Condenser: Cloud Storage Gateway

Jesse Blum
for the
Horizon Digital Economy Research Institute
jesse.blum@nottingham.ac.uk

Nov. 2011

Version 0.2 draft

Revision History

Name	Date	Reason For Changes	Version
SRS-DLP-0.1	11-2011	First draft	0.1
SRS-DLP-0.2	11-2011	Research shift from logging to cloud gateway	0.2

Contents

List of Figures

1 Scope

This Software Requirements Specification (SRS) identifies the requirements for the Condenser Data Logging Platform.

1.1 Problem Statement

A trend across various Horizon projects (Energy and Relate for instance) is to provide distributed (immutable) sensor data capture, process and storage facilities. The connection between the local environment and external control and storage environment, however, is intermittent and unreliable. This disconnection results in many disadvantages such as the emergence of data silos and possible data loss. Horizon projects have heretofore provided bespoke solutions to the external connections, however these have become difficult to maintain with the increase of projects.

1.2 Objectives

The aim of the Condenser (meta)-project is to provide a configurable component for local logging of immutable data to a local server with intermittent synchronisation to external (possibly cloud-based) storage environments. The Condenser component will be made widely available and be data-type neutral.

1.3 Document Overview

Section 2 is divided into functional requirements and the non-functional requirements. The functional requirements explain the interactions between Condenser and its environment (including usage scenarios and cases). The non-functional requirements explain the quality constraints. Section 3 provides critical review of state-of-the-art existing options and existing components that may be leveraged in the delivery of Condenser.

2 Requirements

This section defines functional and non-functional requirements for the Condenser tool. It begins with an enumeration of usage scenarios that Condenser could be involved in. These are used to help determine the following requirements.

2.1 Actors

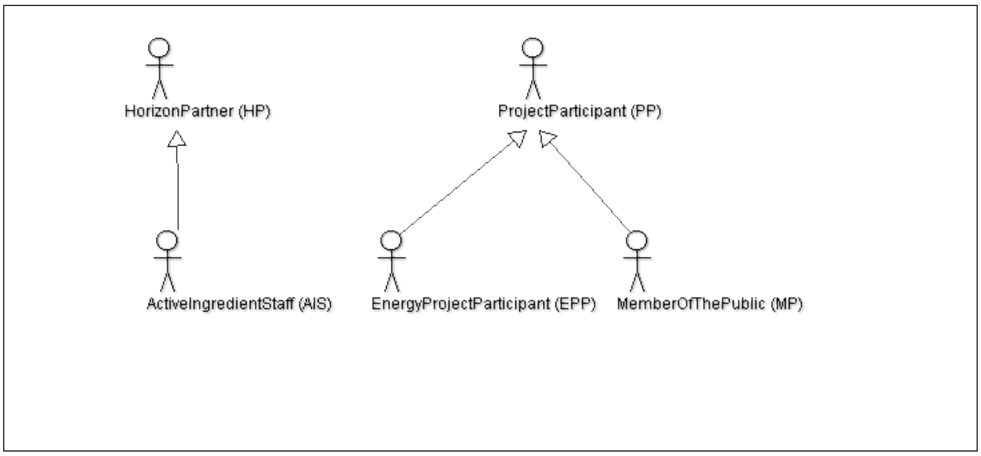

Figure 1: Actor relationships.

The actors shown in figure 1 are involved in the following usage scenarios and cases. They include:

- Horizon project staff (HPS) – These actors are involved in research projects as investigators.
- Horizon Partner (HP) – These actors are involved in research projects as investigators.
- Active Ingredient staff (AIS) – These actors are a type of Horizon Partner.
- Project Participant (PP) – These actors are involved in research projects as participants.
- Member of the public (MP) – These actors are a type of Project Participant recruited from the general public.
- Energy project participant (EPP) – These actors are a type of Project Participant recruited in particular for energy projects.

2.2 Usage Scenarios

2.2.1 Relate Project: Condenser Installation

Participating Actors: Anne:AIS , Harry:HPS
Event Flow:

1. Anne prepares to run an energy and climate data collection exercise in the Brazilian rain forest using open mobile sensing kits (OMSK).

2. Harry installs Condenser on the local server component of the OMSKs.

3. Harry configures a Cloud-server to accept data uploads from each OMSK.

4. Harry uses the Condenser RESTful interfaces to configure them to log the data from their local stores to the cloud when connectivity is available. He also configures them to run in low-powered mode by setting their reconnection attempts to a low amount and to only send data in aggregated bursts every hour.

2.2.2 Relate Project: Oasis of Connectivity

Overview: Condenser's behaviour in long period without connectivity
Participating Actors: Anne:AIS, Miguel:MP
Event Flow:

1. Anne conducts a public art activity in the Brazilian rain forest.

2. Anne gives Miguel an OMSK.

3. Miguel activates the OMSK as part of the activity.

4. On start up the OMSK activates Condenser and begins to capture temperature, humidity, C02 and energy information.

5. No network connectivity is available to Condenser so it evaluates the expected data load and determines that no action needs to occur until the next connection attempt. [Note that this would be much more complicated if Condenser determined that the current data capture rates exceeded the storage capacity]

6. Miguel returns the OMSK to Anne, and she returns to her office with it.

7. **Condenser** successfully attempts to connect to the network after its timeout.

8. **Condenser** transfers the data from the OMSK to the Cloud-store.
9. Anne looks at the day's data using the web-based visualisation tools. These have immediate access to the Cloud-data.

2.2.3 Energy Project: Trickle of Connectivity

Overview: Condenser's behaviour during intermittent connectivity
Participating Actors: Harry:HPS, Ernie:EPP
Event Flow:

1. Harry is conducting an energy study.
2. Harry sets up a Shiva Plug computer with an energy monitor.
3. Harry installs Condenser on the Shiva Plug.
4. Harry configures Condenser to attempt to reconnect whenever possible and to send discrete data points to a rack-mounted server using particular authentication details.
5. Harry installs the energy monitor in Ernie's home and configures it to use Ernie's home router.
6. **Condenser** attempts to transmit data as the energy monitor receives them, but connectivity is intermittent owing to Ernie's unstable Internet connection.

2.3 Use Cases

2.3.1 Installation

The following use cases pertain to installation of Condenser. Figure 2 shows a diagram depicting the relationships between the installation use cases.

Use Cases:

Install local server and DB
Participating Actors: Horizon project staff (HPS)
Event Flow:

1. HPS downloads (or compiles from source) the appropriate Condenser binaries from a well-known repository.

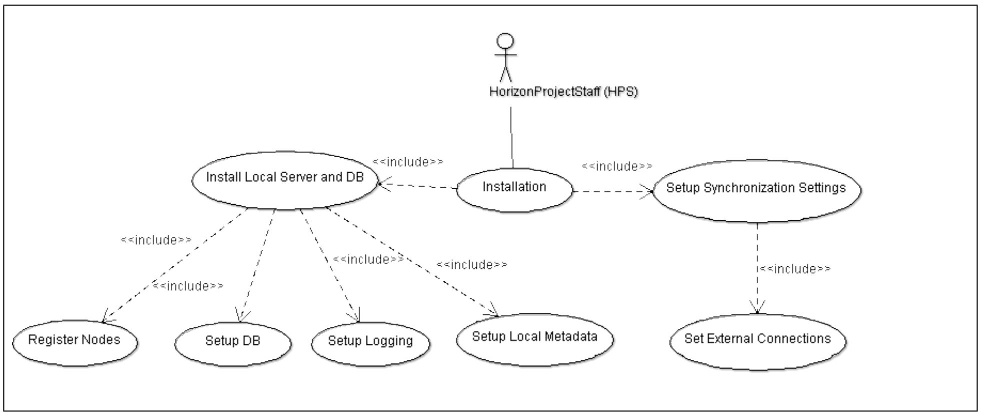

Figure 2: Use cases defining Condenser installation.

2. HPS sets up the Condenser file directory structure on the appliance.
3. HPS sets up Condenser's metadata (such as provenance information). See the setup local metadata use case.
4. HPS downloads and installs the database from a well-known repository (see the Setup DB use case).
5. HPS installs the Condenser service and configures it to cache data in the database.
6. HPS sets Condenser's connection settings (including any public key and password details).
7. HPS sets Condenser's logging settings (see the Setup Logging use case).

Entry Conditions:

- The given appliance meets the minimum hardware and operating system requirements.
- The given appliance is in a clean state (ie. there are no previous installation files/settings that can interfere with this installation).
- HPS has administrative control (and passwords) of the given appliance.
- HPS has Internet access.

Exit Conditions: Condenser is installed on the given appliance and ready to be externally synchronized.
Quality Requirements:

- Condenser is able to connect to the Internet and local sensor network.

- Data pushed to Condenser is cached appropriately.
- Condenser is able to push data upstream securely.

Set synchronization settings
Participating Actors: Horizon project staff (HPS)
Event Flow:

1. HPS accesses the synchronization settings web interface for the given device.
2. HPS sets connection settings to the external (possibly cloud) data storage system (see Set External Storage use case).
3. HPS sets synchronization rules (such as time-based or event-based synchronization).
4. HPS sets local data cache rules (how to handle limited data storage).

Entry Conditions:

- Condenser service is running.
- External (possibly cloud-based) storage has been setup and may be connected to remotely.
- HPS has Internet access.

Exit Conditions:

- Synchronization between Condenser's local cache and external data storage is up and running.
- Condenser's synchronization settings are viewable and update-able by an administrator with suitable privileges.

Quality Requirements:

- Connection settings (such as passwords/keys) to external storage are suitably encrypted.

Set external storage connection settings
Participating Actors: Horizon project staff (HPS)
Event Flow:

1. HPS accesses the external storage connection settings web interface for the given device.

2. HPS adds one or more connections to the connections store.

3. HPS successfully tests the connection(s).

Entry Conditions: Includes Set Synchronization use case entry conditions. **Exit Conditions:** Condenser is shown to be able to connect and transmit data to external storage.

Setup local database
Participating Actors: Horizon project staff (HPS)
Event Flow:

1. HPS downloads the database from a well-known repository.

2. HPS installs the database.

3. HPS sets up the database structure appropriate for the given project or application.

Entry Conditions: No other database is running on the given port. **Exit Conditions:** The local database is setup and teted.

Setup logging
Participating Actors: Horizon project staff (HPS)
Event Flow:

1. HPS accesses the logging settings web interface for the given device.

2. HPS configures the logging settings.

3. HPS tests to make sure that activity is being logged appropriately.

Entry Conditions: Empty log settings and no log files present. **Exit Conditions:** Condenser is setup to log activity to a given degree of verbosity. **Quality Requirements:**

- Different levels of logging are available.

- Options are avaialable to set the disk size for log storage.

- Options are avaialble for log cleanup rules.

Setup server metadata
Participating Actors: Horizon project staff (HPS)
Event Flow:

1. HPS accesses the metadata settings web interface for the given device.

2. HPS configures Condenser's provenance settings.

3. HPS adds any special metadata for the given project or application.

Entry Conditions: Empty metadata settings.
Exit Conditions: Condenser's metadata is setup.

Register nodes
Participating Actors: Horizon project staff (HPS)
Event Flow:

1. HPS accesses the node registry web interface for the given device.
2. HPS registers one or more nodes to capture data for.
3. For each node HPS records data provenance information (such as node type – ie. type of sensor) and other metadata pertaining to the given node.
4. HPS ensures that each node can transmit data to Condenser and that the given data are stored locally.

Entry Conditions: No nodes are registered with Condenser.
Exit Conditions: one or more nodes is registered with Condenser.
Quality Requirements: It must be possible to quickly register a large number of nodes.

2.3.2 Local data Capture

The following use cases pertain to how Condenser records data local to it. Figure 3 shows a diagram depicting the relationships between the local data capture use cases.

Capture Data
Participating Actors: ...
Event Flow:

1. ...

Entry Conditions:
Exit Conditions:
Quality Requirements:

Capture Metadata
Participating Actors: ...
Event Flow:

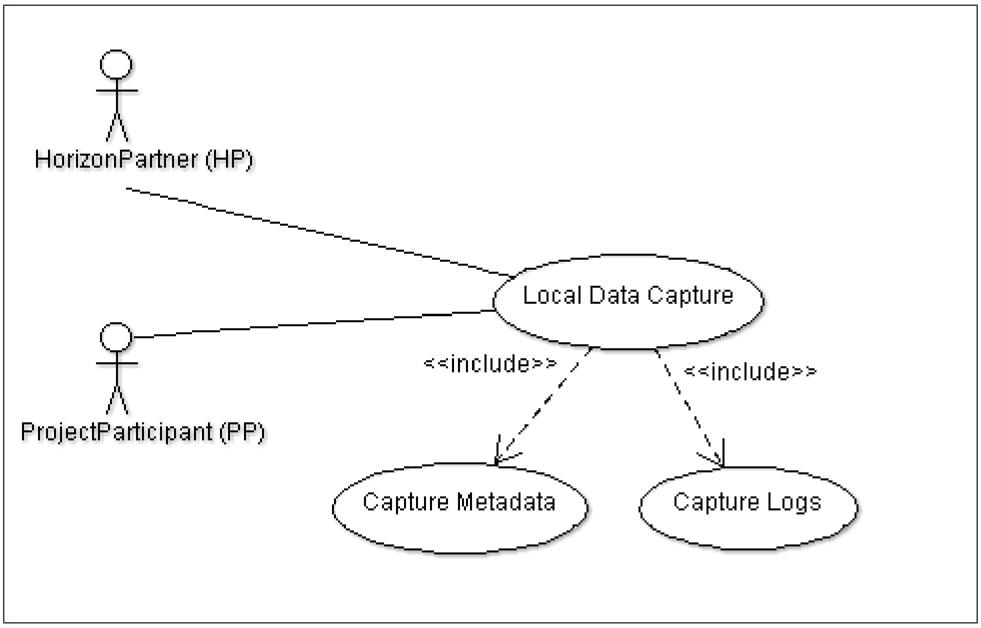

Figure 3: Use cases defining Condenser local data capture.

1. ...

Entry Conditions:
Exit Conditions:
Quality Requirements:

Capture Logging Information
Participating Actors: ...
Event Flow:

1. ...

Entry Conditions:
Exit Conditions:
Quality Requirements:

2.3.3 Data Transfer

The following use cases pertain to how Condenser transfers data to the cloud. Figure 4 shows a diagram depicting the relationships between the data transfer use cases.

Standard Data Transfer
Participating Actors: ...
Event Flow:

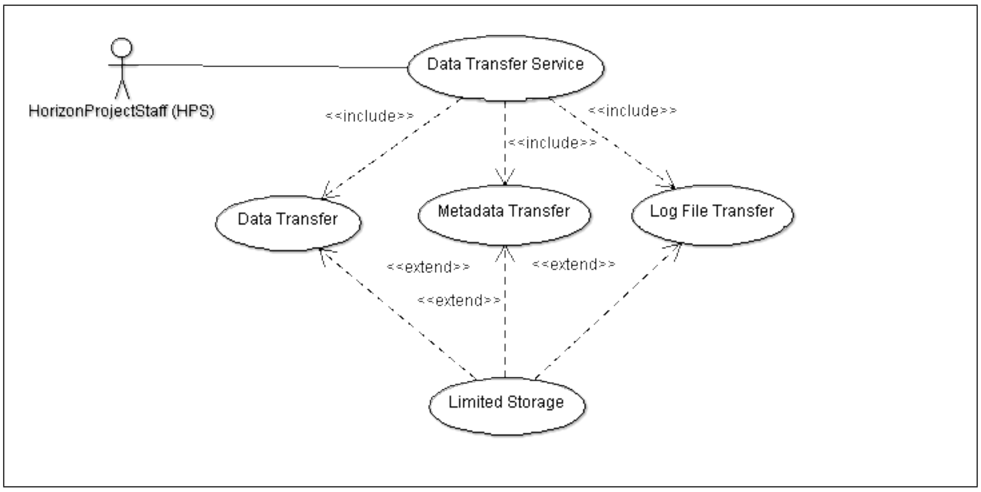

Figure 4: Use cases defining Condenser data transfer.

1. ...

Entry Conditions:
Exit Conditions:
Quality Requirements:

Metadata Transfer
Participating Actors: ...
Event Flow:

1. ...

Entry Conditions:
Exit Conditions:
Quality Requirements:

Handling limited data storage concerns
Participating Actors: ...
Event Flow:

1. ...

Entry Conditions:
Exit Conditions:
Quality Requirements:

2.3.4 Local Service Review

The following use cases pertain to the review of service settings and logs of Condenser. Figure 5 shows a diagram depicting the relationships between the Local service review use cases.

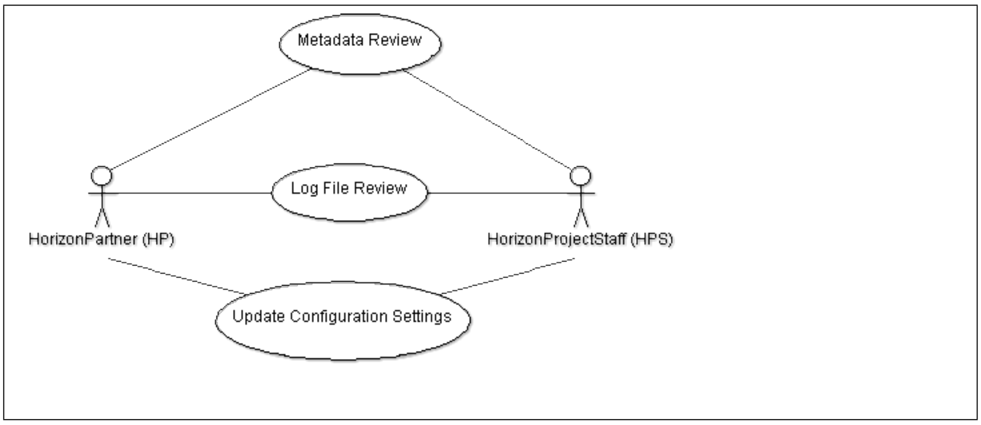

Figure 5: Use cases defining Condenser service review.

Metadata Review
Participating Actors: ...
Event Flow:

1. ...

Entry Conditions:
Exit Conditions:
Quality Requirements:

Log Review
Participating Actors: ...
Event Flow:

1. ...

Entry Conditions:
Exit Conditions:
Quality Requirements:

Update configuration settings
Participating Actors: ...
Event Flow:

1. ...

Entry Conditions:
Exit Conditions:
Quality Requirements:

2.3.5 Decommissioning

The following use cases pertain to the decomissioning of Condenser components. Figure 6 shows a diagram depicting the relationships between the decomissioning use cases.

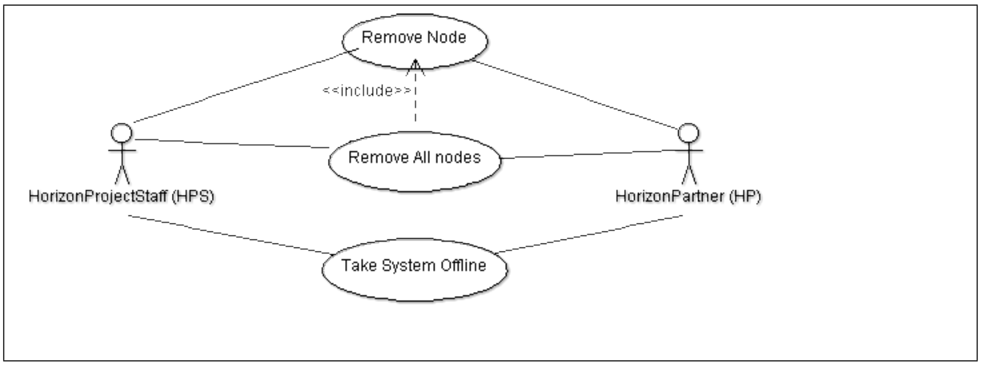

Figure 6: Use cases defining Condenser decomissioning.

Removing a node
Participating Actors: ...
Event Flow:

1. ...

Entry Conditions:
Exit Conditions:
Quality Requirements:

Removing all nodes
Participating Actors: ...
Event Flow:

1. ...

Entry Conditions:
Exit Conditions:
Quality Requirements:

Taking Condenser offline
Participating Actors: ...
Event Flow:

1. ...

Entry Conditions:
Exit Conditions:
Quality Requirements:

2.3.6 Features and Plug-ins

The following use cases pertain to advanced features of Condenser. Figure 7 shows a diagram depicting the relationships between the features and plug-ins use cases.

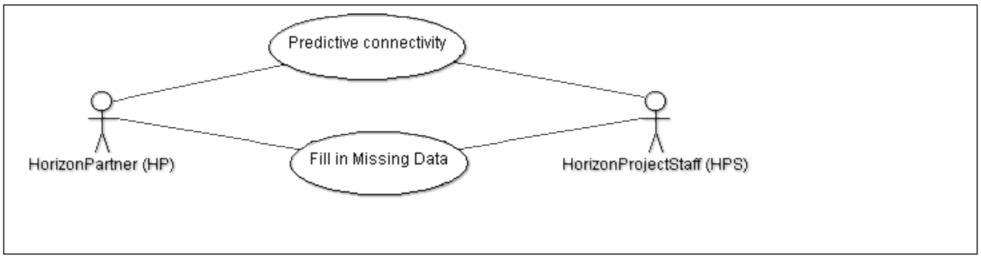

Figure 7: Use cases defining Condenser advanced features.

Predictive connectivity
Participating Actors: ...
Event Flow:

1. ...

Entry Conditions:
Exit Conditions:
Quality Requirements:

Filling in missing data
Participating Actors: ...
Event Flow:

1. ...

Entry Conditions:
Exit Conditions:
Quality Requirements:

2.4 Functional Requirements

- Condenser is a data logging infrastructural component for use in software projects requiring intermittent synchronization with external environments.

- Functionality of Condenser is configurable through a RESTful interface. Clients can use HTTP calls to configure where and how often data is stored. Condenser can be used with cloud-based storage or more traditional server storage.
- Condenser is capable of handling disconnected operation by ensuring data are stored locally until network connection is resumed. Condenser evaluate expected data needs while offline and can be configured to clean, aggregate or delete data to handle limited local drive space.
- Condenser does not have (or need) a graphical user interface.
- Condenser will automatically provide standard metadata.
- Condenser will log its own performance to an external repository (with configurable repository and verbosity).
- Condenser should run as a background service that turns on automatically at startup

2.5 Non-functional Requirements

2.5.1 Reliability and Security

Condenser should continue to work reliably during periods of network disconnect. Error conditions that arise during periods of disconnect should be logged in a similar fashion to sensor data and transmitted externally (to a configurable store) upon resumption of network connection. Configurable options should be made available to allow for secure data transmission and storage. Data store connection settings must be encrypted.

2.5.2 Test Data

Even though Condenser will be data-type neutral, it will be tested using temperature, humidity, energy and C02 sensors data. Data sets will include: time-series data, discreet values and aggregate information. Condenser will be tested with up to 2GB of data.

2.5.3 Performance

Condenser' s external storage performance during times of reliable network connection will be configurable is order for an administrator to choose the amount of bandwidth will be taken up transferring data offsite. Condenser should support the offsite data transfer of up to 4095 sources as well as one more for its own logging information.

2.5.4 Supportability

Condenser tests, code, installation, administration and usage will be well documented.

2.5.5 Implementation

The local logging Condenser component will be operational on a Plug computer. Off the shelf technology may be (and indeed is encouraged to be) used to get a working version of Condenser built as soon as possible. New developments should be test-driven with an emphasis on documentation.

2.5.6 Interface

Condenser should support a RESTful interface for its activation, configuration and data transmission. Condenser should be compatible with other Relate project.

2.5.7 Operation

Condenser will be managed by Horizon staff or project clients.

2.5.8 Packaging

Condenser will be installed initially by Horizon project staff, with a view that client system administrators will be supported in the future. Condenser will initially be rolled-out for beta-testing in February, 2011.

2.5.9 Legal

Condenser will be licensed under The GNU Affero General Public License (AGPL 3). No liability will be assumed by Condenser's developers for losses incurred through its use since it is experimental and research driven software. Only Free (libre) software will be used for the development of Condenser.

3 Related Works

3.1 Data Brokers

3.1.1 Pachube

- RESTful
- Environment, Datastream and Datapoint model
- Push and Pull capabilities with “live” and “frozen” status
- Supports HTTPS/SSL
- Authentication is handled using API keys.

[Pac11]

3.1.2 Others

[Bri09]

3.2 Data Synchronization, Replication and Mirroring

[AST02]

3.3 Distributed Filesystem

[DZP+11]

3.4 Cloud Synchronization

3.4.1 General CS

[UFB10] [AFG+09]

3.4.2 Sensor Network CS

[HSH09] [LH10] [MF11] [Pat11] [SMT10]

3.4.3 Unstructured Database Options

- CouchDB
- MongoDB

3.4.4 Structured Database Options

- SQL Azure DB

3.5 Policies and Rules

3.6 Agents

3.7 Sensor Metadata

4 References

References

[AFG+09] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, et al. Above the clouds: A berkeley view of cloud computing. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28*, 1:1, 2009.

[AST02] S. Agarwal, D. Starobinski, and A. Trachtenberg. On the scalability of data synchronization protocols for pdas and mobile devices. *Network, IEEE*, 16(4):22–28, 2002.

[Bri09] S. Briefings. Can we plug wireless sensor network to cloud? *cloud computing: pinnacle of IT Infrastructure democratization*, 7:33, 2009.

[DZP+11] Y. Dong, H. Zhu, J. Peng, F. Wang, M.P. Mesnier, D. Wang, and S.C. Chan. Rfs: a network file system for mobile devices and the cloud. *ACM SIGOPS Operating Systems Review*, 45(1):101–111, 2011.

[HSH09] M.M. Hassan, B. Song, and E.N. Huh. A framework of sensor-cloud integration opportunities and challenges. In *Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication*, pages 618–626. ACM, 2009.

[LH10] K. Lee and D. Hughes. System architecture directions for tangible cloud computing. In *2010 First ACIS International Symposium on Cryptography, and Network Security, Data Mining and Knowledge Discovery, E-Commerce and Its Applications, and Embedded Systems*, pages 258–262. IEEE, 2010.

[MF11] J. Melchor and M. Fukuda. A design of flexible data channels for sensor-cloud integration. In *Systems Engineering (ICSEng), 2011 21st International Conference on*, pages 251–256. IEEE, 2011.

[Pac11] Pachube. *Pachube API Documentation*, 2011.

[Pat11] S.K. Patil. *Usable, lightweight and secure, architecture and programming interface for integration of Wireless Sensor Network to the Cloud.* PhD thesis, The Ohio State University, 2011.

[SMT10] P. Stuedi, I. Mohomed, and D. Terry. Wherestore: location-based data storage for mobile devices interacting with the cloud.

In *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, page 1. ACM, 2010.

[UFB10] S. Uppoor, M.D. Flouris, and A. Bilas. Cloud-based synchronization of distributed file system hierarchies. In *Cluster Computing Workshops and Posters (CLUSTER WORKSHOPS), 2010 IEEE International Conference on*, pages 1–4. IEEE, 2010.