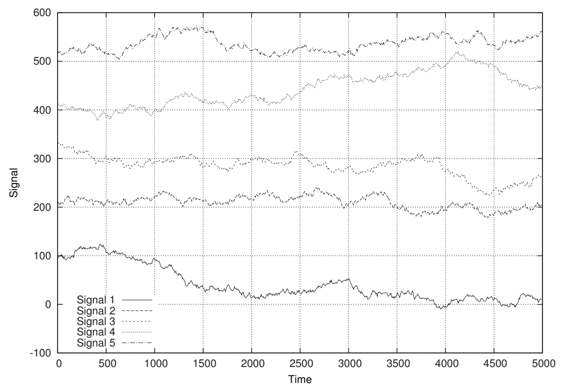
**Figure 5:** Simple control loop

The statement at line 10 sets the temperature feature of the heaters according to the control logic. In case of multiple elements in the heaters relationship all features are set accordingly. Thereby the data management component ensures consistent propagation of the feature updates. Since the underlying system handles the resource lookup the control program will remain the same indenpendently of where it is executed. Also if additional heaters or sensors are built into the oven, no changes are necessary for neither control program nor sensor devices.

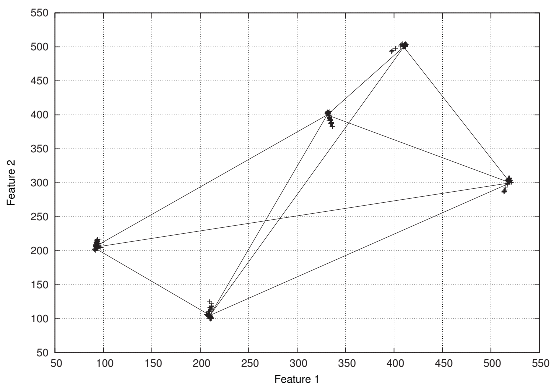
## 6 Simulations

We conducted extensive simulations to demonstrate the stability of our algorithms. To assist real world applicability our simulations are based on realistic network and failure models [9]. We do not assume synchronised clocks at the peers and the simulation is capable to handle failure situations like failing peers and message loss. In order to visualise the simulation results, we simulated peers with each having one resource with two features. Both static features, i.e., having a constant value and dynamic features, i.e., a value changing over time were simulated whereas dynamic features are based on the test signals shown in Figure 6a.

Limited to static features, the algorithm showed fast convergence and stable routing tables after only a few iterations. More interestingly were simulations with dynamic features. Figure 6b shows a snapshot of a simulation with dynamic features after 30 iterations and a cluster radius of 15 units. For this simulation peers where assigned to five feature combinations: signal1:signal2, signal2:signal1, signal3:signal4, signal4:signal5, signal5:signal3.



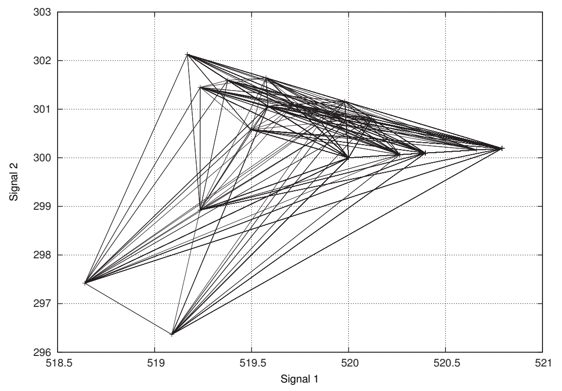
(a) Test signals



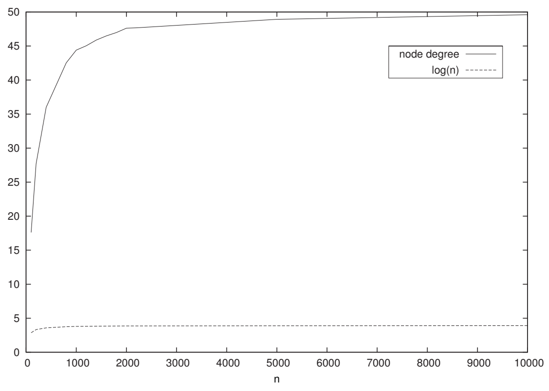
(b) Network configuration after k=30 iterations

**Figure 6:** Simulation results for n=400 peers

As becomes evident in the figure, peers cluster around the five centroids and connections between clusters have been formed. There are slight deviations in the proximity to the centroids due to the unsynchronised clocks and randomised initial feature readings. As can be seen in the figure, the clusters have not fully stabilised with some peers located outside the cluster radius.



(a) Highly connected cluster



(b) Node degree in correspondence to the number of peers

**Figure 7:** Simulation results from n=100 to 10000 peers

Figure 7a provides a closer look on a cluster after 50 iterations. The cluster is fully stabilised and peers within a cluster are highly connected. The average number of connections per peer within a cluster is 36.725 with almost equal cluster sizes of around 80 in the case for n=400 peers. Figure 7b plots the average node degree in correspondence to the total number of simulated peers. The average number of connections per peer grows larger than $log(n)$ which yields, according to Erdös-Reyni, connectedness of the graph with high probability. In general it is possible that clusters cannot stabilise since service descriptions are not correlated. In this situation, the network will be highly connected because each peer acts as bridge to all other peers.

## 7 Conclusion

We presented a light-weight method and architecture for declarative resource discovery in distributed automation systems. Our approach differs from other content addressable network approaches e.g. [16] in that it is less complex and hence easier to stabilise. The simplicity of our method alleviates implementation on resource constrained embedded devices. The approach is particularly suited to the automation domain. It benefits from the static environment and the, in comparison to the Internet, small number of nodes. Clearly, using the mean as correlation function will work only for simple correlations. In more complex and dynamic scenarios other methods might be more appropriate. However, connectedness of the overlay is guaranteed due to inter-cluster connection. The k-means clustering is highly efficient for static scenarios where features describe the static capabilites of the device. Already in the static case the late-binding capabilites of the described method become effective and hence have influence on engineering complexity.

Based on the simulations conducted, we are convinced that our method is well suited for multidimensional resource lookup in modern automation systems. The declarative approach will simplify the engineering process while at the same time increase robustness and enable automation systems to react adaptively to reconfigurations and failures. The approach is not disruptive but allows smooth migration from existing systems towards the more flexible solution as devices can be gradually upgraded. Benefits become effective from the second device on. New is the initial phase where the system collects autonomously all relevant information required for operation. Formally this process was conducted manually by an engineer or an engineering tool at design time. Once initialised the system can function as before with the addition of continuous but low bandwidth monitoring and optimisation of the overlay network.

## 8 References

[1] A. Andrzejak and Z. Xu. Scalable, efficient range queries for grid information services. In *In: Proc. of IEEE Conf. on Peer-to-Peer Computing*, pages 33–40, 2002.

[2] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Proximity neighbor selection in tree-based structured peer-to-peer overlays. Technical Report MSR-TR-2003-52, Microsoft Research, 2003.

[3] P. P.-S. S. Chen. The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[4] C. J. Date. A critique of the SQL database language. *SIGMOD Rec.*, 14(3):8–54, 1984.

[5] D. Drinjakovic and U. Epple. Search methods in p2p-networks of process control systems. *Industrial Informatics, 2004. INDIN '04. 2004 2nd IEEE International Conference on*, pages 101–107, June 2004.

[6] H. Evans and P. Dickman. Peer-to-peer programming with teaq. *Revised Papers from the NETWORKING 2002 Workshops on Web Engineering and Peer-to-Peer Computing*, 2002.

[7] I. Foster and C. Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure (The Morgan Kaufmann Series in Computer Architecture and Design).* Morgan Kaufmann, November 2003.

[8] P. Ganesan and H. Garcia-Molina. Efficient queries in peer-to-peer systems. *IEEE Data Eng. Bull.*, 28(1):48–54, 2005.

[9] K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: estimating latency between arbitrary internet end hosts. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 5–18, New York, NY, USA, 2002. ACM.

[10] Herault, Lemarinier, Peres, Pilard, and Baeuquier. Self stabilizing spanning tree algorithm for large scale systems. Technical Report 1457, LABORATOIRE DE RECHERCHE EN INFORMATIQUE, August 2006.

[11] Holonic Manufacturing Consortium. Holonic manufacturing systems overview. Online, 2002.

[12] R. Huebsch, B. Chun, J. Hellerstein, B. Loo, P. Maniatis, T. Roscoe, S. Shenker, I. Stoica, and A. Yumerefendi. The architecture of pier: an internet-scale query processor, 2005.

[13] R. W. Lewis. *Modelling Distributed Control Systems Using IEC 61499.* Institution of Electrical Engineers, Stevenage, UK, UK, 2001.

[14] M. G. Mehrabi, A. G. Ulsoy, and Y. Koren. Reconfigurable manufacturing systems: Key to future manufacturing. *Journal of Intelligent Manufacturing*, 11 (4):403–419, 2000.

[15] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM.

[16] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content addressable network. Technical Report TR-00-010, Intel Research Berkeley, New York, NY, USA, 2001.

[17] S. Ratnasamy, J. M. Hellerstein, and S. Shenker. Range queries over dhts sylvia ratnasamy,. Technical Report IRB-TR-03-009, Intel Research Berkeley, June 2003.

[18] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM.

[19] D. Stutzbach and R. Rejaie. Towards a better understanding of churn in peer-to-peer networks. Technical

report, Department of computer science, University of Oregon, November 2004.

[20] R. Wolff and A. Schuster. Association rule mining in peer-to-peer systems. *Systems, Man, and Cybernetics, Part B, IEEE Transactions on*, 34(6):2426–2438, Dec. 2004.

[21] C. Zhang, A. Krishnamurthy, and O. Y. Wang. Brushwood: Distributed trees in peer-to-peer systems. In *In Proceedings of the 4th International Workshop on Peer-to-Peer Systems (IPTPSÕ05*, pages 47–57, 2005.