

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

1	Scope	4
1.1	Problem Statement	4
1.2	Objectives	4
1.3	Document Overview	4
2	Requirements	5
2.1	Actors	5
2.2	Usage Scenarios	5
2.2.1	Relate Project: Condenser Installation	5
2.2.2	Relate Project: Oasis of Connectivity	6
2.2.3	Energy Project: Trickle of Connectivity	6
2.3	Use Cases	7
2.3.1	Installation	7
2.3.2	Local data Capture	11
2.3.3	Data Transfer	12
2.3.4	Local Service Review	16
2.3.5	Decommissioning	17
2.4	Functional Requirements	19
2.5	Non-functional Requirements	19
2.5.1	Reliability and Security	19
2.5.2	Test Data	19
2.5.3	Performance	20
2.5.4	Supportability	20
2.5.5	Implementation	20
2.5.6	Interface	20
2.5.7	Operation	20
2.5.8	Packaging	20
2.5.9	Legal	20
3	Cloud Storage Synchronization Literature Search	22
3.1	Sensor Networks And Cloud Computing Defined	23
3.2	Sensor Cloud Integration	24
4	References	34

List of Figures

1	Actor relationships.	5
2	Use cases defining Condenser installation.	7
3	Use cases defining Condenser local data capture.	11
4	Use cases defining Condenser data transfer.	12
5	Use cases defining Condenser service review.	16

6	Use cases defining Condenser decomissioning.	17
---	--	----

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, CO2 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.

Figure 2: Use cases defining Condenser installation.

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.
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. This may be done by selecting from connection templates for well-known external storage or by adding in one's own bespoke connection settings (which can be stored for future use as a template).
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.**Quality Requirements:** Major external repositories should be supported through templates (such as Amazon S3, Windows SQL Azure, Rackspace CloudFile, Google Storage and traditional FTP).

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.
4. HPS configures limited data handling rules as appropriate for the given project. Such rules include how data, log files and metadata ought to be compressed, aggregated or removed in times of low local data storage space.

Entry Conditions: No other database is running on the given port.

Exit Conditions: The local database is setup and tested.

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 available to set the disk size for log storage.
- Options are available 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 subscribes Condenser to one or more nodes (data publisher or broadcaster) 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.

Figure 3: Use cases defining Condenser local data capture.

Use Cases:

Capture Data

Participating Actors: Horizon Partner (HP) and/or Project Participant(PP)

Event Flow:

1. The HP or PP initiates data transfer from one or more nodes to Condenser.
2. As Condenser receives data from its subscriptions, it stores the data and metadata (see Capture Metadata use case) to its database.
3. Condenser logs activity as per its logging settings (see Capture Logging Information use case) .

Entry Conditions: Condenser is installed in an environment and nodes are registered with it.

Exit Conditions: Condenser is recording (and possibly transferring) one or more data streams.

Quality Requirements:

- Stored data is time stamped and connected to appropriate metadata.
- Condenser should offer a high degree of reliability regarding data transfers to it (very little to no data loss)
- Condenser should be able to handle receiving secure (encrypted) and unsecured data streams.

Capture Metadata

Participating Actors: Horizon Partner (HP) and/or Project Participant (PP)

Event Flow:

1. Data streams begin to provide data points when the HP or PP initiate data transfer.
2. Each data point is captured in the database along with provenance metadata such as time stamps. Some of these metadata points will be links to descriptive records.

Entry Conditions: Metadata pertaining to given nodes was registered at time of node subscription.

Exit Conditions: Node data is described appropriately.

Quality Requirements: Metadata writing should not interfere with node data capture reliability.

Capture Logging Information

Participating Actors: Horizon Partner (HP) and/or Project Participant (PP)

Event Flow:

1. Data streams begin to provide data points when the HP or PP initiate data transfer.
2. Condenser should make log recordings of events such as error conditions within its own recording and/or transmission schemes, or anomalous node behaviour (such as if nodes go offline unexpectedly).

Entry Conditions: Logging is set up to at a sufficient level of verbosity.

Exit Conditions: Log records can be made of events.

Quality Requirements: Log writing should not interfere with node data capture reliability.

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.



Figure 4: Use cases defining Condenser data transfer.

Use Cases:**Data Transfer Service****Participating Actors:** Horizon Partner (HP) and/or Project Participant(PP)**Event Flow:**

1. The HP or PP initiates data transfer from one or more nodes to Condenser.
2. Data is cached locally.
3. When a data transfer event is initiated Condenser attempts to connect to and authenticate itself with one or more external data storage providers.
4. If there is external network connectivity then data is transferred off site (see time-orientated and event-orientated data transfer use cases) along with metadata see (see metadata transfer use case) and log information (see log file transfer use case). The data may be transferred periodically (see Time-orientated Data Transfer use case) or in response to events (see Event-orientated Data Transfer use case).
5. See Handle unavailable connectivity use case for when connectivity to the external network cannot be established.

Entry Conditions: Condenser is setup and has a stable Internet connection.**Exit Conditions:** Data are transferred to external storage as well as cached locally.**Quality Requirements:** Bandwidth used for data transfer should be minimised.

Time-orientated Data Transfer**Participating Actors:** Horizon Partner (HP) and/or Project Participant(PP)**Event Flow:**

1. The HP or PP initiates data transfer from one or more nodes to Condenser.
2. Condenser attempts to connect to the external data storage system at fixed intervals or at particular points in time as given in the configuration rules – see the set synchronization settings and set external storage connection settings use cases for more information.
3. If a connection is established, data, metadata and log files are transferred to external storage. These will be encrypted if the configuration settings are set for this.

4. If a connection fails the non-transferred data are enqueued for subsequent transfer attempts.

Entry Conditions: Condenser is configured to do time-based synchronization and has a stable Internet connection.

Exit Conditions: Data are transferred to external storage as well as cached locally.

Event-orientated Data Transfer

Participating Actors: Horizon Partner (HP) and/or Project Participant(PP)

Event Flow:

1. The HP or PP initiates data transfer from one or more nodes to Condenser.
2. Condenser attempts to connect to the external data storage system when particular events occur are per its configuration rules – see the set synchronization settings and set external storage connection settings use cases for more information. For instance, an event could be that the local data cache has reached a particular size or that Internet bandwidth is at a particular threshold.
3. If a connection is established, data, metadata and log files are transferred to external storage. These will be encrypted if the configuration settings are set for this.
4. If a connection fails the non-transferred data are enqueued for subsequent transfer attempts.

Entry Conditions: Condenser is configured to do event-based synchronization and has a stable Internet connection.

Exit Conditions: Data are transferred to external storage as well as cached locally.

Metadata Transfer

Participating Actors: Horizon Partner (HP) and/or Project Participant(PP)

Event Flow:

1. The HP or PP initiates data transfer from one or more nodes to Condenser.
2. If there is external network connectivity then metadata is transferred off site. Condenser should deduplicate metadata and only transfer that which has not already been transferred. Duplication could otherwise arise because some metadata will pertain to many data points.

Entry Conditions: Condenser is setup and has a stable Internet connection.

Exit Conditions: Metadata are transferred to external storage as well as cached locally.

Quality Requirements: Bandwidth used for metadata transfer should be minimised.

Log File Transfer

Participating Actors: Horizon Partner (HP) and/or Project Participant(PP)

Event Flow:

1. The HP or PP initiates data transfer from one or more nodes to Condenser.
2. If there is external network connectivity then log files are transferred off site. Condenser should use delta encoding to only transfer portions of log files that have not already been transferred.

Entry Conditions: Condenser is setup and has a stable Internet connection.

Exit Conditions: Log files are transferred to external storage as well as cached locally.

Quality Requirements: Bandwidth used for log file transfer should be minimised.

Handling limited local data storage

Participating Actors: Horizon Partner (HP) and/or Project Participant(PP)

Event Flow:

1. The HP or PP initiates data transfer from one or more nodes to Condenser.
2. If there is external network connectivity difficulties then its is possible that local data storage could fill up.
3. When the local data store is too full then Condenser should compress or aggregate or otherwise eliminate unnecessary data, metadata and log files as per Condenser's configuration settings (see Setup local database use case).

Entry Conditions: Condenser is configured with data handling rules.

Exit Conditions: Data are compressed, aggregated or deleted to maximise space for new data.

Handling limited external data storage

Participating Actors: Horizon Partner (HP) and/or Project Participant(PP)

Event Flow:


1. The HP or PP initiates data transfer from one or more nodes to Condenser.
2. If external storage capacity (or financial constraints) are exceeded then Condenser will act as if the network connection failed and cease data transfer.
3. Condenser will notify its administrator of the failure.

Entry Conditions: Condenser is configured with data handling rules.

Exit Conditions: Condenser suspends further data upload until it is unsuspended, and it notifies its administrator of the failure.

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.

Use Cases:

Metadata Review

Participating Actors: Horizon Project Staff(HPS) and/or Horizon Partner (HP)

Event Flow:

1. HPS or HP accesses the metadata settings web interface for the given device.
2. HPS or HP re-configures Condenser's provenance settings.
3. HPS or HP adds any special metadata for the given project or application.
4. Condenser's new settings are saved as a new version. Old versions are archived for future inspection.

Entry Conditions: Condenser is running and can be connected to through the Internet.

Exit Conditions: Condenser has new provenance settings.

Quality Requirements: Metadata is maintained under version control.

Log Review

Participating Actors: Horizon Project Staff(HPS) and/or Horizon Partner (HP)

Event Flow:

1. HPS accesses the logging settings web interface for the given device.
2. HPS re-configures the logging settings.
3. HPS tests to make sure that activity is being logged appropriately.

Entry Conditions: Condenser is running and can be connected to through the Internet.

Exit Conditions: Condenser's logging settings are updated.

Update configuration settings

Participating Actors: Horizon Project Staff(HPS) and/or Horizon Partner (HP)

Event Flow:

1. HPS accesses the configuration settings web interface for the given device.
2. HPS re-configures the settings.

Entry Conditions: Condenser is running and can be connected to through the Internet.

Exit Conditions: Condenser's settings are updated.

2.3.5 Decommissioning

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

Figure 6: Use cases defining Condenser decommissioning.

Use Cases:**Removing a node**

Participating Actors: Horizon Project Staff(HPS) and/or Horizon Partner (HP)

Event Flow:

1. HPS or HP accesses the node registry web interface for the given device.
2. HPS or HP un-subscribes to a node.
3. Condenser ceases to receive data from the node.

Entry Conditions: A node is registered with Condenser.

Exit Conditions: The node is no longer registered with Condenser.

Removing all nodes

Participating Actors: Horizon Project Staff(HPS) and/or Horizon Partner (HP)

Event Flow:

1. HPS or HP accesses the node registry web interface for the given device.
2. HPS or HP un-subscribes all nodes.

Entry Conditions: At least one node is registered with Condenser.

Exit Conditions: No nodes are registered with Condenser.

Taking Condenser offline

Participating Actors: Horizon Project Staff(HPS) and/or Horizon Partner (HP)

Event Flow:

1. HPS or HP accesses the uninstall web interface for the given device.
2. After a suitable warning HPS or HP initiate uninstall.
3. Condenser attempts to backup all outstanding data, metadata and longs.
4. Condenser removes its local data storage contents and schema.
5. Condenser removes all of its installed software.

Entry Conditions: Condenser is running.

Exit Conditions:

1. All data cached by Condenser is replicated elsewhere.
 2. Condenser is uninstalled leaving no footprint.
-

2.4 Functional Requirements

- Condenser is a cloud gateway infrastructural component for use in software projects requiring data transmission and 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 cached locally until network connection is resumed. Condenser evaluates expected data needs while offline and can be configured to clean, aggregate or delete data to handle limited local drive space.
- Condenser uses a web-based interface to set certain configuration settings.
- 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 CO2 sensors data. Data sets will include: time-series data, discrete 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 in 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.

When dealing with external storage, there is a bounded relationship between cost, capacity and latency. These factors need to be balanced for any Condenser solution.

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, HomeWork and Energy project components.

Condenser should be able to store data externally using a common solutions such as Amazon S3, Windows SQL Azure and FTP.

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 Cloud Storage Synchronization Literature Search

Ubiquitous digital technologies have the potential to capture knowledge about how we behave and about how the effect that the environment and the systems around us have on us. Key enabling technologies have emerged in the form of sensor networks that provide the capabilities to capture, store, transmit and analyse data about ourselves and the world. Such sensor networks have been used for various applications such as environmental monitoring, subject tracking, health care, and intelligent buildings.

Data management becomes crucial in ensuring that the correct lessons are learned from the mighty deluge of data that has emerged. Sensors are able to generate a staggering amount of data and their provenance and storage must be ensured. Furthermore, we have not reached a stage of global network reliability. Instead, coverage is interspersed, systems go on and offline and latencies range from the practically non-existent to maddening delays.

Cloud computing, the provisioning of on-demand computing resources, may be able to help with the storage and computational analysis needs of researchers and industry. Their data are being generated at high frequencies from a multitude of devices that are deployed all about the us, or worn or carried by us. These data however, require entry into the cloud.

This sensor-cloud connection is a problem worth considering. In some cases plain old data transfer solutions (PODTS) may be the best choice. Such users could simply collect sensor reading data into files on a device and direct it to periodically post the files to a data store using a well known protocol such as FTP. This is a viable solution when the users don't care about lost datapoints (perhaps owing to device or network failure), data tampering, data size, cache failure, or possibly even storage provider lock-in. Given these concerns PODTS is probably not appropriate for many situations and projects.

Considering data transfer from the local capture network to the cloud therefore motivates determining answers to these key issues:

- What options are available for transferring and storing data from local capture environments to the cloud? What are the tradeoffs for the various options?
- What cloud storage options are available?
- What existing architectures and system components have already been proposed for moving data into the cloud? What are their advantages and limitations, and is there room for improvement?
- What methods exist for data security and privacy, both in terms of transfer and storage? When should such measures be taken? Is it possible to allow secure analysis of the data within the cloud? How should key management work within this context?

- What metadata ought to be transmitted with the data? What standards are currently available? Are these standards sufficient and do they enable appropriate reasoning about the data and their provenance?
- How do various Horizon projects currently handle these issues and what recommendations can be made to upgrade the projects to best practices?
- How can we model or simulate sensor cloud integration in order to minimise deployment costs and problems?

The following literature review attempts to answer these questions by examining the academic and industrial state-of-the-art in sensor-cloud integration. This section begins with a review of general cloud computing and sensor network ideas and terminology then looks into specific sensor-cloud integration. The review began as an examination into cloud (and general file/data) synchronization options but has exposed various cloud gateway issues such as compression, security, metadata management, compression and external connection handling. The general method of research for this section involved the identification and search of key journals, conferences and industrial player white papers to find relevant high impact works and to critically analyse these to answer the above questions.

3.1 Sensor Networks And Cloud Computing Defined

Sensor networks are a well-researched area of distributed computing. The design of them and their network dynamics have been well-studied and are, in most part, outside the scope of this survey. The curious reader is invited to review key surveys of sensor networks such as [ASSC02], [RM04] and [YMG08]. Of interest, though from sensor networks is the notion of sink nodes. These nodes tend to collect data from the other nodes and forward them on to base stations. For our purposes, we will consider such nodes to be gateways between the local data capture environment and the external storage and analysis environment. We will limit our survey to how these gateway nodes exchange information between the local environment and the external ones.

Cloud computing, the provisioning of on-demand computing resources, has garnered a great deal of recent attention. Cloud computing shares characteristics with earlier distributed computing methods such as cluster computing and grid computing however there are unique features specific to it. Here, we follow the NIST definition of Cloud computing from [MG11]:

“Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be

rapidly provisioned and released with minimal management effort or service provider interaction.”

NIST describes three service models: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). For this survey, we are particularly interested in the data storage facet of Cloud computing. Cloud data storage can be provisioned using any of the service models. For instance, an IaaS solution might be hosting a NoSQL database solution on a series of Amazon EC2 compute nodes, a PaaS solution may be to use a storage web service such as Microsoft SQL Azure, and a SaaS solution may be to use an online solution such as Dropbox.

Cloud storage, according to [AFG⁺09] is likely to become a dominant sensor storage solution because of the potential convenience it offers to data set collectors and users. Cloud storage offers the appearance of infinite on-demand capacity as the provisioning of services to process, analyse and mashup data. They also pointed out, however, key obstacles that must be overcome to realise the vision of cloud storage. Chief amongst these are service availability, service provider lock-in, data confidentiality issues, and data transfer bottlenecks and costs. These issues are addressed throughout this document. [CCB11] provides an IT industry high-level overview of cloud storage and is a good primer regarding existing storage APIs and requirements.

According to [Ram09] different categories of data storage applications have relative likelihoods of using cloud for storage. They posit that transactional mission critical data will remain out of the cloud owing to potential security or privacy attacks on the data. On the other hand they believe that cloud storage is well-suited to analytical and business intelligence data. A key ingredient for success according to them will be being able to operate on encrypted data (which is considered in greater detail below in the section on data security).

We will return to cloud storage options further down in this survey, but first we must survey the current state of connecting sensors and the cloud.

3.2 Sensor Cloud Integration

There has been relatively little attention given to integrating sensor networks with cloud computing compared with other topics in either domain. Of note however are the works of [Bri09], [HSH09], [MF11], [SMT10], [LMHJ10], [KTH⁺11], [Pat11].

A system model and architecture for sensor network cloud integration is proposed in [Bri09]. Each sensor in their system is programmed individually to send data to a gateway node running in the cloud. The gateway receives raw bytes from the sensor network and enques them as packets. It also provides capabilities to reprogram nodes. A storage manager module processes and archives the sensor data. There are also interfaces for data consumption.

Unfortunately, the paper is extremely high level and no further details were given of these components, nor were any test results provided.

A key motivating factor for existing sensor-cloud integration works has been immediate response to sensor readings. Such works however, tend to neglect long-term storage of the readings. While immediate reaction is important (such as for emergency response), long-term storage also has value. For instance, it is important to be able to review long-term changes to sensor readings in a building in order to best maintain it. An integration framework for connecting sensor networks to cloud applications was proposed in [HSH09]. They chiefly focused on determining events from the data and using a pub/sub model to inform clients of the given events. In their framework sensors pass data to gateway nodes that transmit them to a pub/sub broker. A policy engine controls how data is analysed. While this work provides a good starting architecture for responding to immediate sensor events, it does not discuss how data can be stored within the cloud however, brief mention is given to this problem in some follow up work in [LLT⁺10] where they mention (but provide no details about) storing data in a database local to the cloud gateway. They also provide a diagram that includes a "cloud db" object, but provide no information about this. Additional follow on work was reported about wireless sensors transmit data to cloud resources in [KTH⁺11] in order to build healthcare services. For this work a cloud gateway was deployed in the home of a patient and attached to a router. The data were then transmitted to servers running in the cloud. Unfortunately, to explain how this was done, they simply referred back to the previous work. The work in [MF11] was also motivated by immediate response and they showed how to provide immediate sensor data to in cloud data analysis jobs. In their architecture the gateway node (known as the sensor server) maintains an FTP server delivering data stored on each sensor that responds to pull requests. Such a solution is somewhat unsatisfactory however, because it is possible for data loss to occur resulting from limited sensor storage resources.

Another motivating factor for existing sensor-cloud integration works has been the opposite problem to the one that is under examination here, which is how to fetch data from the cloud and cache it on local devices. Such works are worth considering because they integrate cloud and local environments. In the work presented in [SMT10] smart phone data caches are synchronised with cloud storage. Cache synchronisation was handled using the Cimbiosys pull-orientated platform for replication [RRT⁺09]. Such pull-orientated solutions, however, are undesirable owing to the potential for data loss as discussed above.

Integrating sensor monitoring and modeling applications was the focus of [LMHJ10]. They noted that sensor networks data analysis and modeling has tended to be ad hoc. In their architecture the gateway is used to send sensor data to sensor network models that run in the cloud. Unfortunately, no details are provided regarding how the data are transferred, nor about

problems regarding load or connection failure handling.

The work in [Pat11] points out the benefits of sensor network cloud integration: data storage and computation scalability as well as world-wide resource access. Their system, Intortus, interfaces with sensor network gateways to securely store data and make the data available to other applications. In this architecture gateways receive sensor data and dispatch them as XML messages using web services to data storage services running on Google App Engine. Data are encrypted using AES-128 before transmission to the client and decrypted by clients when data are requested. This work does not, however, address issues to do with connectivity, using data in the cloud (besides for backup), handling multiple cloud storage vendors, metadata descriptions, or sensor network cloud simulation.

Although not directly related to sensor network cloud integration, the work in [KGS11] is of interest because they suggest an approach that integrates the local environment with the cloud. They were concerned by great latencies in operation for home environments reliant on cloud services that could become disconnected from the Internet. Their solution, called VS-tore++, monitors resource availability in the local and external environment and directs services to them. This approach has some beneficial characteristics, however, would probably be overkill for most sensor networks whose main tasks will be to simply collect data from devices.

The work presented in [ZHM⁺11] Towards Reliable, Performant Workflows for Streaming-Applications on Cloud Platforms^{***} The following notes are direct copies of text (unless prefaced by ***): - propose and present a scientific workflow framework that supports streams as first-class data, and is optimized for performant and reliable execution across desktop and Cloud platforms - combine streaming data arriving from sensors with historic data available in file archives along with structured collections of weather forecast data that help the large scale computational model make an energy use prediction in near real time - workflow architecture that natively supports the three common data models found in science and engineering applications – files, structured collections and data streams – with the ability to seamlessly transition from one data model to another - logical stream abstractions have to be more robust than simple TCP sockets, given the unreliability and opaqueness introduced by operating in a distributed environment across desktop and Cloud with different characteristics from a typical local area network. Reliability of VMs hosting workflow tasks is another concern to be addressed. Too, there has to be intelligence to avoid costly (both in time and money) duplicate movement of the same logical stream - Streams are a continuous series of binary data - transient unless mapped to another data model - In the future, Cloud providers may provide such streams as IaaS abstraction - registry service that maintains a list of known streams and the endpoints where particular streams are provided - fault resistance: (1) transient or permanent loss of physical network, and (2) loss of virtual machines

in the Cloud or services running on them.

Efficient Storage Retrieval and indexing of time series data

sensor web Semantic Sensor Web New Generation Sensor Web Enablement OGC Sensor Web Enablement Overview and High Level Architecture Semantic Sensor Observation Service Semantically-Enabled Sensor Plug & Play for the Sensor Web The SID Creator A Visual Approach for Integrating Sensors with the Sensor Web

Cloud experiences Adaptive Rate Stream Processing for Smart Grid Applications on Clouds - adaptive rate control to throttle the rate of generation of power events by smart meters, which meets accuracy requirements of smart grid applications while consuming 50% lesser bandwidth resources in the Cloud - power events generated at a peak rate of 1 KB/min from 1.4 M consumers in the Los Angeles Smart Grid will require 2 TB/day of streaming data to be processed and analyzed at an average cumulative bandwidth of 200Mbps - need for an adaptive stream rate control mechanism for generating power usage events - use the difference between the available power capacity at the utility and the current cumulative power usage by the consumers as our throttle function - disadvantages of static publishing rate: - setting too high a rate at which power events are published will cause excess resources (bandwidth, compute VMs for stream processing) to be utilized - setting too low a static rate at which power events are published can cause the utility to miss detecting a breach of a power usage threshold during peak load periods - adaptation logic (throttle) - as the total power usage within the utility approaches total available capacity, power usage events are required more frequently to detect/forecast a peak load event with low latency. - As future work, we plan to evaluate the scalability of the stream processing system with increasing number of VM instances, with the eventual goal of scaling up to 1.4 million smart meter streams that is expected in the Los Angeles Smart Grid. Both throughput and latency of the pipeline need to be measured as the stream rates adapt. Additional factors like availability of compute resources (# of VMs), throughput of the stream processing system, and cost trade-off between Cloud resource usage and power conserved can be incorporated into the throttle policy.

Deploying Database Appliances in the Cloud

Cloud Storage Gateway General resources wikipedia.org A cloud storage gateway is a network appliance or server which resides at the customer premises and translates cloud storage APIs such as SOAP or REST to block-based storage protocols such as iSCSI or Fibre Channel or file-based interfaces such as NFS or CIFS.

CloudStorageSurvey-Avere - make cloud storage appear as local storage - translates cloud storage APIs to file interfaces or storage transport protocols - gateway resides at customer site - can be dedicated appliance/server, virtual appliance/server, or s/w application - features: compression, deduplication, encryption, caching, backup and recovery - Provides useful table columns for

comparing cloud grteways, but unfortunately the table is incomplete (only the Avere FXT was examined and these cost \$50000+)

cloudStorage.pdf redundant cloud storage with octopus

<http://gladinet.blogspot.com/2010/09/how-to-pick-cloud-storage-gateway.html>

- pick a cloud storage gateway, there are several questions: - 3 forms: s/w, virt. appliance, physical - which supported cloud storage providers - how to store data on cloud: - block level - file level - what encryption/compression algo used

<http://www.cloudswitch.com/page/now-everybody-wants-a-cloud-gateway>

- gateway simplifies the integration and management of cloud resources so people can get on with using the cloud rather than struggling to make it work

<http://www.cloudswitch.com/blog/category/Cloud-> a well-designed cloud gateway needs to: - guarantee security - gateway should allow users and administrators to monitor and manage applications running in a cloud as if they were running locally, using existing tools and policies in a single, integrated environment - protect roles and access

www.nasuni.com/blog/28dirtysecrets5weaknessesofcloudstorage - gateway or device (virtual or physical) that needs to be installed at your site - benefits: - improved performance - security - compression - dedup - WAN optimization - cache data - limitations - internet connection (can be mitigated using a bulk data migration service) - Caches are not perfect and sometimes you will get read and write misses - write miss: the space you need to write to in the cache is temporarily full and has not been freed up by offloading/sending data to the cloud - ability to feed data to the appliance most likely is significantly faster than your Internet connection - You can't get at your data without the appliance because the appliance adds a level of security - cloud storage gateways change the data written to them - compression, dedup, encrypted, chunked for parallelism

<http://www.nasuni.com/blog/32blocksvsfileswhichapproachisbetterforcloud>

- block-based gateway - gateway works directly with blocks, the 512-byte fundamental units of storage - raw chunks of storage on a drive - Applications assume that blocks exist in local, fast storage - cloud: connections between the applications and the storage can become strained (latency issues) - Intelligent caching is really, really difficult - Block-based gateways restore at the volume level, essentially restoring the blocks in a volume to a previous point in time - file-based gateway - Files provide a clear picture of the relationships among the blocks that support them - File-based gateways provide restore capabilities at the file, directory, or complete file system level - best for unstructured data Commercial offerings Ctera - cloud storage gateways that provides shared storage, cloud backup, folder synchronization and remote access in single devices - only uploads diffs/changed data - de-duplication - user bandwidth throttling (time based or constant) - 128-bit SSL - data encrypted using 256-bit AES & fingerprinted by 160 bit SHA-1 - Restore

all or individual files - Rules-based CIFS, WebDAV and rsync based sync - FTP - Dashboard - Resource utilization - Event logs (system, access, backup, sync, audit) - Backup schedules - Email alerts - Automatic updates - Config backup/restore - rack mounted and plug form factors - remote centralised monitoring/management through a portal (SaaS or installed software) Nasuni - Nasuni Filer, a storage controller that runs in a datacenter as a hardware appliance or a virtual appliance - primary storage - built-in backup - offsite backup - all data and metadata is stored in the cache - cache snapshots - each file is encrypted, compressed and backedup to the Nasuni Service - cache management removes rarely re-used docs to respect the cache capacity - portal provides - volume & cache metrics - alerts/notifications - remote file access - control file sharing in the network - restore from snapshot Citrix <http://www.opensourcerack.com/2011/05/25/citrix-netscaler-cloud-gateway-a-product-tour/> - gateway to provision application access to users by administrators Gladinet - Windows clients - Cloud server (<http://gladinet.blogspot.com/2011/10/introducing-gladinet-cloud-server.html>) - on premise gateway - file server that allows you to mount different cloud storage services such as Amazon S3 (default), Windows Azure, Google Storage, EMC Atmos into a Windows File Server - Map Drive to the Attached Cloud Storage Folder - Cache - Connected to account - Mount service accounts (such as Amazon S3) - Active directory integration - Cloud for teams service Atmos <http://www.emc.com/collateral/software/white-papers/h9505-emc-atmos-archit-wp.pdf> - store, manage, and protect globally distributed, unstructured content at scale Intel cloud builders guide <http://www.emc.com/collateral/software/data-sheet/h5770-atmos-ds.pdf> Open <http://openstack.org/> - backend oriented (S3 competitor) Nimbus Project <https://github.com/nimbusproject/nimbus> <http://www.nimbusproject.org> - cloud computing for science - allows a client to lease remote resources by deploying virtual machines (VMs) on those resources and configuring them to represent an environment desired by the user Cumulus Cumulus: an open source storage cloud for science - storage cloud implementation - compatible with the Amazon Web Services S3 REST API - quota management - used against many existing clients (boto, s3cmd, jets3t, etc) - open source implementation of the Amazon S3 REST AP - configure Cumulus with existing systems such as GPFS, PVFS and HDFS, in order to provide the desired reliability, availability or performance trade-offs - upload data to the cloud, monitor its status, and download it from the storage cloud as needed. - provides an image store for Nimbus compute clouds - S3 interface allows clients to write, read, and delete objects or organize them into buckets - Authentication mechanisms, based on request signature by symmetric key - authorization database - Redirection module used to handle scalability - implemented in python - the concept of a storage cloud is a fusion between data transfer and storage management Eucalyptus <http://www.cca08.org/papers/Paper32-Daniel-Nurmi.pdf> - opensource software framework for cloud computing that implements IaaS

JetS3t <http://jets3t.s3.amazonaws.com/index.html> - free, open-source Java toolkit and application suite for Amazon Simple Storage Service (Amazon S3), Amazon CloudFront content delivery network, and Google Storage for Developers - 5 applications: - Cockpit: graphical application for transferring files, viewing and managing the contents of an Amazon S3 or Google Storage account - Synchronize: command-line application for synchronizing directories on your computer with an Amazon S3 or Google Storage account - Files are copied to/from the Amazon S3 or Google Storage service - by default, only new or changed files are transferred - synchronize.properties file - accesskey and secretkey which define AWS access credentials - upload.max-part-size - files larger than this value will be split into smaller parts no larger than the value and uploaded as Multipart Uploads - upload.ignoreMissingPaths - Synchronize will perform an upload despite missing or unreadable source files - files can be gzipped or encrypted during synchronization - upload.metadata - Custom metadata to apply when uploading new files to S3 - no action option to generate a report of what will happen - force sync when files are up-to-date - Gatekeeper: servlet that acts as an authorization service running on a Service Provider's server to mediate access to S3 accounts - CockpitLite: graphical application that Service Providers with S3 accounts may provide to clients or customers without S3 accounts - Uploader: graphical application that Service Providers with S3 accounts may provide to clients or customers without S3 accounts

s3cmd <http://s3tools.org/s3cmd> - command line tool for uploading, retrieving and managing data in Amazon S3 - conditional/unconditional transfer - http and https - GPG encryption

Synchronization Tools General resources Synch comparison chart [en.wikipedia.org/wiki/Comparison](http://en.wikipedia.org/wiki/Comparison_of_file_synchronization_software)

Common synch tools rsync - invented by Andrew Tridgell - first announced on 19 June 1996.[1] Rsync 3.0 was released on 1 March 2008 - requires an rsync server running rsync daemon - algorithm - synchronizes files and directories from one location to another while minimizing data transfer using delta encoding

- application - standard Linux utility - ported to Windows (via Cygwin), Mac OS - rsync is capable of limiting the bandwidth consumed during a transfer - supports compression and decompression - SSH Efficient Algorithms for Sorting and sync

DeltaCopy - open source Windows' wrapper of rsync - GPL 3

Unison - Open Source by Benjamin C. Pierce - Mac, Windows, Linux <http://www.stanford.edu/~pgbovine/unisonguide.htm> - batch option - ssh What's in Unison File Synchronization with Unison - two-way rsync with a bit of revision control mixed in - uses the rsync algorithm to keep network traffic down and should be tunneled through SSH over untrusted networks - Unison is programmed in OCaml - Own tests - Unison will not sync a file currently being written to

Synkron - written in C++ and uses the Qt4 libraries - GPL v2 - multiplat-

form - synchronising multiple folders at the same time - can sync subfolders
 - file exclusions maintained in a blacklist; filters - scheduler - restore files
 overwritten during sync

DirSync Pro - GPL3 - Java (JRE 1.6.0 and higher.) - Bi-directional
 (Two way) and mono-directional (One way) synchronization mode - Option
 to synchronise subdirectories recursively - Synchronizes files/folders any file
 system - handling time-stamps - Schedule Engine - gui and create a command
 line and save it to a batch file - logging/reporting facilities - no encryption

Open source online tools SparkleShare - opensource alternative to Drop-
 box <https://github.com/hbons/SparkleShare> <http://sparkleshare.org/> - Sa
 collaboration and sharing tool - linux, mac, android - Documents synchro-
 nised to all peers when changes are made - notifications when someone has
 made a change - have as many projects as you'd like - use as much space
 as you'd like - run on as many hosts as you'd like - uses the GIT system
 as its backbone - store files on own Git server (SSH), Gitub, Gitorious
[http://www.makeuseof.com/tag/sparkleshare-great-open-source-alternative-](http://www.makeuseof.com/tag/sparkleshare-great-open-source-alternative-dropbox-linux-mac/)
[dropbox-linux-mac/](http://www.makeuseof.com/tag/sparkleshare-great-open-source-alternative-dropbox-linux-mac/)

dvcs-autosync <http://gitorious.org/dvcs-autosync> explanatory article: <http://www.mayrhofer.eu.o>
 autosync - open source replacement for Dropbox/Wuala/Box.net/etc - based
 on distributed version control systems (DVCS) - Git is is being tested most
 thoroughly as the backend storage, but other DVCS such as Mercurial are
 also supported - A single Python script monitors the configured directory
 for live changes, commits these changes to the DVCS (such as git) and syn-
 chronizes with other instances using XMPP messages - linux only (since it
 relies on inotify)

Syncany - open-source dropbox alternative - encrypts the files locally -
 plug-in based storage system supporting FTP, Amazon S3, Google Storage,
 Rackspace Cloud Files, WebDAV, Picassa, Box.net - Currently immature
 (alpha not yet released - Windows & MAC versions are even further behind)

Online services Windows only symform [http://www.symform.com/resilient-](http://www.symform.com/resilient-storage-architecture.aspx)
[storage-architecture.aspx](http://www.symform.com/resilient-storage-architecture.aspx) Windows only LiveMesh [http://www.codeproject.com/Articles/37200/Cloud-](http://www.codeproject.com/Articles/37200/Cloud-Based-Source-Control-using-Live-Mesh-and-Git)
[Based-Source-Control-using-Live-Mesh-and-Git](http://www.codeproject.com/Articles/37200/Cloud-Based-Source-Control-using-Live-Mesh-and-Git)

SpiderOak Free with limited functionality by Spideroak Mac, iPhone,
 iPad, Windows, Linux, Online, Android - zero-knowledge privacy approach -
 SpiderOak operates its own hardware and data centers without outsourcing -
 backup, sync, share - SpiderOak keeps historical versions of every file - 2048
 bit RSA and 256 bit AES - \$10/mo/100 GB

Dropbox Free with limited functionality Mac, iPhone, iPad, Windows,
 Linux, Online, Android, Blackberry - Dropbox transfers just the parts of a
 file that change - Very common - Manually set bandwidth limits - Sharing -
 \$10 / month/50GB; \$20 / month/100GB; - Secure Sockets Layer (SSL) and
 AES-256 bit encryption

Ubuntu One Free with limited functionality by Canonical Ltd. Windows,
 Linux, Online, Android, iPhone \$3/mo/20GB - Ubuntu One's data storage

is simply a set of publically accessible CouchDBs.

SugarSync Free with limited functionality by SugarSync Mac, iPhone, iPad, Windows, Win Mobile, Online, Android, S60, Blackberry - backup any folder - versioning -real-time upload of changes - no file size limits - business plans

Wuala Free by LaCie Mac, iPhone, iPad, Windows, Linux, Online, Android - encrypts the data on your computer before it is uploaded - sync across multiple computers - share - Business and personal licenses - Business: 100 GB for 5 users - Eur 279 /year

CrashPlan Free with limited functionality by Code42 Mac, Windows, Linux, Online - plans per computer or per GB

Box.net Free with limited functionality by Box.net iPhone, iPad, Windows, Online, Android, Blackberry - personal, business and enterprise pricing - business: \$15/user/month - 1TB; 2GB file limit - Enterprise: \$? unlimited storage; 2 GB file limit - file versioning - comments and discussions - tasks - full text search Syncplicity Free with limited functionality by Syncplicity Mac, iPhone, Windows, Online - personal edition: \$15/mo/50GB - Business edition: \$45/mo/unlimited storage - no file size limit TeamDrive Free by TeamDrive Systems GmbH Mac, Windows, Linux, Online - store on their TeamDrive cloud, your own server, or WebDAV servers

ZumoDrive Free with limited functionality by Zecter Inc. Mac, iPhone, iPad, Windows, Linux, Online, Android - 10 GB \$2.99 / month; 25 GB \$6.99 / month; 50 GB \$9.99 / month; 100 GB \$19.99 / month ; 200 GB \$37.99 / month ; 500 GB \$79.99 / month

Tonido Free by CodeLathe Mac, iPhone, iPad, Windows, Win Phone 7, Linux, Online, Android, Blackberry - personal cloud software - make files and media in that computer available anywhere - remotely access or share your music, photos, calendar, files, and more from any computer or mobile phone - TonidoPlug - Connect your TonidoPlug to your router, then connect any external USB hard drive to your plug to access your files, music and media stored in that hard drive from anywhere via any browser. Minus Free by Minus Inc Mac, iPhone, iPad, Windows, Win Phone 7, Linux, Online, Android, Android Tablet, Blackberry - drag and drop

Duplicati Open Source by Kenneth Skovhede, mortenmie, et al Windows, Linux - c# LGPL - storing the initial copy and then differentials to go from the initial version to the current - works with a number of different backends, eg. FTP, WEBDAV and S3, Rackspace Cloud File - built-in AES-256 encryption and backups can be signed using GNU Privacy Guard - built-in scheduler command-line client

Acid Rain Open Source Mac, Windows, Linux, Online - uses Mercurial - run own server or use a Mercurial hosting service - Acid Rain Server is a Linux distribution based on Open Suse 11.3 Sharebox - immature (in development) - a filesystem that will synchronize arbitrary data between several machines - developed in c - storage through git-annex lipsync - a lightweight

commandline service that securely synchronizes your data - linux only (server and client) - openSSH - rSync-based

Personal Cloud servers Pogoplug - £40 - S/w for PC, Mac, Linux, iPhone, iPad, Android Iomega ix2-200 and ix4-200d network storage devices

(Distributed) File Systems General Resources Cloud-based synchronization of distributed file system hierarchies Systems HekaFS <http://git.fedorahosted.org/git/?p=CloudF>
Coda GlusterFS RFS RFSa network file system for mobile devices and the cloud [102] addresses three main issues as follows: maintaining seamless connection between users and clouds, controlling cache consistency, and supporting data privacy WebDAV FUSE

Cloud storage services general resources Cloud storage survey (unpub.)
An automated approach to cloud storage service selection MetaStoragecamera-ready Cloud Storage: Adoption, Practice and Deployment - Best practices *
Choosing a provider: redundancy, fail-over, versioning, data back-up, mgmt console, pricing * local storage as well as cloud? Providers Amazon S3 Windows Azure ATandT Synaptic Storage Rackspace CloudFile Peer1 CloudOne Nirvanix Mezeo Google Storage traditional FTP WebDav server 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.

Database Replication Unstructured CouchDB MongoDB

Weak connections weaklyconnectedusers Peer-to-peer Data Replication Meets Delay Tolerant Networking Deduplication Building a high-performance deduplication system

Cloud data security Challenges in Secure Sensor-Cloud Computing Toward Securing Sensor Clouds <http://gigaom.com/cloud/the-cloud-meets-the-law-where-wikileaks-went-wrong/> Structured Encryption and Controlled Disclosure Computing Arbitrary Functions of Encrypted Data Every Cloud has An Encrypted Lining: The Effectiveness of Cryptography in Cloud Computing Fully homomorphic encryption using ideal lattices Distributing data for secure database services Addressing cloud computing security issues Silverline toward data confidentiality in storage-intensive cloud applications [92] Energy-efficient incremental integrity for securing storage in mobile cloud computing Lightweight and Compromise Resilient Storage Outsourcing with Distributed Secure Accessibility in Mobile Cloud Computing [93] Authentication in the clouds: a framework and its application to mobile users Every Cloud has An Encrypted Lining- The Effectiveness of Cryptography in Cloud Computing

SN/Cloud simulation Avrora Scalable Sensor Network Simulation CloudSim2010
simulating wireless sensor networks with omnet++

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, and M. Zaharia. Above the clouds: A berkeley view of cloud computing. Technical Report EECS-2009-28, EECS Department, University of California, Berkeley, 2009.
- [ASSC02] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393 – 422, 2002.
- [Bri09] S. Briefings. Can we plug wireless sensor network to cloud? *SET-Labs Briefings*, 7(7):33, 2009.
- [CCB11] D. Connor, P.H. Corrigan, and J.E. Bagley. Cloud storage: Adoption, practice and deployment. Technical report, Storage Strategies NOW, April 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.
- [KGS11] S. Kannan, A. Gavrilovska, and K. Schwan. Cloud4home – enhancing data services with @home clouds. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 539 –548, june 2011.
- [KTH⁺11] Asad Masood Khattak, Phan Tran Ho Truc, Le Xuan Hung, La The Vinh, Viet-Hung Dang, Donghai Guan, Zeeshan Pervez, Manhyung Han, Sungyoung Lee, and Young-Koo Lee. Towards smart homes using low level sensory data. *Sensors*, 11(12):11581–11604, 2011.
- [LLT⁺10] Xuan Hung Le, Sungyoung Lee, Phan Truc, La The Vinh, A.M. Khattak, Manhyung Han, Dang Viet Hung, M.M. Hassan, M. Kim, Kyo-Ho Koo, Young-Koo Lee, and Eui-Nam Huh. Secured wsn-integrated cloud computing for u-life care. In *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*, pages 1 –2, jan. 2010.
- [LMHJ10] K. Lee, D. Murray, D. Hughes, and W. Joosen. Extending sensor networks into the cloud using amazon web services. In *Networked*

Embedded Systems for Enterprise Applications (NESEA), 2010 IEEE International Conference on, pages 1–7, nov. 2010.

- [MF11] J. Melchor and M. Fukuda. A design of flexible data channels for sensor-cloud integration. In *21st International Conference on Systems Engineering (ICSEng), 2011*, pages 251–256. IEEE, 2011.
- [MG11] P. Mell and T. Grance. The nist definition of cloud computing. Technical Report Special Publication 800-145, Recommendations of the National Institute of Standards and Technology, Computer Security Division Information Technology Laboratory National Institute of Standards and Technology Gaithersburg, MD 20899-8930, September 2011.
- [Pat11] Sharada Krishna Patil. Usable, lightweight and secure, architecture and programming interface for integration of wireless sensor network to the cloud. Master’s thesis, Graduate School of The Ohio State University, 2011.
- [Ram09] R. Ramakrishnan. Data management in the cloud. In *Data Engineering, 2009. ICDE ’09. IEEE 25th International Conference on*, page 5, 29 2009–april 2 2009.
- [RM04] K. Romer and F. Mattern. The design space of wireless sensor networks. *Wireless Communications, IEEE*, 11(6):54 – 61, dec. 2004.
- [RRT⁺09] Venugopalan Ramasubramanian, Thomas L. Rodeheffer, Douglas B. Terry, Meg Walraed-sullivan, Ted Wobber, Catherine C. Marshall, and Amin Vahdat. Cimbiosys: A platform for content-based partial replication. nsdi. Technical Report MSR-TR-2008-116, Microsoft Research, 2009.
- [SMT10] Patrick Stuedi, Iqbal Mohomed, and Doug 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, MCS ’10*, pages 1:1–1:8, New York, NY, USA, 2010. ACM.
- [YMG08] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292 – 2330, 2008.
- [ZHM⁺11] D. Zinn, Q. Hart, T. McPhillips, B. Ludascher, Y. Simmhan, M. Giakkoupis, and V.K. Prasanna. Towards reliable, performant workflows for streaming-applications on cloud platforms.

In *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, pages 235 –244, may 2011.