Vertical Information Integration for Cross Enterprise Business Processes in the Energy Domain

Christoph Gerdes[1], Udo Bartlang[1], and Jörg P. Müller[2]

[1] Siemens AG, Corporate Technology, Information and Communications, Munich, Germany
[2] Clausthal University of Technology, Germany

Abstract. The continuing growth and decentralization of power networks creates immense interoperability and integration challenges for ICT systems performing control and coordination tasks. On the one hand, large amounts of data coming from low-level field and automation systems need to be interpreted, aggregated, and made available to the business information systems to inform local decisions within power generation companies and operators; on the other hand, this *vertical integration* needs to be complemented by and connected to a *horizontal integration* capability which links the internal information and process models of different players in the market in (soft) real time, to inform and enable cross-enterprise coordination and optimization.
In this paper, we introduce an integration architecture that supports and combines vertical as well as horizontal integration, and methods to improve interoperability and increase automation of cross enterprise business processes in the energy domain. We support vertical integration by means of a novel decentral information aggregation and routing platform to manage large-scale data intensive systems. Horizontal integration is enabled by a peer-to-peer content (document) repository which is used to coordinate and integrate local processes in a cross-organizational fashion, thus allowing, e.g., reliable and timely provision of electricity outage reports and preparation of coordinated action plans. We evaluate important properties of these methods and report experimental results.

Key words: Peer-to-Peer, vertical integration, content repository, business process management

1 Introduction

Increasing demand for electrical energy has been leading to continuous extension of power grids world-wide. In Europe, national grids were joined to a synchronously operated AC grid. This union for the co-ordination of transmission of electricity (UCTE) provides electrical energy for more than 500 million consumers. This network enabled larger and more efficient power plants on the one hand and a maximum of reliability and availability on the other hand. Recent developments like de-central generation and de-regulation efforts, however, lead

to rapid increase of load on electricity infrastructures. In consequence, large-scale blackouts like in August 2004 in the USA, on the Swiss-Italian border in 2003, and in the Weser-Ems area in Germany in 2006 endanger reliable electricity delivery. The latter left millions of households throughout Europe without electricity and resulted from lack of communication between transmission system operators (TSO). Initially a high voltage line was shut down over the Weser-Ems channel in Germany which caused a cascading effect — thereby overloading other lines until major parts of the European grid where separated. Efforts taken by individual TSOs to stabilize the network failed as they had no information on control actions taken by their colleagues in neighboring transmission areas yielding continuous aggravation until total disruption of service. In later analysis[1] the European commission emphasizes the importance to:

- Accelerate the adoption, in the context of a new Community mechanism and structure, of essential common binding network security standards;
- Enhance the coordination between transmission system operators to ensure an effective real-time operation of the European grid; efforts should be made to have a gradual evolution towards regional system operators; this should require effective unbundling as discussed in the Commission Strategic Energy Review;
- Improve investments in the European grid both to ensure its reliability and the construction of a truly competitive European market.

Already in 1999, the European transmission systems operators organization (ETSO) was founded to harmonize and develop the European electricity market. ETSO members include UCTE, the association of TSOs in Ireland (TSOI), the United Kingdom TSO association (UKTSOA), and the association of the nordic TSOs (NORDEL). ETSO takes care of cross TSO data exchange and standardization. ETSO also standardizes cross TSO processes like imbalance settlement, reserve resource planning, and outage transmission.

Illustrated by using the application scenario of electricity outage management as described by ETSO, the first contribution of this paper is a conceptual integration architecture for interoperability and automation of cross enterprise business processes in the energy domain. The architecture combines a *vertical* dimension, which supports the flow of information and coordination between low-level automation systems and enterprise systems, with a *horizontal* dimension, which links the internal information and process models of different players in the market, and enables cross-enterprise coordination.

The second, more technical contribution of the paper is that it illustrates two important functional building blocks to support successful integration:

- A decentrally organized information aggregation and routing platform to manage large-scale data intensive systems. Providing an interface of declarative querying and programmable data sources it abstracts from the underlying physics; also, it can be extended to include assets unknown during design time as well add new functions for situation based analytics. Simulations show the

[1] Cf. `http://europa.eu/rapid/pressReleasesAction.do?reference=IP/07/110`

performance of the indexing group which scales well to 100000 peers and beyond. From a business perspective, this approach can increase the efficiency of today's power networks as well as other industrial applications that rely on large, globally distributed networks with thousands of nodes.
– A peer-to-peer based content repository to substitute the centralized implementation of the ETSOVista platform's back-end system avoids a single point of failure in critical situations. From a technical point of view, the proposed system is able to benefit distributed collaboration in outage management processes by supporting a publish-subscribe mechanism to enable the rapid notification of critical events to interested parties. The evaluation shows that the system is robust against document losses and achieves good performance regarding document access. From a business point of view, the decentralized solution shows the potential to avoid vendor lock-in situations resulting from a proprietary market information aggregator.

A word on terminology: For our work, architectures and mechanisms for decentralized resource management, organization and coordination are at the heart of both multiagent systems and peer-to-peer computing. In this paper, we use the two terms often interchangeably. For a more detailed and more educated discussion of properties, commonalities, and differences between the two concepts, we refer to [1].

The paper is structured as follows: Section 2 provides the application background and describes the electricity outage scenario. In Section 3, the overall integration architecture is presented, followed by the definition of the technical building blocks: Information aggregation and routing (Section 4) and the distributed content repository (Section 5). Experimental results evaluating our approach are given in Section 6. The paper ends with a discussion of related work in Section 7 and a summary and outlook in Section 8.

2 Application Scenario

In this section, we present the application context of our work. We indicate the overall view of vertical and horizontal integration in distributed power networks. Subsequently, we describe a particular scenario to serve as a use case to illustrate our research: the distributed management of electricity outages.

2.1 Vertical and Horizontal Integration in Distributed Power Networks

Figure 1 illustrates the typical roles and relationships in a distributed power network. Generation companies (GenCos) operate power plants; they can be located at different levels of the physical energy network, ranging from supergrid (e.g., large coal power plants or nuclear plants) over high-voltage grid (e.g., industrial powerplants) to medium- and low-voltage grids (wind or solar parks down to private solar installations).

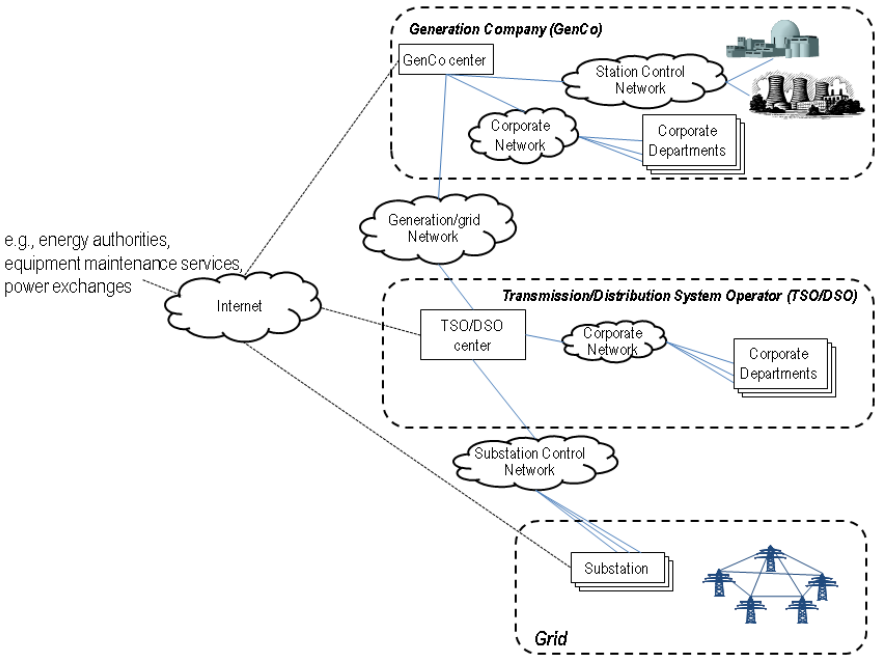


Fig. 1. Power Network Topology (inspired by [2])

The control center of a GenCo is responsible for coordinating two levels of activities, which in the sequel we shall call *vertical integration* and *horizontal integration*: firstly, using supervisory control and automation (SCADA) systems the GenCo operating center remotely controls the equipments located in the generation stations, with which it is normally connected via a dedicated plant control network to guarantee real-time control. These activities depend on field information and events coming from a multitude of sensors, devices, plants, and substations to be filtered, aggregated, and visualized. The process of gathering, aggregating, routing, and interpreting this information within an enterprise control system is called *vertical integration*.

Secondly, the information gathered using vertical integration is used as a basis for planning, enacting, monitoring, and coordinating both internal and cross-organizational business processes. Of particular focus for this paper are business processes involving cross-organizational interaction and coordination with the partners in the energy network, and, in particular the transmission system operators (TSOs), and the distribution system operators (DSOs). We call this type of activities *horizontal integration*. One example of this is the process of coordinating activities between a GenCo, TSOs, and DSOs in case of an outage. Vertical integration is used to determine type and extent of a local outage at the GenCo. In dealing with a local outage situation, the resulting information will inform decisions made in the business processes that guide coordination

between the GenCo and its partners to prevent the effects of the local outage from spreading and causing malfunction in other parts of the network.

2.2 Use Case: Distributed Outage Management

In general, the propagation of an outage (document) involves the three principal actors. The first actor is a system operator (TSOs/DSOs) who has a complete overview of the tie line maintenance and operation and is in a position to provide a coherent picture of the situation at a given instance in time. The second actor is some market information aggregator, who may be a commercial entity that simply specializes in providing electricity market information and who makes the information available to the public. Such a provider may also make the information available to a selected distribution list as an additional service. Finally, the third actor is an information receiver or interested market participant who wishes to obtain such information.

The ETSOVista data platform is a service provided by ETSO aiming to facilitate access to information for all market participants and stake holders. Founded in 2006, it supports the publication of all information on current situation of the electricity market. A state of the art approach is to implement ETSOVista as a centralized web-based application with interfaces to aggregate and visualize respective data. For example, system operators can use ETSOVista's standardized interfaces to provide detailed information concerning their area of responsibility: in this context, a critical issue of availability is the knowledge of the outages.

In addition, ETSO has standardized an electronic document that system operators may use to transmit information about outages to an market information aggregator such as ETSOVista. An outage document is issued by a system operator and refers to information for generation over 100 MW network lines which have influence on the offered capacity. Outage events can be either planned, i.e., planned shutdown of an asset, or unplanned, i.e., the forced shutdown due to failure or other emergency situations. In the current release, the outage document is limited to outages of tie lines and network interconnections. Future extension to other outage object types is intended.

For example, the outage information publication process can be broken down to three sub-processes, namely (i) outage information creation, (ii) outage information modification, and (iii) outage information deletion:

- *Outage Information Creation:* whenever an outage situation occurs (either forced or planned) a system operator sends the information to the ETSOVista platform. The platform validates the conformity of the information received. If the information is incorrect the platform ignores the transmission and logs the incident. If the information is correct the platform shall enter it in its persistent data storage and shall publish it.
- *Outage information modification:* an outage situation may be modified to indicate its progress or to correct any data that is found to be invalid. Accordingly, the following possibilities exist:

  - For an outage the following information may be revised: start date, start time, end date, end time, and affected interconnector.
  - For an outage the following information may be added: start time, end date, end time, and affected interconnector.
  - If an affected asset is incorrect or the planned maintenance is canceled prior to the start time, the outage in question has to be deleted and eventually new outage information has to be provided.
- *Outage information deletion:* a given outage may be deleted with an update that makes use of the *Delete* attribute. This has the effect of deleting the outage from the published list.

3 Overall Architecture

In this section, we show the overall view of the used ICT architecture. In addition, we indicate challenges to achieve vertical and horizontal integration, and consequently state our technical approach.

3.1 Overall view of the ICT architecture

In this paper, we propose a decentralized architecture to support vertical and horizontal integration, as well as the coupling between vertical and horizontal activities. Figure 2 gives an exemplary view on the IT architecture of a distributed energy system including one ore multiple GenCos as well as TSO/DSOs. It also illustrates the two integration dimensions considered in this paper.

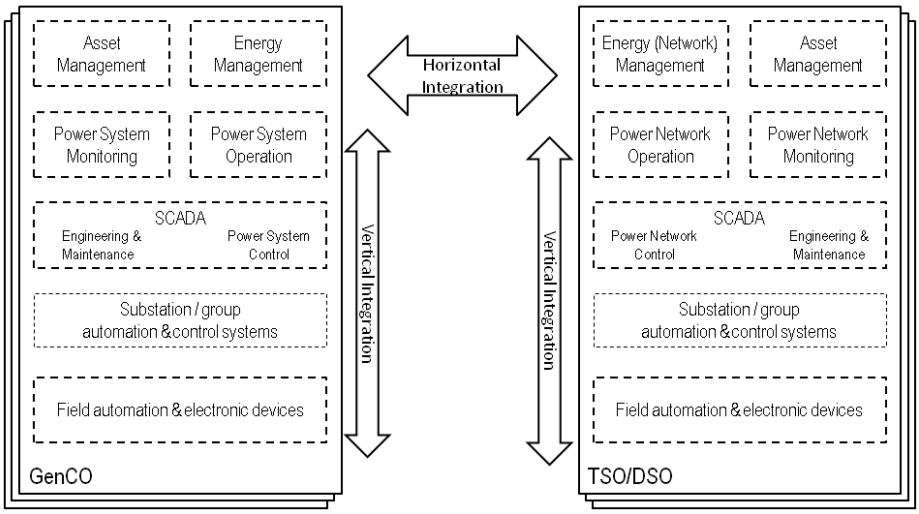


Fig. 2. ICT Architecture of a Distributed Energy System

The uppermost layer is the business systems layer including energy management, asset management as well as standard ERP systems. The ICT systems at this layer are instrumental in (internal and cross-enterprise) business process enactment; they need to be supported by appropriate business process models. They receive real-time decision-relevant information from the systems for power system/network monitoring and operation. These systems again rely on different types of supervisory control and data acquisition systems (SCADA), which collect and aggregate data from local field automation and control systems controlling individual power groups, subnetworks, or substations. At the bottom end of this information food-chain are numerous electronic devices and sensors producing large amounts of operation data.

Also note, that horizontal integration will take place in two interrelated ways: on the one hand, different types of business information systems supporting different business functions within a company (such as, e.g., asset management, energy management, and other ERP functions) need to be coordinated and integrated to optimize the overall performance of each individual company. On the other hand, horizontal integration entails the cross-enterprise coordination of business processes such as, e.g., dealing with an outage.

3.2 Challenges

Even considered in isolation, vertical integration raises a number of difficult challenges. Large amounts of data need to be handled, correlated and abstracted, routed upwards, interpreted and visualized to allow operators to obtain an up-to-date view of the status and performance of the plants/networks. The sources producing this data are largely distributed - due to the large data load, centralized processing is not feasible. Also, there are *interoperability barriers* at different levels. While there are a number of upcoming standards in the energy sector[2], there is still a certain heterogeneity of systems and interfaces, which makes the vertical information a challenging task. The same can be said for horizontal integration, where interoperability needs to be provided across business information systems and business process models of different companies, to enable timely and coordinated reaction to critical events.

In addition, due to changing regulations, market situations and technology innovations, the architecture displayed in Figure 2 needs to be open to change. For example, a regulatory change (e.g., new laws) may require modifications to be made to the internal or cross-enterprise business processes, which again may impact lower layers of the architecture. Also, new communication standards or middleware technologies may provide new, more efficient ways to realize the different layers of automation and control systems, which may affect the upper layers of the architecture. Ideally, our distributed energy system should be easily adapted to these changes at different levels.

[2] We refer, e.g., to the IEC 61850 standard series for communication of energy automation systems, where a common information model is being developed.

3.3 Technical Approach

To combine the requirements to deal with large amounts of data at different levels, to connect (locally autonomous) technical systems with business processes, and to provide internal and cross-enterprise interoperability and flexibility in the face of change, our overall research work combines a number of technology ingredients covering both design-time and run-time aspects:

- Design-time:
  - We provide a multiagent-based architecture to describe roles and capabilities of the parties in the distributed energy system, and to model (and subsequently enact) cross-enterprise coordination processes.
  - We use a model-driven system engineering approach (see, e.g., [3]) to enable the maintenance of process, service, and information models at different levels, and the top-down and bottom-up synchronization and adaptation between different layers based on model transformations.
- Run-time:
  - We support vertical integration by an information aggregation and routing approach based on a structured peer-to-peer framework (see, e.g., [4], [5]), enabling the decentralized management of ad-hoc queries as well as publish-subscribe style communication
  - We take a document-centric approach to horizontal integration by proposing a peer-to-peer content repository (see [6]) allowing different parties to publish, modify, version, and synchronize documents (e.g., outage reports), to subscribe and search for certain types of documents, and to route relevant documents to interested parties using content-based rules. This approach benefits both scalability and reliability, and ensures consistency of concurrent operations.

In this paper, we shall focus on the run-time aspects of our approach. In particular, in Section 4 we present a vertical integration approach based on a decentralized information aggregation and routing framework. In Section 5, we complement this approach by describing the architecture and methods of a peer-to-peer based distributed content repository to enable document-centric horizontal integration processes.

4 Information Aggregation and Routing

The state of the power grid is recorded by a large number of digital devices distributed over the entire supply territory. Sensing equipment includes intelligent electronic devices (IED), smart sensors and meters. In the context of vertical information integration, i.e., integration of field data with the business processes of the system operator, the digital devices act as data sources which provide data in a specified quantity and quality. In correlation with the kind of data source, the measured value its validity and the sampling interval, high volume measurement streams may occur. To make these streams usable in an operation context

the raw data needs to be processed to information which is representative of the current network state. In traditional power grids, there exists a well defined power flow from few generators down to a large number of consumers. This allows operators to estimate the network state accurately without comprehensive measurements. In scenarios such as decentralized generation, power flows might reverse causing unforseen dynamics and failure situations which cannot be fully predicted by the system operator. Hence, in order to identify critical situations, timely collection and processing of high resolution data becomes mandatory. In this situation, in-network approaches are advisable as they limit the load on the communication infrastructure and yet yield a high quality of service (QoS) and fine granularity of measurements.

In this section, we introduce a data-centric perspective on multiple sensing equipment providing a unified access paradigm on distributed measurement data. Based on the standard query language (SQL), we developed a programming language to formally express an information interest and appropriate processing functions. Using declarative queries, measurement streams can be routed through the network, thereby being processed and aggregated at dynamically determined processing nodes. In the following, the integration architecture, query language and query execution subsystem are introduced. A set of examples show the capabilities of each component.

4.1 Vertical Integration Architecture

The foundation for information aggregation and routing is constituted by the query execution architecture. The architecture links all nodes of the network in an application layer overlay thereby providing a global address space for all assets in the network. The overlay manages the mapping from virtual peer addresses to physical hardware addresses thereby virtualizing network topologies and technology. The mapping supports both keywords as well as complex declarative queries. Keyword mappings are realized through a distributed hashtable (DHT) spanned over all network devices (Figure 3). In order to support complex queries, a subsection of the address space is dedicated to the so called index cloud. The cloud consists of dedicated index peers which manage device meta data and a query catalog. Additionally, data sources host a local query engine which is capable of processing and optimizing queries as described in Section 4.3. Power grid equipment has lifecycles of often up to 50 or more years hence yielding systems with a large variety of hardware with differing compute resources, communication standards and protocols.

Traditionally, all field data is transferred to a central service, however, with the limitations detailed in the above sections. The integration architecture introduced in this paper can achieve higher efficiency by pushing query engines into the network. This, however, must be either directly supported by the data sources other integration methods must be deployed. Our architecture supports three levels of integration: for data sources with sufficient compute resources, the query engine can be directly hosted by the data source. Data sources which do not provide the required resources to support a query engine by themselves

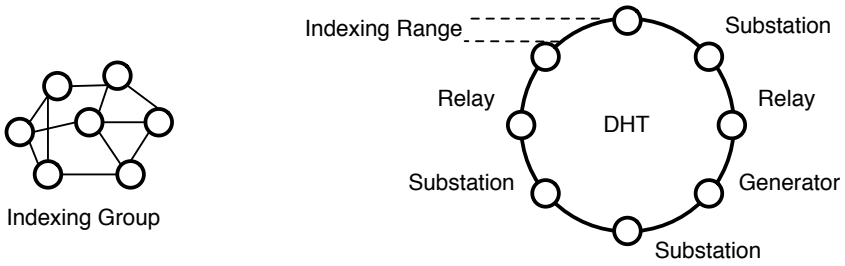


Fig. 3. Vertical Integration Architecture

but have a device in their proximity are integrated by the neighboring query engine acting a as proxy reading individual measurements from the device, and processing them in the query context. Finally, data sources which do not support a common communication standard require an additional hardware component which acts as gateway to read, process, and forward data in the query context.

Device meta data provides information on device capabilities and attributes along with a device locator. Hence devices can be addressed based on its descriptions (e.g., "send a message to a device which can measure frequency"). The query catalog maintains all globally visible queries currently in execution. This information is used to optimize query execution and stream routing. Both concepts in combination enable locating information sources and routing their output to respective processing sinks (e.g., "send measurements from devices that can measure frequency to a device that can archive data"). To formally specify query and routing information, the following section introduces a query language especially designed for this purpose.

4.2 Query Language

The query language supports both snapshot queries (single result set) as well as queries initiating data streams. It allows us to apply filters on streams and to route streams to different nodes throughout the network. Despite known restrictions and flaws of SQL [7], we chose it due to the wide acceptance among developers and good integration with standard applications, such as JDBC. However, to meet the high performance requirements, query statements are compiled to an efficient encoding; as soon as they enter the network only this encoding is used. This improves execution performance and decreases network traffic. Given semantic equivalence, different query languages can be supported by adjusting the compiler. Representing the query, the compilation itself is a sequence of instructions, i.e., basic operations the query engine can perform to retrieve the query result. The query engine analyzes the instructions sequence and creates an execution plan as described in Section 4.3.

Figure 4 provides an example query. It demonstrates an assignment specified through the right arrow, a standard *SELECT ... WHERE* clause and a

```
ALARMS@ -> SELECT * FROM MMXU WHERE Hz < 59.001
    WINDOW(now, forever, 1s)
    REPLICATION=3
    RESTRICT TO GenCo
    TRACEABILITY=ON
    RECEIVER(rcv_node)
```

Fig. 4. Example of how to retrieve and route information within the platform

set of QoS constraints. In the example the result set is assigned to a variable named *ALARMS*. Thereby the @ operator specifies where the variable is hosted. *ALARMS@* means the variable is globally visible and located in the indexing cloud. *ALARMS@node1*, on the contrary would allocate a variable at node1. 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. The window parameter implicity controls response time and throughput and is one of the key QoS parameters. In other words it advises the query engine to retrieve and deliver results sets timely as specified. To achieve this goal the query engine coordinates all in the query participating data sources and schedules their network and memory resource utilization. If this process of allocating resources fails, e.g., because of resource shortage or other unavailability, the query will not be scheduled. The *REPLICATION* parameter controls availability as it specifies the number of copies the system keeps of the result set. A replication factor of 3, means that the result set is replicated to three other nodes in the index cloud. Thereby each replica inherits all other QoS parameters as specified in the query. *RESTRICT TO* in one of the parameters to enforce security during query execution. In the example *RESTRICT TO* restricts access to the result set and all intermediate results which may be stored during query execution to the user or group identified by *GenCo*. Finally *TRACEABILITY* adds an additional attribute to the result set containing information on all data sources that participated in the query and each operation, i.e., individual execution plans, that were executed in the process. This enables precise analysis and verifiability of the information execution process. Finally the *RECEIVER* parameter specifies an receiver of the result set. Thus it can be used to route a data stream to one or more receivers in the network. The parameter does not limit to a single receiver e.g. *RECEIVER(SELECT * FROM NODES WHERE type='Archive')* would forward the stream to all nodes which are of type *'Archive'*.

Besides specification of queries the language supports imperative elements and basic operators found in many other programming languages. Hence, the language and its execution architecture make the network programmable enabling to define data processing filter functionality like the simple PID controller in Program 1. Thus, instead of pulling high volume data streams to a single location, programs can be defined which process data in-network close to the data sources. Aggregated key indicators can then be calculated and sent to other nodes for further processing, stored in archives or visualized at control centers.

This enables high performance applications such as high resolution monitoring of the network state. In [8] we introduced a special operator called MON_k which is capable of monitoring a large number of assets including detection of anomalies. Depending on the targeted accuracy and responsiveness of the detection, the MON_k operator causes intensive message exchange between involved assets. However, since communication is peer-to-peer and in-network, the operator scales well to a high number of network nodes.

```
FUNCTION PID(Pv, Sp)
  Kp -> 100
  Ki -> 0.9
  Kd -> 1000

  Error -> Sp - Pv
  @TotalError -> TotalError + Error

  Pgain -> Kp * Error
  Igain -> Ki * TotalError
  Dgain -> Kd * (Error - @Derror)
  @Derror -> Dgain

  return Pgain + Igain + Dgain
END
```

Program 1: Basic PID controller

4.3 Query Execution

The process of retrieving information specified through a complex query statement is non trivial in large-scale distributed systems. We apply a five step process for query execution starting with an analysis of the query statement, followed by locating respective data sources, building a distributed data structure which coordinates individual data sources to deliver the requested content, merging and filtering individual data items and finally generating a result set.

Once injected into the network, queries are compiled to binary form and loaded in a local query engine. An execution plan (e.g. Figure 5) is generated and scheduled for processing. The plan lists all actions necessary to deliver the requested result set. The plan depicted in Figure 5, for example, first checks locally if the query can be evaluated with local information alone. If so query execution is complete. Otherwise the query is optimized and rewritten for distributed execution. Query optimization is a complex procedure where the query is restructured to reflect the overlay topology, the specified QoS parameters 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 and is able to allocate local resources to meet

to constraints specified in the query. Subsequently, the query as a whole or in part is assigned to the peer accordingly. Intermediate results due to local computation or obtained through subquery assignment are stored in the local storage. The process continues until all requested information is contained in the local storage. During execution a number of events may cause query processing to fail. Without violating the QoS constrains the query engine tries to compensate failures by reissuing actions or assigning new subqueries an alternative sources. In case the maximum number of retries is reached, a timeout occurred, or the query is not computable, an exception is raised and the execution ends freeing all allocated resources.

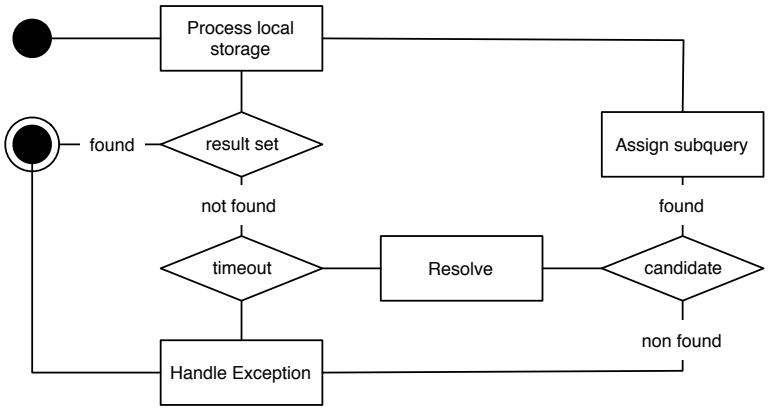


Fig. 5. Query execution plan

Deallocation is highly complex in cases of failures of complex queries with large numbers of peers involved since individual assets may be unreachable. To ensure appropriate deallocation of resources all variables in the query engines are temporary only. Variables reaching a defined age in a query engine's memory are automatically garbage collected. To persistently store results they must be forwarded to entities providing archiving functionality.

The vertical integration architecture provides a flexible high performance platform to manage large-scale data intensive systems. Based on peer-to-peer network virtualization and through providing an interface of declarative querying, it abstracts from the complexity of the heterogeneous and dynamic system. Since the platform is programable it can be extended to include assets unknown during design time and implement new functions to introduce new methods of analytics. Having unified access from high level IT systems down to individual automation equipment is the first key component to enable fully automated cross enterprise business processes. In the following sections a horizontal integration component is introduced which enables the flexible interlinking of individual business players.

5 Horizontal Integration Using a P2P-based Content Repository

The ETSOVista transparency platform's main objective is to provide to the electricity market participants all the relevant, valued, and trusted information in a transparent way. That is, system operators shall use standardized electronic documents to transmit information about outages to the platform for diffusion to the market. As amongst the principle criteria of availability is the knowledge of the outages, we propose to substitute the centralized implementation of the platform's back-end system by a peer-to-peer based content repository [6] avoiding a single point of failure. In addition, the approach supports a publish-subscribe mechanism to enable the rapid notification of critical events to interested parties.

In the following, we present the overall architecture and exemplified methods of a peer-to-peer based, decentralized content repository to enable document-centric horizontal integration processes.

5.1 Modular Architecture of a P2P-based Content Repository

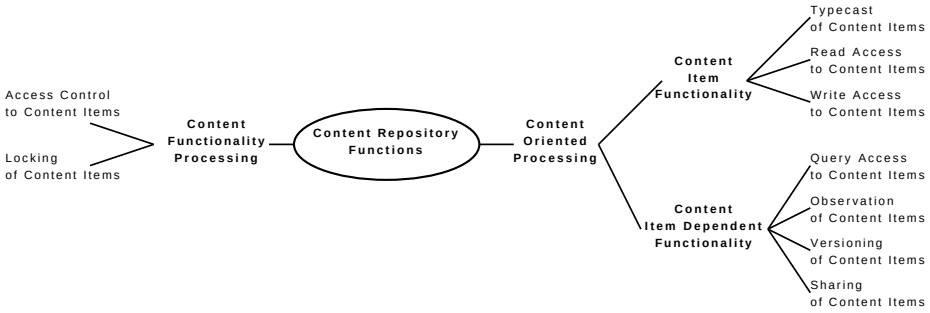


Fig. 6. Functional Components of the Content Repository

The content repository enables the management of both structured and unstructured content: its repository model defines the meta model to identify and structure content data on a logical level—from a user's point of view; it supports to express functional operations on content data. The concrete implementation translates these operations into actual corresponding actions affecting its used (peer-to-peer) storage subsystems. The repository model adopts the *Content Repository API for Java* (JCR) [9], which is defined as open standard to improve application interoperability:

– The repository model offers a generic, hierarchical content data model and several levels of functionality for content services on a logical level: a repository consists of an unlimited set of named *workspaces*; each workspace establishes a single-rooted, virtually hierarchical, *n-ary* tree-based view of content *items*. E.g., a peer-to-peer workspace supports a decentralized document exchange.

– In addition to the basic repository model, a content repository is constituted by a set of essential functional building blocks, as illustrated in Figure 6.
– Documents may be accessed using either *direct* or *traversal* access. Each outage document can be uniquely identified to ease direct access. Consequently, it is independently addressable from its position within the workspace hierarchy. The traversal item access targets on walking through the content tree of a workspace, step by step, using (relative) paths. To support the verification of access rights, each document is allocated an unique identification of its sender and optionally (multiple) receiver identification(s) to restrict read access.

Figure 7 shows the modular content repository approach considering horizontal and vertical system decomposition [6]: on the one hand, the distribution degree of content repository functionality regarding the persistent storage support may vary—for example, the storage management for local or distributed workspaces. On the other hand, different modules, for instance, are responsible for different management tasks. Each of the architecture's layers is briefly introduced and discussed in the following:

The *content application layer* offers the *content repository* API. It hides sub-layers to free users to deal with peculiarities of content storage.

The *content repository layer* represents the mapping of the logical repository model to corresponding system modules; at its core, it implements several registries and managers. For example, the *session* subsystem basically uses a *transient item state manager* to cope with a content item's transient state per session. The *workspace* subsystem uses several managers to deal with the repositories functional building blocks. It is responsible to create consistent items in persistent storage. For instance, a *query manager* is used to support query access to content items, a *version manager* is used to support versioning of content items, and an *observation manager* is used to support observations of content item changes. Such manager use the *persistent item state manager* to actually obtain read and write access to a workspace's content items—that is, to obtain an actual content item view of persisted data; it represents the connection between the workspace scope and the used persistent storage back-end subsystem; a persistent item state manager shall trigger the observation mechanism if interests in corresponding item changes exist—usually, reflected by some *access manager* of the *persistent storage layer*. This shall enable an observation manager to asynchronously subscribe for changes in a workspace. An *item access manager* uses a persistent item state manager to enforce its functions, that is, the support of access control for content items. Accordingly, the item persistent item state manager needs to obey such enforced restrictions.

The *policy layer* administrates the scope of different storage policies that may be used by the content repository layer to actually access the persistent storage layer. Therefore, it uses *policy managers* matching corresponding *access managers* of the persistent storage layer. As illustration, the usage of a peer-to-peer policy manager enables the definition of potentially fine-granular policies at peer-to-peer data level—rather than on item level. Thus, each type of content or rather content instance may have its own policy; some examples of storage

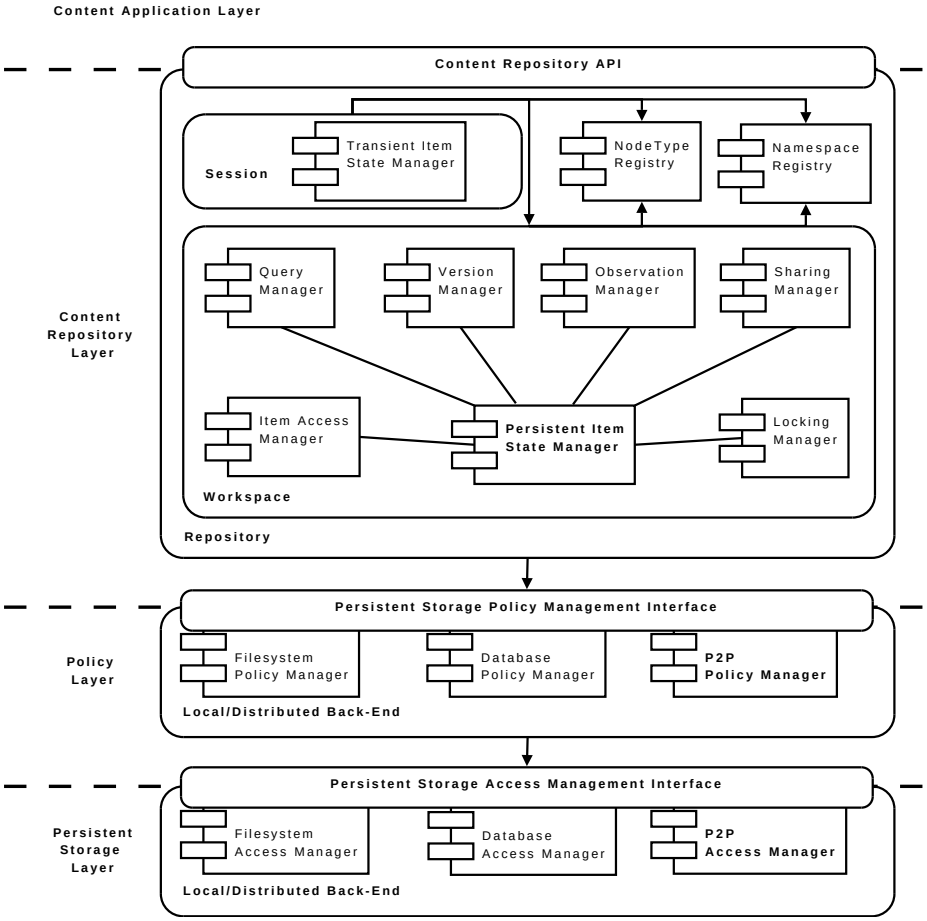


Fig. 7. Layered Architecture of the Modular Content Repository Decomposition

policies in peer-to-peer case may include the life of content, that is, if content shall be stored infinitely or temporarily.

The *persistent storage layer* defines the subsystem to deal with local or distributed persistent storage at data level. It is indirectly usable by the *persistent item state manager* of the *content repository layer* by exposing a generic *persistent storage access management interface*. Using this interface, several *access managers* for persistent storage may be used, for example, the *peer-to-peer access manager*. Such peer-to-peer access manager supports a mapping between a workspace view of content at item level and a raw data view at back-end storage level; thus, it is necessary to use some interpreter to recognize raw data as content items, that is, to retrieve item semantic from raw data resources.

Figure 8 illustrates the applied *naming concept* regarding the transformation process of a repository item (between logical and physical objects): (i) the *content application layer* actually deals with item *objects* existing in transient

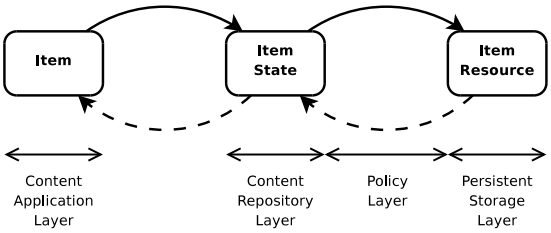


Fig. 8. Transformation Process of a Repository Item

storage. (ii) The *content repository layer* has a more sophisticated view on items: the layer knows the internal state of an item as it is responsible to manage core repository functionality. However, at content repository level, an item and its represented content forms one logical unit. (iii) An item which shall be persisted needs to be transformed from its *item state* to a lower-level *item resource* object. Such a transformation process involves the *policy layer* to specify a policy for the corresponding *access manager* at *persistent storage level*. The *item resource* object resulting from this transformation shall reflect all necessary information to represent and reconstruct its item state. In addition, it contains policy information. An *item resource* deals with all the low-level details of actual data storage—which is transparent to the content repository layer.

As a result, the content repository system is able to support different content storage policies in a flexible way. For example, such policies may reflect how content may be actually persisted and accessed. It supports different granularity level hierarchies to be build by grouping aggregations of objects to represent larger objects (collections); regarding the granularity level, data objects may be restructured and build from atomic values on demand. Thus, content data must adhere to some global uniform semantics to deal with and ease content integration—specified by storage policies. For example, such concept facilitates the existence of multiple object copies but may hide their actual physical locations.

5.2 P2P-based Methods for Persistent Storage Management

As most important feature, the peer-to-peer based content repository supports methods for *fault-tolerant* and *consistent* content management: as, once an outage document content is stored to the system, it shall not be lost. This raises the challenge to coordinate concurrent activity in a dynamic peer-to-peer environment and to protect the consistency of created artifacts to keep content up-to-date across geographically distributed locations. For example, in case of the revision or addition of outage information, the same document identification is used and a new document version is created. This has the effect of canceling the previous transmission and replacing it with the new one.

The system uses *DhtFlex* [10], a distributed algorithm to enable flexible atomic data operations for replicated data at peer-to-peer persistent storage

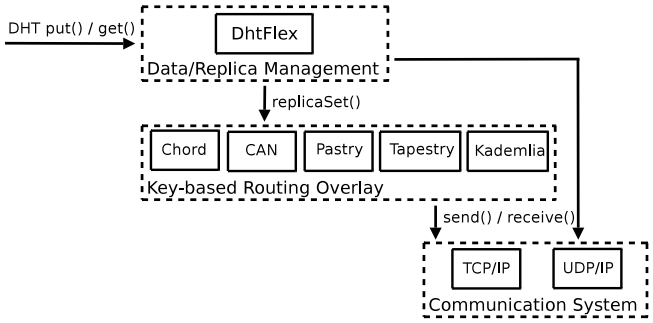


Fig. 9. Major Building Blocks of DhtFlex's System Environment

level—a method being trimmed for a highly concurrent and fluctuating environment. As illustrated in Figure 9 [10], DhtFlex is responsible for the complete data management, including replication handling.

The query model of DhtFlex supports simple DHT *put* and *get* operations for data items that are uniquely identifiable. In addition, techniques are used to extend a DHT in order to deal with the content management requirements. DhtFlex supports optimized operations for both immutable as well as mutable data resources and offers flexible consistency strategies. For example, once a certain version of an outage report is defined, it remains forever unchanged within the corresponding version chain.

Therefore, DhtFlex imposes an annotated data resource concept to typify replicated data. This allows the differentiation between mutable and immutable data items and thus an efficient processing for both data resource types. Especially for mutable data resources, DhtFlex provides strong consistency guarantees enabling atomic DHT *put* and *get* operations. Therefore, it exploits techniques of Leslie Lamport's famous *Paxos* algorithm [11] to coordinate the *recast* process of a data resource's replication group: DhtFlex serializes concurrent *put* and *get* requests over the *master* of a replication group in order to accelerate these operations; in addition, it is able to deal with master failures by automated handovers—ensuring consistency. DhtFlex allows system grows to large scales and updates to be made from anywhere in the system.

For example, the approach enables to implement a close-to-open model by retrieving the latest item resource via a *get* operation once the item should be locally opened and keep it as a cached copy by the *content repository layer* until access is closed. All succeeding requests to an item's potential child items can be satisfied using information from the cached copy. If the item should be modified, the locally cached copy is updated to reflect the changes; hence, *put* efforts and corresponding changes are locally buffered by a *session* before stored to the network in order to minimize local write latencies. Finally, once item access is closed, all cached changes are flushed to the peer-to-peer network and tried to be committed. This scheme works especially well when using versioning, as immutable item resources that store corresponding contents are never removed

from the network; hence, links to certain versions of item resources could not be invalid as they cannot be removed from the system.

The support of *observations* at shallow operational scope relies on the usage of *special* observation resources. Using a subscribe-like feature [6], the basic eventing-notification mechanism can be implemented, which allows the triggering of a notification if a suited content resource for a certain path in the virtual tree of a workspace is stored. This is achieved by placing an observation resource as *subscription* at the corresponding peers, which perform matching tests reacting on the adding, removing, and modifying of affected item resources. The support of deep operational scope requires such observation resource to be attached to every item resource of the rooted subtree. As items are always added as the leaves of such tree, this method enables to pass and apply such deep observation pattern to the whole subtree.

Partitioning Strategy DhtFlex employs a structured peer-to-peer overlay as *Chord* [4] for data resource placement: such overlay protocol can be used to determine the *root* of a resource, that is, the Chord peer which possesses the numerically closest matching identifier in comparison to the resource's identifier.

However, as a dynamic peer environment is assumed, network conditions can change over time. As a result, a resource's corresponding root may vary. Peers that enter or leave the network demand the used overlay protocol to adjust responsibilities for affected key ranges; for instance, gaps in the overlay resulting from crashed or temporarily unavailable peers need to be closed. As DhtFlex does support crash-recovery, as well as crash-stop failure models, it is able to exploit positive dynamics of a structured peer-to-peer overlay, where available peers may take over the key range of failed ones.

Replication Strategy DhtFlex's partitioning strategy basically maps identifiers to values; thereby, a value may be an arbitrary object or item represented as data resource, which may be replicated and persistently stored. An object is retrieved by using the identifier under which it was published. DhtFlex uses replication in order to ensure high availability and durability of administrated data resources. Thereby, it supports a flexible degree of replication, that allows an adjustment per data resource type: if a peer leaves the system, for example, by crashing, its administrated data resources become unavailable. A replication mechanism increases data availability by storing data at several peers. But, in the face of concurrent modifications mutual consistency of replicated data resources may be violated, some replicas may not be up to date. The requirements of content repository functionality demands for DhtFlex to be able to get the current valid replica.

A replicated data item is independent of the peer on which it resides and may be regarded as virtual. This applied *virtualization* enables DhtFlex to employ structured overlay routing as partitioning strategy. Thereby, DhtFlex manages all replication functions; the overlay is accessed only to conduct necessary information to construct a *replication group*. A replication group configuration is a set of peers that are responsible to administrate a certain replicated data resource.

The size of such set is defined by the resource's replication degree. A replication group of size n consists of one master and $n-1$ replicas.

Regarding the replication model, DhtFlex implements a *primary-copy* replication pattern [12] per replication group: a replication group's master is used to serialize and apply all updates to a mutable data object. In order to benefit the partitioning strategy, DhtFlex uses the unique identifier of a data resource to configure the corresponding root in the overlay as master. Accordingly, DhtFlex targets to fill the replication group set with the *available* $n-1$ peers succeeding a root in the overlay, the $n-1$ *root successors*. Hence, a replication group of size n shall contain those $n-1$ peers that are relevant to become a root for the key after network conditions change. Regarding fault-tolerance aspects, these $n-1$ peers are ideal candidates to place the replicas of a given data object.

The master of a replication group is responsible to ensure the replication factor for the data resources that fall within its key range. This means, in addition to their conservation in local storage, the master needs to replicate the resources to the remaining replicas. This implies, that changes on resources have to be propagated to all replicas in order to ensure consistency.

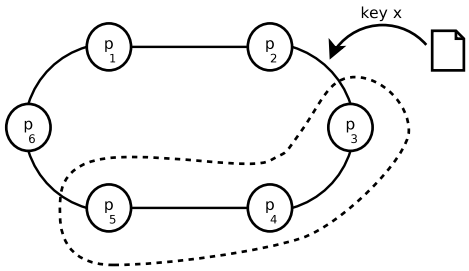


Fig. 10. Combination of Replication Strategy and Partitioning Strategy

The replication strategy in combination with the used partitioning strategy is exemplified in Figure 10. It shows a replication group consisting of one master peer p_3 and two additional replica peers: the master p_3 replicates the data object for identifier (*key*) x at peer p_4 and p_5. Hence, p_5 stores values that fall into the ranges $(p_2, p_3]$, $(p_3, p_4]$, $(p_4, p_5]$. As explained, the employed structured peer-to-peer overlay allows each peer to determine which peers should be contained in the replication group for a certain key.

6 Evaluation

The key contributions of this paper are (i) a vertical integration architecture coupling low-level field automation systems with business information systems using a decentralized routing and aggregation methods and a declarative query interface and (ii) a peer-to-peer content (document) repository which can be used to coordinate and integrate local processes in a cross-organizational fashion,

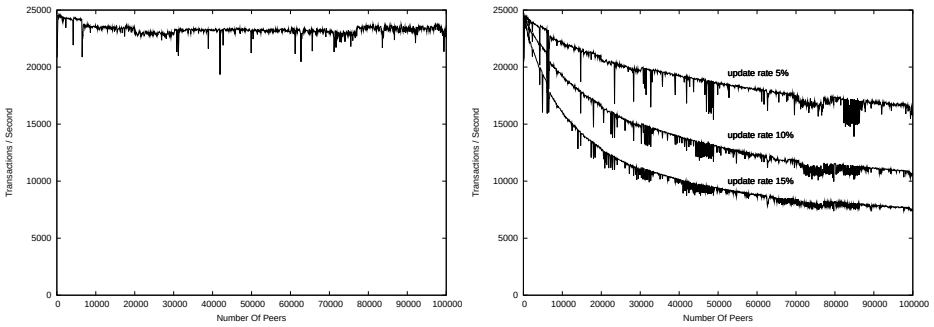
allowing, e.g., reliable and timely provision of outage reports. The following paragraphs evaluate both contributions based on the application scenario in Section 2.

6.1 Aggregation and Routing

The declarative approach and utilization of peer-to-peer protocols introduces rather obvious benefits like scalability, resilience, resource virtualization and adaptiveness to the power network domain. With engineering and maintenance costs being major matters of expensive, self configuration and adaption to new devices and topologies yield immediate cost reduction for utilities. Efficient management of complex networks will become even more important in the near future. With the emergence of distributed generation retrieving detailed real time information on the current network state becomes more and more important as, for example, reversed power flows can circumvent protection systems and yield equipment failure. Among the challenges of retrieving a correct network state is the extraction of information from vast amounts of sensor data. The declarative approach proposed enables the calculation of key performance indicators in-network, and thus eliminates the need to transfer high volumes of data to a central point. The example Query 4 presented in Section 4.2 shows nicely how the complexity of the underlying network is hidden. Queries never use physical network addresses but virtual identifiers. Keywords get resolved and queries reevaluated periodically when messages are sent thus the platform adaptively reorganizes itself in the event of failure or the appearance of better suited nodes. Processing will still work even if small fractions of nodes are unavailable as is common in large networks.

Using a centralized meta data index is an efficient and entirely pragmatic approach but also a potential performance bottleneck. We ran extensive tests and simulations to optimize the meta data index performance. The testbed consisted of ten commodity PCs each equipped with relatively low end, Intel Pentium IV processors and 1GB of RAM. Since the project is in an early state of implementation, we are not interested in absolute peak transaction rates but rather how the system adapts to increased node numbers and update rates. As test data the configuration of up 100000 peers was loaded into the indexing group. Of each node 20 attributes where indexed. To simulate how the indexing group performs with increasing node numbers, we ran 10000 queries with numbers of peers ranging from 100 to 100000.

Figure 11a illustrates the results. Plotted are the transactions per second as measured directly on the indexing nodes, i.e., not including the time required for communication and compilation. The rate remains almost constant with a minimal decrease towards 100000 peers. As stated earlier, indexing nodes are optimized for read operations as we assume an update rate of below 1% in regular power networks. In consequence, Figure 11b shows the impact of high update rates on performance. While for low node numbers performance only deviates slightly from the read only case, transactions rates are down one third in the case of 100000 peers and an assumed 5% update rate and down to 19%



(a) Read only transactions (b) Read/write transactions

Fig. 11. Queries are resolved through the query catalog of the centralized index. The catalog maps to nodes that are responsible for the data requested.

for the 15% update rate case. However update rates of 5% or even 15% are not realistic as it would mean that a utility would lose or replace a large fraction of its network which in return would dramatically reduce read only transactions.

The average round trip time for a query was about 400ms in the case with 100000 peers. This includes parsing, compiling transferring, processing and receiving results. However, while it provides a rough idea on processing time, this figure has little value as it strongly depends on the network topology and may increase or decrease depending on the communication infrastructure used. Some power network applications are extremely time critical. A backup protection system as discussed in [13], [14], [15], and [16] requires full query execution and data delivery in about 200ms real time. In such cases, the node executing the protection logic would subscribe to necessary measurements in advance. Results would then be delivered to a local cache from which they can be taken for state assessment. In the context of the application scenario of Section 2, these performance measures are sufficient.

In order to achieve a highly efficient outage report process, network failures must be detected automatically. The programmability of the vertical architecture reflects this aspect as new filters and detection algorithms can be implemented on demand. Since these programs run in-network, their analysis can be based on high detail measurement data enabling early detection of complex failure patterns.

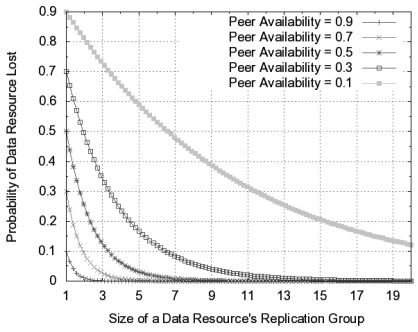
6.2 P2P-based Content Repository

The evaluation of the peer-to-peer based content repository highlights two important properties considering the decentralized management of different versions of outage documents: (i) the availability of a certain document version. (ii) The latency to retrieve a certain version.

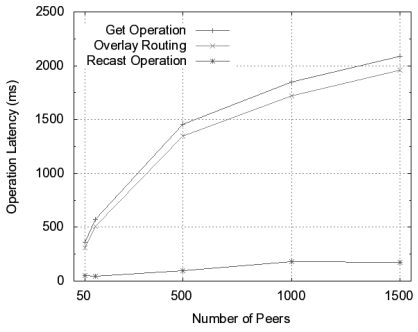
For example, the DhtFlex approach ensures reliability of different data resource versions using replication as redundancy scheme: a certain number of identical copies are stored at different peers. As a data resource's replication factor ρ influences its availability, the value of ρ should be set appropriately depending on the demanded degree of availability. As in worst case, DhtFlex only requires a single copy of the ρ replicated (immutable) data resources to be available to progress successfully, the probability P_{fail}—the failure of a data resource's whole replication group—is given by the following equation (we assume that peers fail independently and that all peers have an identical (average) availability α_{peer}):

$$P_{fail} = P(all\ \rho\ replica\ peers\ fail) = P(one\ replica\ peer\ fails)^{\rho} = (1 - \alpha_{peer})^{\rho} \quad (1)$$

A resource's replication factor ρ can be adjusted depending on the desired availability aim, as stated by the following formula: $\rho = \frac{log(P_{fail})}{log(1-\alpha_{peer})}$. Figure 12(a) depicts the probability P_{fail} to actually lose an immutable data resource—depending on different values for α_{peer} and ρ: one observation is that comparatively small values for the size of a replication group suffice to reduce the probability loosing a certain data resource significantly.



(a) Worst Case Probability a Document Version is Lost



(b) Latency of DhtFlex for Immutable Data Resources using Iterative Overlay Routing

Fig. 12. Evaluation of Availability and Performance Properties

The evaluation of performance properties to retrieve a certain outage document version uses an experimental setup integrating a *King* data set [17]. This approach enables to gain estimations of DhtFlex's communication costs based on real measurement data of thousands of Internet hosts. Figure 12(b) indicates the latency of DhtFlex for operations on *immutable* data resources, for example,

outage document versions. All shown latencies represent the average value of ten measurements per operation; each data resource is allocated to a replication group of six (different) peers. The results show that a *get* operation is strongly affected by the costs to perform peer-to-peer overlay routing, which increase with the number of peers—the operation itself introduces, however, rather constant overhead. As the *recast* operation does not require overlay routing, its latency is comparatively small. In addition, outage document versions may be arbitrarily cached at client-site to avoid overlay lookup latencies and thus to reduce the latency of *get* operations—this is an important remark as these are supposed as the most requested kind of operation for such resources.

As a result, the peer-to-peer based content repository is able to ensure high availability of outage documents and to achieve good performance properties regarding their data access. This may benefit the adoption of the decentralized solution in comparison to the state of the art system (see Section 2.2).

7 Related Work

In the context of wide area sensor network Aberer et al. [18] published research that provides similar concepts to the approach taken in Section 4. Our architecture, however, was heavily influenced by applicability in current industrial systems. Our sensor model is determined by IEC 61850 which supports an integrative approach with current automation equipment. Compiling queries and programs to binary form allows us to run query engines on low cost embedded devices as well as desktop PCs.

A good overview of distributed databases is given in [19]. [20] provides an overview of current research topics on data stream management. In [21], D. Kucuket et al. introduce a streaming database solution to monitor power quality. Mariposa [22] introduces an architecture for wide area distributed databases in perspective of traditional distributed data management systems (DBMS). The approach includes a micro economic paradigm used for query and storage optimization. AURORA [23], STREAM [24], Cougar [25] and others discuss general query processing in sensor networks. AURORA allows users to create queries in a graphical representation. STREAM and Cougar extend the SQL with temporal semantics but do not provide the extensions of a programming language.

A considerable body of research is available on peer-to-peer systems, including DHTs [4], [26], and gossip based algorithms [27]. In contrast to peer-to-peer systems deployed in the Internet, in industrial systems, we find a more comfortable environment in some respect: we can expect lower churn rates and less problems with network address translation. However, we must adapt algorithms originally designed for media distribution in the Internet to the resource-constrained environment of industrial systems. Hence, issues such as determinism and real-time capabilities must be addressed. In previous work [5] [28] [8] we showed already the potential of peer-to-peer algorithms in industrial domains beyond content distribution.

Peer-to-peer systems generally lack capabilities to support content repositories. In addition, most of these systems use a rather monolithic approach and usually do not target consistency of concurrent operations. Moreover, the systems commonly do not offer some degree of flexibility regarding certain properties, as different availability demands. In order to delimit our approach of a peer-to-peer content repository, we subsequently present selected related work. It is worth mentioning that to our knowledge, our work is the first peer-to-peer implementation available, which addresses the powerful features of JSR-170.

For instance, the Cooperative File System (CFS) [29] provides a peer-to-peer infrastructure for wide-area storage and focuses on guarantees for efficiency, robustness, load balancing, and scalability. CFS offers a simple file-system interface, which interprets constituent blocks that may be stored at different peers as files. On bottom, CFS is based on a Chord DHT routing scheme for the lookup support of data blocks. CFS basically offers a read-only system from a user's point of view where consistency is hardly a problem. In contrast to our approach, only a single user, that is, the publisher, is able to modify its own data. However, CFS does not support a publish-subscribe mechanism nor our multi-level naming concept.

OceanStore [30] is designed as an Internet-scale, persistent peer-to-peer data store for incremental scalability, secure sharing, and long-term durability. Data objects are identified by global unique IDs and persisted in an underlying DHT. For an efficient storage, OceanStore splits up data objects into blocks. Multiple users are supported to work on the same data objects, respecting concurrent modifications. There exists also the possibility to equip data files with ACLs. Unlike our work, there are no private storage sections available, nor exists the possibility to store sensitive data locally publishing only metadata, but allowing a transparent retrieval. In addition, no advanced features like observations or type concepts are offered and imposed.

OGSA-DAI [31] is a middleware that enables heterogeneous data resources, e.g., underlying relational or XML databases, to be queried, updated, transformed, compressed, and delivered via web services within a Grid environment. Regarding security, OGSA-DAI establishes a role-based model to grant data access permissions, mapping credentials to corresponding underlying database roles. In contrast to our work, OGSA-DAI targets on concepts of a more generic middleware layer, especially integrating client–server solutions. For the horizontal integration, however, we focus on peer-to-peer techniques with the aim of eliminating central entities in the network. None the less, our repository solution may be integrated to OGSA-DAI as data storage.

8 Conclusion and Outlook

In this paper we introduced an integration architecture and methods to improve interoperability and increase automation of cross enterprise business processes. While the focus of this paper is on the energy domain, we believe that the results are not restricted to this area of application. The vertical integration approach

illustrated in this paper constitutes a high performance platform to manage large scale data intensive systems. Providing an interface of declarative querying and programmable data sources it abstracts from the underlying physics and can be extended to include assets unknown during design time as well add new functions for situation based analytics. Simulations show the performance of the indexing group which scales well to 100000 peers and beyond. Everything considered, the architecture increases the efficiency of today's power networks as well as other industrial applications that rely on large, globally distributed networks with thousands of nodes. Considering horizontal integration, the approach of using a peer-to-peer based content repository to substitute the centralized implementation of the ETSOVista platform's back-end system avoids a single point of failure in critical situations. From a technical point of view, the proposed system is able to benefit distributed collaboration in outage management processes by supporting a publish-subscribe mechanism to enable the rapid notification of critical events to interested parties. The evaluation shows that the system is robust against document losses and achieves good performance regarding document access. From a business point of view, the decentralized solution shows the potential to avoid vendor lock-in situations resulting from a proprietary market information aggregator.

What we have not yet considered in this paper is support of modeling and enacting the cross-enterprise business processes which will be required to operate on top of the document repository. In the future, we shall investigate the suitability of previous work on model-driven design and enactment of cross-organizational business processes (see, e.g., [3],[32], [33]). Also, there has been considerable research in agent-supported market mechanisms for efficient resource allocation[3], which can be employed on top of our run-time infrastructure. A further area of future of work will be to support interoperability and automation of outage report publishing at the system operator's side. This would eliminate the need to manually commit a report to the market information aggregator. At the side of an interested market participant, future work may turn towards the development of techniques to establish a business logic to automatically react to (critical) outage events.

References

1. Fischer, K., Müller, J.P., Stäber, F., Friese, T.: Using peer-to-peer protocols to enable implicit communication in a BDI agent architecture. In Bordini, R.H., Dastani, M., Dix, J., Fallah, A.E., eds.: Programming Multi-Agent Systems 4. Volume 4411 of LNAI. Springer-Verlag (2007) 15–37
2. CRUTIAL: Analysis of new control applications. Deliverable D2 (2006) Critical Utility Infrastructural Resilience, EU Project IST-FP6-STREP-027513.
3. Roser, S., Bauer, B., Müller, J.P.: Model- and architecture-driven development in the context of cross-enterprise business process engineering. In: SCC '06: Proceed-

[3] Just one example out of many is the work on agent-based auction done at EPRI, see `http://www.agentbuilder.com/Documentation/EPRI/index.html`

ings of the IEEE International Conference on Services Computing, Washington, DC, USA, IEEE Computer Society (2006) 119–126
4. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: 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, New York, NY, USA, ACM (2001) 149–160
5. Stäber, F., Gerdes, C., Müller, J.P.: A peer-to-peer-based service infrastructure for distributed power generation. In: Proc. of 17th IFAC World Congress, Seoul, Korea, Intl.l Federation of Automatic Control. (2008)
6. Bartlang, U., Stäber, F., Müller, J.P.: Introducing a JSR-170 standard-compliant peer-to-peer content repository to support business collaboration. In Cunnigham, P., Cunnigham, M., eds.: Proceedings of eChallenges e-2007 Conference. Volume 4 of Information and Communication Technologies and the Knowledge Economy., IOS Press (2007) 814–821
7. Date, C.J.: A critique of the SQL database language. SIGMOD Rec. **14** (1984) 8–54
8. Gerdes, C., Müller, J.P.: Data centric peer-to-peer communication in power networks. In: Proceedings of the First Workshop on Global Sensor Networks (GSN '09) at KIVS'2009, Kassel, Germany (2009) Forthcoming.
9. Day Management AG: Content Repository API for Java^{TM} technology specification (2005) Java Specification Request 170, version 1.0.
10. Bartlang, U., Müller, J.P.: Dhtflex: A flexible approach to enable efficient atomic data management tailored for structured peer-to-peer overlays. In Mellouk, A., Bi, J., Ortiz, G., Chiu, D.K.W., Popescu, M., eds.: Proc. Third International Conference on Internet and Web Applications and Services. Volume 3., IEEE Computer Society Press (2008) 377–384
11. Lamport, L.: The part-time parliament. ACM Trans. Comput. Syst. **16** (1998) 133–169
12. Gray, J., Helland, P., O'Neil, P., Shasha, D.: The dangers of replication and a solution. In: SIGMOD '96: Proceedings of the 1996 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM (1996) 173–182
13. Wang, X.R., Hopkinson, K.M., Thorp, J.S., Giovanini, R., Coury, K.B.D.: Developing an agent-based backup protection system for transmission networks. In: Power Systems and Communication Systems Infrastructures for the Future. (2002)
14. Stedall, B., Moore, P., Johns, A., Goody, J., Burt, M.: An investigation into the use of adaptive setting techniques for improved distance back-up protection. IEEE Transactions on Power Delivery **11** (1996) 757–762
15. Kezunovic, M.: Intelligent systems in protection engineering. In: Proc. of InternationalConference on Power System Technology. Volume 2. (2000) 801–806 vol.2
16. Kim, M., Damborg, M., Huang, J., Venkata, S.: Wide-area protection using distributed control and high speed communications. In: 14th Power Systems Computation Conference. (2002)
17. Gummadi, K.P., Saroiu, S., Gribble, S.D.: King: estimating latency between arbitrary internet end hosts. In: IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment, New York, NY, USA, ACM (2002) 5–18
18. Aberer, K., Hauswirth, M., Salehi, A.: Infrastructure for data processing in large-scale interconnected sensor networks. Mobile Data Management, 2007 International Conference on (2007) 198–205
19. Oezsu, M.T., Valduriez, P.: Principles of Distributed Database Systems. Prentice Hall (1999)

20. Golab, L., Oezsu, M.T.: Issues in data stream management. SIGMOD Rec. **32** (2003) 5–14
21. Kucuk, D., Boyrazoglu, B., Buhan, S.: Pqstream: A data stream architecture for electrical power quality. In: International Workshop on Knowledge Discovery from Ubiquitous Data Streams. (2007)
22. Stonebraker, M., Aoki, P.M., Litwin, W., Pfeffer, A., Sah, A., Sidell, J., Staelin, C., Yu, A.: Mariposa: A wide-area distributed database system. VLDB Journal: Very Large Data Bases **5** (1996) 48–63
23. Cherniack, M., Balakrishnan, H., Balazinska, M., Carney, D., Cetintemel, U., Xing, Y., Zdonik, S.: Scalable Distributed Stream Processing. In: CIDR 2003 - First Biennial Conference on Innovative Data Systems Research, Asilomar, CA (2003)
24. Garofalakis, M., Gehrke, J., Rastogi, R.: Data Stream Management: Processing High-Speed Data Streams (Data-Centric Systems and Applications). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2007)
25. Gehrke, J., Madden, S.: Query processing in sensor networks. IEEE Pervasive Computing **3** (2004) 46–55
26. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: IFIP/ACM International Conference on Distributed Systems Platforms. Volume 2218. (2001) 329–350
27. Boyd, S., Ghosh, A., Prabhakar, B., Shah, D.: Gossip algorithms: design, analysis and applications. INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE **3** (13-17 March 2005) 1653–1664 vol. 3
28. Friese, T., Müller, J.P., Smith, M., Freisleben, B.: A robust business resource management framework based on a peer-to-peer infrastructure. In: In: Proc. 7th International IEEE Conference on E-Commerce Technology. (2005) 215222
29. Dabek, F., Kaashoek, M.F., Karger, D., Morris, R., Stoica, I.: Wide-area cooperative storage with CFS. In: SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles, New York, NY, USA, ACM Press (2001) 202–215
30. Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Wells, C., Zhao, B.: OceanStore: an architecture for global-scale persistent storage. In: ASPLOS-IX: Proceedings of the ninth international conference on Architectural support for programming languages and operating systems, New York, NY, USA, ACM (2000) 190–201
31. Karasavvas, K., Antonioletti, M., Atkinson, M., Hong, N.C., Sugden, T., Hume, A., Jackson, M., Krause, A., Palansuriya, C.: Introduction to OGSA-DAI services. In Herrero, P., Pérez, M.S., Robles, V., eds.: Scientific Applications of Grid Computing. Volume 3458 of LNCS., Springer Berlin/Heidelberg (2005) 1–12
32. Kahl, T., Zinnikus, I., Roser, S., Hahn, C., Ziemann, J., Müller, J.P., Fischer, K.: Architecture for the design and agent-based implementation of cross-enterprise business processes. In et al., R.G., ed.: Enterprise Interoperability II: In Proceedings of the 3rd International Conference on Interoperability of Enterprise Software and Applications (I-ESA'07), Springer-Verlag (2007) 207–218
33. Stiefel, P., Müller, J.P., Bessling, S., Hausknecht, C., Dokters, T.: Realizing dynamic product collaboration processes in a model-driven framework: Case study and lessons learnt. In: Proceedings of the 14th International Conference of Concurrent Engineering, Lisbon, Portugal (2008) 983–990