tsc++

USER GUIDE

v1.3

August 13, 2009

Contents

Chapter 1

Fact Sheet

tsc++ is an implementation of the TSC (Triple Space Computing) paradigm. It is an extension of Tuple-based computing which has been introduced in parallel programming languages (e.g. Linda) to implement communication between parallel processes. Processes can write, read and take (read and delete) data tuples form a persistent space. Thus the communication is completely asynchronous. Triple Space Computing extends this paradigm by using P2P and Semantic Web technologies like RDF(triples) and SPARQL.

This project aims at developing Triple Space Computing as a communication and coordination middleware framework for Semantic Web and Semantic Web services. It uses Sesame or Owlim for persistent storage of RDF triples. The network communication and coordination is provided through the JXTA framework.

tsc++ is based on the TSC [13] project and was initiated to overcome licence problems caused by the proprietary CORSO layer used within the TSC project. *tsc++* is available at [15].

1.1 Change Log

1.1.1 Change Log for Release 1.3

New features

- introduction of subspace logic
- changed the pub/sub mechanism to just send the currently advertised template
- added support for local full sparql queries

Bug fixes

- major bugfix in subscribe for RDV peer
- bugfix in restoring spaces

Maintenance

- startup if kernel is not already start up

Known Problems

- issues regarding firewalls and NAT

1.1.2 Change Log for Release 1.2

New features

- introduction of a new REST remote Kernel for testing
- subscribe returns a list of advertised templates

Bug fixes

- compressed messages can now be received if compress messages property is not set
- minor bugfixes in SpaceManager
- advertise subscribe mechanism repaired
- bugfix in random walk logic
- empty template not allowed any more

Maintenance

- getting started application updated
- userguide updated

Known Problems

- issues regarding firewalls and NAT

1.1.3 Change Log for Release 1.1

New features

- with this version the support of Yars as data access layer ends
- full support for RDFXML, TURTLE and N3 for preloading and triple serialisation
- extended properties
- Linux support for getting started application
- introduction of an independent STI world peer (JXTA)
- read and query functionality changed [refer to userguide]
- method to take triples
- error handling extended
- implementation of an access log
- integrated Sesame 2.1.3

Bug fixes

- fixed Owlim delete persistency issues

Maintenance

- extend Owlim support
- metadata manager implementation for introduction of subspaces

Known Problems

- issues regarding firewalls and NAT

1.1.4 Change Log for Release 1.0.1

New features

- create space and write initial file (containing RDF triples) to space
- integrated Sesame 2.0.1
- extended JXTA properties
- advertise subscribe sample application in getting started
- userguide for new version
- javadoc upgrade
- error handling extended

Bug fixes

- fixed usage of Sesame, Owlim or Yars via config file for TSKernel

Maintenance

- extend Owlim support
- support for RDFXML, TURTLE and N3 preload format

Known Problems

- Sesame performance issues writing big sets of triples

1.1.5 Change Log for Release 1.0

New features

- integrated Sesame v2.0 (http://openrdf.org) and Yars (http://sw.deri.org/2004/06/yars/) store
- integrated Owlim 3.0 beta (http://www.ontotext.com/owlim/) store, attention - beta version for project internal use only
- using JXTA (https://jxta.dev.java.net/) for network communication and coordination
- compression of XML messages for saving bandwidth, deflate algorithm
- properties logic refactored to a central Props store

Bugfixes

- none

Maintenance

- extend Owlim support
- support for RDFXML, TURTLE and N3 preload format

Known Problems

- Sesame performance issues writing big sets of triples

1.2 Interfaces, Standards, Requirements

Nature: Java library without graphical user interface, middleware solution

Platform: JDK v1.5

Interfaces: Java API, various external libraries

Supported Standards: RDF, SPARQL (partially through templates), TURTLE, RDFXML, N3 (triple formats used for file loading)

Required Libraries:

- http://jxta.org/ the following .jar files contained in jxse-lib-2.5.zip
  - jxta.jar
  - javax.servlet.jar
  - org.mortbay.jetty.jar
  - bcprov-jdk14.jar
- http://logging.apache.org/log4j/
  - log4j.jar
- http://yaslibrary.sourceforge.net/logging.shtml
  - slf4j-api-1.4.3.jar
  - slf4j-log4j12-1.4.3.jar
- for usage of Sesame as data access layer http://www.openrdf.org/
  - openrdf-sesame-2.1.3-onejar.jar
- for usage of Owlim as data access layer http://www.ontotext.com/owlim/swiftowlim-3.0.beta7-sesame-2.0.zip
  - trree-3.0.beta7.jar
  - owlim-3.0.beta7.jar

1.3 Licensing

tsc++ License Agreement:

ICT - Technologie Park Innsbruck, 2nd Floor
Technikerstrasse 21a
6020 Innsbruck, Austria

tsc++ is a free software. One can redistribute and/or modify it under the the terms of the GNU Lesser General Public License (LGPL). Further licensing information is available at http://tsc.sti2.at#license

Licensing of Third Party Libraries:
Please refer to the respective project homepages.

1.4 Installation and Usage

The latest version of *tsc++* is available at http://tsc.sti2.at#download. The Project is distributed as a ZIP archive containing the full source code, changelog, the tsc++[current version].jar file and a sample application. Also a jar-only version and the full javadoc as ZIP are available. The external libraries have to be downloaded separately and copied into the lib folder. Additionally a REST interface to communicate with a remote kernel installed on an STI server can be downloaded. This kernel is ideal for becoming familiar with *tsc++* without downloading the full distribution (since the jar file is small and no external libraries must be downloaded). The RESTKernel is fully compatible with the regular distribution. A full installation and configuration description is provided in Chapter 5 of this userguide.

1.5 Future Plans

- Storage optimisation
- Full Owlim 3.0 support
- Implementation of a metadata management
- New network layer discovery and distribution algorithms

Chapter 2

Introduction

The World Wide Web ('WWW' or simply the 'Web') is the medium of the 21th century. With at least 13.97 billion pages [18] it holds more information then any library could ever keep. Because of this huge amount of information and the variety of types (text, images, and other multimedia) it becomes more and more difficult to use the Web efficiently. Tim Berners-Lee's endeavors (i.e. HTML and HTTP) led into the creation of the Web and its ongoing development.
Technically the Web is based on interlinked hypertext documents and runs over the Internet. To navigate through the Web a so-called Web browser is needed. In the beginning of the Web desktop computers were used to gain access to Web resources. Nowadays notebooks are very handy and are used in various ways. Last but not least more and more mobile devices come into play. Although the Mobile Web is still confronted with interoperability and usability problems [16] it can become reality. W3C [17] started the Mobile Web Initiative [7] to overcome those problems.
Current Web research is very much about adding semantics to the Web. This enrichment with metadata will allow machines to understand the actual meaning of the content and in return will provide support in accessing and processing this information. The Web for humans can be transformed into a Web for machines - the Semantic Web. Another trend is using the Web not only as a content provider for humans but also as a global platform for distributed computation and integration of various applications. This is possible when using Web Services. Semantic Web Services (combining Semantic Web and Web Service technology) could lead the Web to its full potential - allowing automated service discovery, service configuration, combination as well as automated negotiation. Web Service technologies currently cannot provide this state of automization and integration. The communication of Web Service applications is based on message exchange which requires strong coupling in terms of reference and time. Web Services also do not follow the Representational State Transfer (REST, [23]) principles. The Web is based on the 'persistently publish and read' paradigm. Information on the Web can be reused and read multiple times but that is not possible with traditional Web Services because of synchronous information exchange [24]. All together Web Services do not have much in common with the Web.

2.1 Space Based Computing

Space based computing is an approach to address the widely recognized problems with Web Services. It also can help to overcome heterogeneity in communication and cooperation [14]. A space acts like a 'blackboard' - everybody can read/write from/to it. This allows complex

message exchange to be replaced by simple read/write operations. 'Blackboard'-communication no longer requires that the partner processes 'know' each other (*anonymity*) and that they run simultaneously (*asynchronity*) [20]. In the beginning the concept of spaces was used in parallel programming to allow a simple and decoupled communication between parallel processes. Processes are able to read and write tuples from/to a persistent space. Such spaces are called Tuple Spaces and contain tuples that are visible to all processes. The limitations of Tuple Spaces however are concerned with the lack of support for namespaces, semantics, and structures for describing a tuple's content.
A Tuple Space can be extended into a Triple Space by adding the notion of triples *<subject, predicate, object>* describing content and semantics of information. The Resource Description Framework (RDF) provides the fundamentals for such a shared space based on RDF triples. Using Triple Space based computing allows interaction between machines to be achieved at Web scale.
There exist two frameworks which follow the new paradigm based on Triple Spaces - Triple Space Computing (TSC FIT-IT, [13]) and Triple Space Communication (TripCom, [12]). Both frameworks consist of multiple layers but implement the coordination layer, as one of the most important parts, rather differently. TSC's coordination layer uses CORSO (Coordinated Shared Data Objects, [4]) while TripCom's coordination layer is based on DNS and P-Grid [9].

2.2 Semantic Repositories

Semantic repositories are systems similar to database management systems allowing storage, querying and management of structured data. The main difference between DBMS's and SR's is the use of ontologies as semantic schemata which allows automatic reasoning. The storage solutions are often based upon semantic web technologies like RDF and SPARQL (an SQL like query language for RDF data). Two different repositories are integrated in the *tsc++* framework.

- Sesame 'is an open source RDF framework with support for RDF Schema inferencing and querying. Sesame has been designed with flexibility in mind. It can be deployed on top of a variety of storage systems like relational databases, tools to developers to leverage the power of RDF and RDF Schema, such as a flexible and several query languages, of which SeRQL is the most powerful one' [11].
- OWLIM 'is another semantic repository, packaged as a storage and inference layer (SAIL) for Sesame. OWLIM uses the TRREE engine to combine RDFS, OWL DLP, and OWL Horst support with high-performance reasoning and reliable persistence strategy' [8].

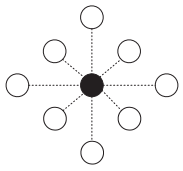
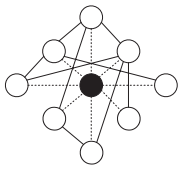
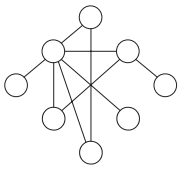
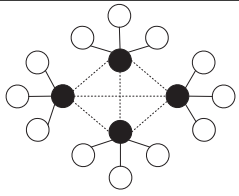
The used storage solution can be be configured in the *general.properties* file. Further information following in Chapter 5 Installation and Configuration.

2.3 Peer-to-Peer Systems

P2P networks are meeting points for peers and are according to the Merriam-Webster dictionary a peer is 'one that is of equal standing with another; especially: one belonging to the same group'. The Peer-to-Peer Working Group (P2Pwg) defines P2P as 'sharing of computer resources and services by direct exchange'.
Today P2P networks cause up a significant traffic portion of the whole Internet traffic [22], nevertheless because peers communicate directly the network bandwidth is better utilized.

The disruptive technology of P2P networking encounters an enormous growth – various popular software solutions today use P2P technology to guarantee their services (e.g. Skype). P2P became famous because of a file-sharing system called Napster – this was in 1999. Other than traditional client-server networks, where content providers are strictly divided from content consumers, P2P networks are highly decentralized and any peer is providing and consuming content [1]. Because peers are not distinguished between content providers (servers) and content consumers (client) they are often named 'servent' (**serv**er + cli**ent**) [21].
P2P networks are subject to frequent changes – peers joining and leaving the network – therefore not well shared data can become unavailable, a result of the dynamic environment.
A P2P network creates an overlay network over the Internet – spanning a new network over an already existing one. The Internet with its specific network details is not discussed here, but most design decisions in P2P networks only become clear after understanding the Internet as a communication network.
Peer-to-Peer systems either use structured or unstructured network designs [21]. This categorization can be fine grained – the subset of unstructured P2P systems are known as *centralized*, *pure* and *hybrid* P2P (see: Figure 2.1) whereas structured P2P systems are *DHT-based*.
Centralized P2P enforces central entities to provide the service to the other peers. Removing all central entities leads to the loss of functionality. Pure P2P lacks central entities, and is often thought of when talking about P2P. Hybrid P2P is combination of centralized and pure P2P, specifying two roles of peers. The centralized behavior is modeled by 'dynamic central' peers working in a cooperative manner to provide the service to the other peers. Peers in charge of the 'dynamic central' role are also often called 'superpeers' or 'ultrapeers' in the research literature whereas the peers in charge of the 'default' role are only called peers. JXTA uses a superpeer structure in form of Rendezvous (meeting point for peers) and Relay (forwarding messages through firewalls and NAT) peers and is therefore a hybrid P2P system.
DHT-based P2P belongs to structured P2P systems and guarantees, in the absence of network failures, perfect recall.

Table 2.1: Network Topologies

Client-Server	**Peer-to-Peer**		
	Unstructured P2P		
	Centralized P2P	Pure P2P	Hybrid P2P
			

[1] resources, information or data

Chapter 3

Overview

3.1 TSKernel

To start a new *tsc++* instance a new TSKernel (*at.sti2.tsc.kernel*[1]) object must be instanced. The following classes are mandatory for running a kernel:

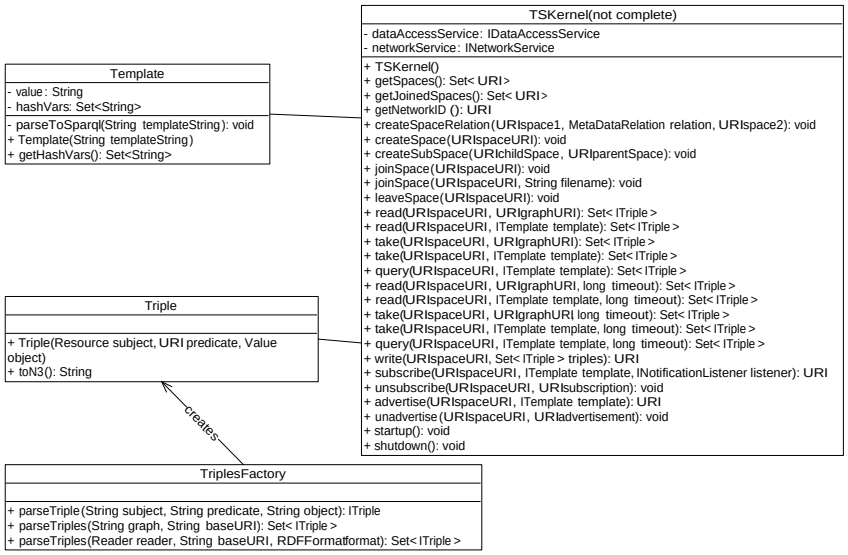


Figure 3.1: main classes to use

Every TSKernel instance must be started using the startup() method. The createSpace(spaceURI) method creates a new space or the createSubSpace(URI childSpace, URI parentSpace) creates a new subspace. Afterwards joinSpace(spaceURI) enters the created space. The joinSpace(spaceURI, filename) method joins a space and writes an initial set of triples, contained in a file, to the space. This feature can be used to load a backup file, an ontology or an initial dataset.

NOTE: to stop a kernel the shutdown() method must be executed due to persistency issues. The stored state (joined spaces and triples) will be recovered on startup(). Spaces will be normalized (if the URI does not end with '/' the slash will be appended) in order to guarantee the correct serialisation of graphs.

[1]Package names in this document are provided in italic font type

The leaveSpace() method leaves a previously joined space. The SEND_TRIPLES_ON_LEAVE parameter in *jxta.properties* causes the kernel to send all his triples, stored in the left space locally, to a random kernel in the space. A kernel can create/join multiple spaces and return a list of the already joined ones by calling getJoinedSpaces(). The operations that can be performed are mentioned in detail in Section 4.2.

3.2 RESTKernel

The RESTKernel (*at.sti2.tsc.kernel* offers the possibility to become familiar with *tsc++* without downloading the full distribution. The RESTKernel jar contains all necessary interfaces and classes to run a kernel instance and therefore is compatible with TSKernel. Instead of creating a kernel locally the kernel's functionality is wrapped and the operations are performed on a remote TSKernel on a STI server. The communication follows the REST [10] (Representational state transfer) principle.

NOTE: There is only 1 TSKernel available on the server that is shared by every RESTKernel instance. So a new RESTKernel() call will NOT create a new kernel but will access the instance (see: Figure 3.2) shared on the server. The RESTKernel can be used to get a general overview over the API or as additional resource for tests with local kernels.

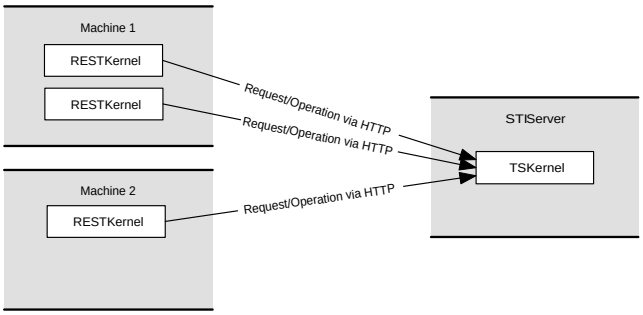


Figure 3.2: RESTKernel communication

3.3 Triple

A Triple object is the Java internal representation of a RDF triple created by using the TriplesFactory. The factory offers the possibility to create a single triple object by passing three String values to the parseTriple(subject, predicate, object) method. The parseTriples(graph/reader, baseURI) methods enable the system to parse a set of triples out of a String/reader (e.g. FileReader). It is important to choose the correct RDF format in order to parse the data correctly. A triple can be retrieved in N3 format using toN3().

3.4 Template

To read, query and take triples from spaces SPARQL templates can be used. Templates are Strings proposed to specify the WHERE-clauses of SPARQL queries. More details on templates and their matching rules are given in Section 4.3.

3.5 Exceptions

The following Exceptions can occur during the execution of a kernel object:

TSException generic Triple Space Exception (all following exceptions are derived from this). Is thrown if null values occur as method paramenters.

ConnectionException is thrown if problems occur during connection to the Rendezvous or Relay peer.

PropertiesException is thrown if a property is set to an invalid value

SpaceException generic space exception

- SpaceAlreadyExistsException is thrown when trying to create a space that already exists. **NOTE:** due to the nature of Peer-to-Peer sometimes the space advertisement will not be found in time (also occurs if TIMEOUT_DISCOVER in *jxta.properties* is too small) leading to the situation that a space can be created twice or more often.
- SpaceNotExistsException is thrown if a join, read, write, take or query operation is performed on a space that does not exist. **NOTE:** due to the nature of Peer-to-Peer sometimes the space advertisement will not be found in time (also occurs if TIMEOUT_DISCOVER in *jxta.properties* is too small) leading to the situation that an existing Space cannot be joined.
- SpaceNotJoinedException is thrown when trying to write triples to a not joined space
- SubSpaceAlreadyExistsException is thrown when trying to create a subspace that was already created before

SPARQLQueryException is thrown when a full SPARQL query can not be joined

MalformedTemplateException is thrown if a template String cannot be transformed into a SPARQL query.

TripleParseException is thrown if triples cannot be parsed from a source. This often occurs if the set RDF format is wrong. Also Strings containing malformed URIs will cause this exception.

3.6 Subspaces

A space can have multiple subspaces attached to it. Every query, read or take operation will be performed on the spaces and/or its subspaces (see: Section 4.2).

3.7 Metadata manager

The metadata manager component is responsible for storing and handling the relations between spaces as well as handling the suspace logic. The following relations can be defined between spaces:

- s0 isSusbspaceOf s1 inverseOf s0 hasSubspace s1
- s0 hasSubspace s1 inverseOf s0 isSusbspaceOf s1

- s0 seeAlsoSpace s1
- transitive + symetric property: s0 isSimilarTo s1
- transitive + symetric property: s0 isRelatedTo s1

To define a relation the createSpaceRelation(URI spaceURI1, MetaDataRelation relation, URI spaceURI2) is used.

3.8 Access log

The access log component logs operations (refer to 4.2) performed on a space. It can be activated in the *general.properties* file. If a space is created/joined the associated access log space will be automatically created/joined. Consider a space ts://www.example.org/example/ and its access log space ts://www.example.org/example/accesslog/ (the access log space URI consists of the space URI plus 'accesslog/'). The access log information consists of triples of the form *<client's JXTA id, date, access type>*. Following access types will be logged:

- http://www.tripcom.org/ontologies/tsonto#AccessLogEntry/type/read when read (URI or template) is performed
- http://www.tripcom.org/ontologies/tsonto#AccessLogEntry/type/write when write is performed
- http://www.tripcom.org/ontologies/tsonto#AccessLogEntry/type/query when query is performed
- http://www.tripcom.org/ontologies/tsonto#AccessLogEntry/type/take when take (URI or template) is performed

The access log information is deleted when leaving a space. It is stored persistently on shutdown of the kernel.

Chapter 4

Architecture

4.1 UML chart

The UML classdiagram (see: Figure 4.2) shows an overview of *tsc++'s* three main layers. The first represents the interface for integration of the middleware. An instance of TSKernel *(at.sti2.tsc.kernel)* implements the ITripleSpace *(at.sti2.tsc)* interface offering the main operations which can be performed on the space. A detailed description follows in this Chapter.
The ITripleSpace *(at.sti2.tsc)* uses two different layers for data storage and network issues. This separation guarantees the exchangeability of either the storage or the network layer. Currently two different storage solutions are integrated in the *tsc++* framework. For the network communication a P2P based solution is used.

Storage Layer handles the storage of RDF triples in a semantic repositories. The major operations can be found in the IDataAccess *(at.sti2.tsc.dataaccess)* interface. In the current implementation one can choose between Sesame or Owlim. The data is stored persistently on the Harddisk. The usage of well tested frameworks guarantees good scalability and performance. The IDataAccessService *(at.sti2.tsc.dataacess.*.service)* is a wrapper for starting IDataAccess *(at.sti2.tsc.dataaccess)* operations in a Java internal threadpool. *tsc++* is a highly multithreaded application.

Network Layer handles the network communication and coordination for exchanging data. Currently JXTA, a P2P Java framework, is executing this task. The communication and coordination operations are almost the same ones as for the data access storage layer. The write Operation simply calls the data access write method. So writing information locally is the same as writing information to a space or in other words locally stored information is accessible remotely through the space. Every space has its own unique main kernel (Rendezvous or RDV peer) which can be configured via WAIT_FOR_RDV_CONNECTION Property in the *jxta.properties* file. If this parameter is false the kernel should become RDV peer. If it is set true kernels joining a space should communicate via the already started RDV peer. The RDV peer handles the communication between kernels. The information is shared as follows:

- Flood Based communication means that the RDV peer sends the query or read information (gotten by a kernel in the space) to all kernels in the space. The querying or reading kernel gets a response from each kernel satisfying the query.

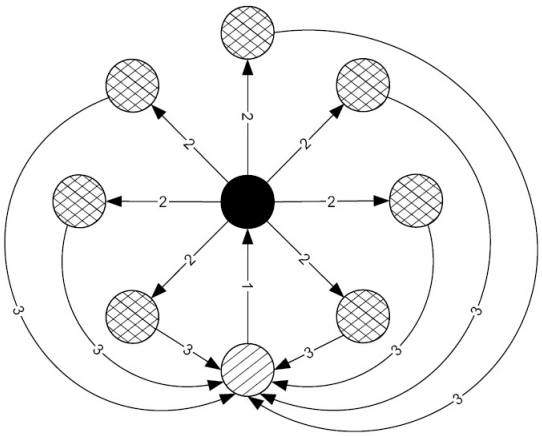


Figure 4.1: Flood Based communication (where black node is Rendezvous peer)

- Random Walk Based communication means that a query 'walks' to randomly choosen kernels. The Walk has a maximum amount of hops to pass before termination.
- Biased Random Walk Based communication is an extension of Random Walk communication. The walk is biased by previous responses from other kernels.
- Publish/Subscribe Based communication is used for solving the information dissemination problem.

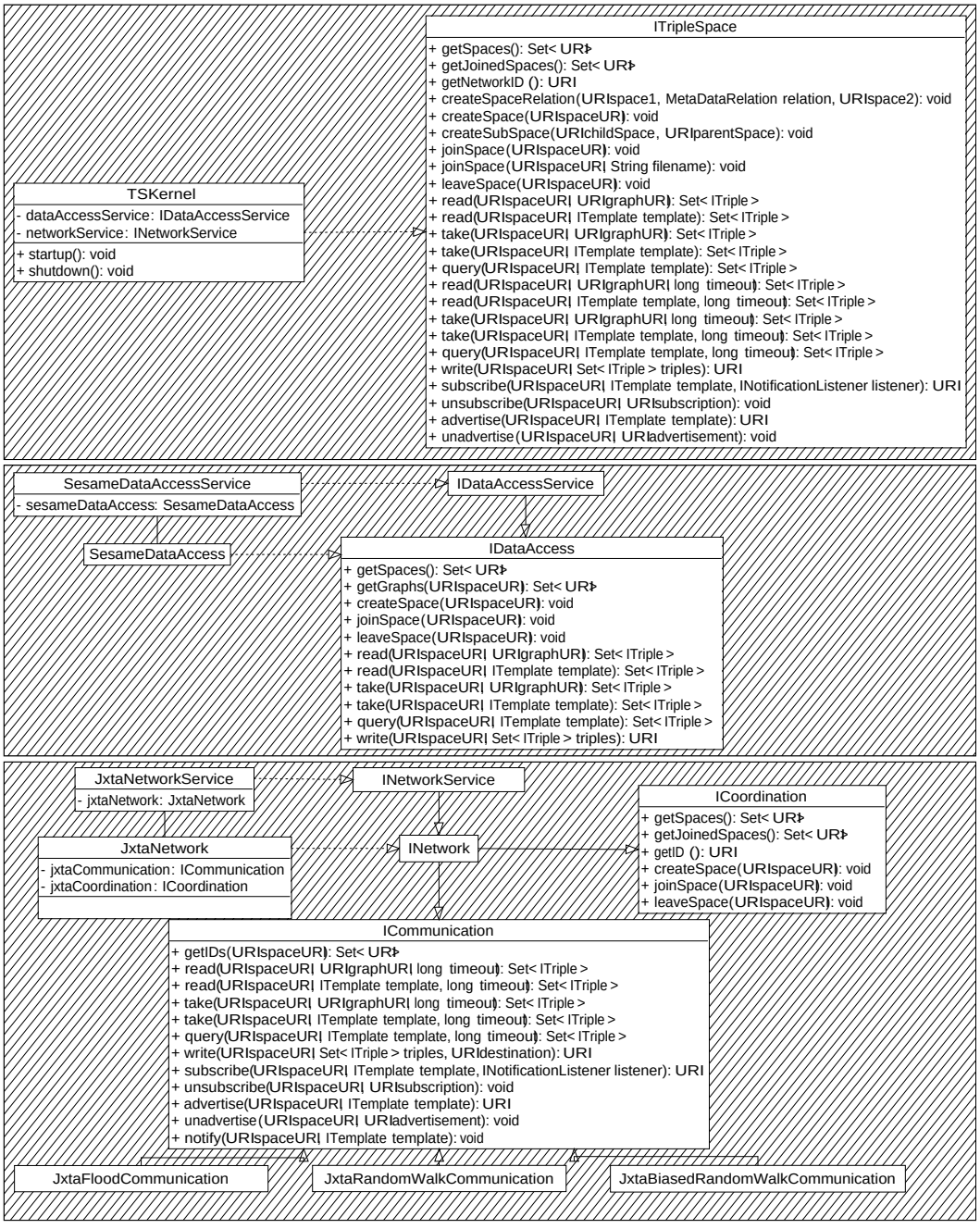


Figure 4.2: classdiagram

4.2 Operations

4.2.1 Abstractions

Abstractions in respect of Triple Space Computing are highlighted in Figure 4.3 meaning:

- spaces contain graphs
- graphs contain triples / are sets of triples
- triples are of the form *<subject, predicate, object>*

Moreover a space is identified by a URI (space URI, e.g. ts://www.example.org/space/) and all graphs contained in a space are distinguishable by using graph URIs as identifiers (e.g. ts://www.example.org/space/178b8c6b-cf3d-4055-914b-9227b159e0cf/).
Assume there are infinite sets of U (URIs), B (blank nods), and L (literals). The elements in $U \cup B \cup L$ are RDF *terms*. A triple $(v_1, v_2, v_3) \in (U \cup B) \times U \times (U \cup B \cup L)$ is an RDF triple. The value v_1 is called the *subject*, v_2 the *predicate* and v_3 the *object*. An RDF graph is a set of RDF triples.
Some example types are:

- URIs: http://members.sti2.at/membername
- Blank Nodes: _:r
- Plain Literals: '18', 'Member'
- Typed Literals: '18'^^xsd:int, 'Member'^^xsd:string

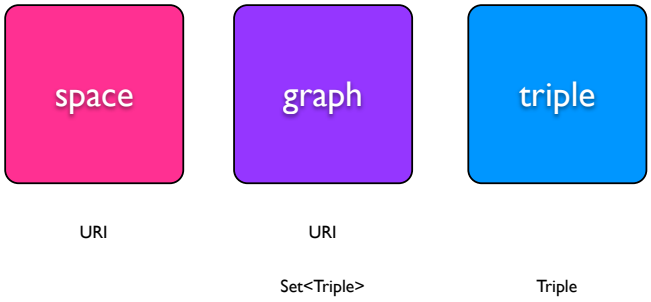


Figure 4.3: abstractions

4.2.2 Write, Read and Take vs. Query

All operations are based on the exchange of triples. Before this is possible triples need to be represented in the system as objects or in other words as a collection of (triple) objects. Whenever speaking about triples or a graph as a collection of triples – a set (Set, *java.util.Set*) of triple objects (Triple, *at.sti2.tsc.data.triple*) – is meant. Since there are various formats for representing the subject-predicate-object relation, serializing and deserializing of triple objects needs to be possible. String representations (serialized triples) of various formats (N3, NTRIPLES, RDFXML, TRIG, TRIX, TURTLE) can be read or written – TriplesSerializer (*at.sti2.tsc.data.triple*) uses RDFFormat (*org.openrdf.rio*) for doing so.

Writing triples implies storing them in the RDF store and leads to the creation of a URI as a valid identifier. This URI is called graph URI because it represents a RDF graph. Such a graph URI consists of the concatenation of two parts a space URI and a graph name. A graph name is a UUID[1] that can be created from specified set of triples or randomly. The graph name being a UUID enforces the whole graph URI being a unique identifier.
Creating the graph name from a set of triples has the advantage of creating the same UUID for the same triples. Writing the same triples with just another graph URI is therefore not possible – it will create the same graph URI and will not be accepted.
A graph URI as well as templates, as shown later, can be used for reading triples. Triples are stored within certain contexts (space URI, graph URI) – speaking in simple terms – data is tagged in order to allow precise retrieval later. Triples will always be in one space – by being tagged with a space URI – and in one graph – by being tagged with a graph URI. The kernel offers following operations:

- Set<ITriple> read(URI space, URI graph)
- Set<ITriple> read(URI space, ITemplate template)

There are two possible ways of performing a read operation, either by specifying a graph URI or a template. In contrast to the query operation read looks for only one graph, meaning a set of triples that is in the context of one graph URI. By providing a graph URI (e.g. ts://www.sti2.at/space/178b8c6b-cf3d-4055-914b-9227b159e0cf) these triples have to be in the defined context. Providing a template enforces the search for triples in the context of one graph URI (not all graph URIs) matching the template. More on template matching in Section 4.3. Read will return the same triples that were written together. Read is a operation for getting a maximum of one (existing) graph in one space (even if more graphs matching the template exist in the space). Retrieval is always first of all governed by remote querying, as publication is a priori local. In that way the the retrieval algorithm queries primarily unknown data. If no data is found remotely the local store will be searched through. The specifiede space is always queried first. If no result is found the space's subspaces are traversed until a result is found and returned. A non successful read operation returns null.

- Set<ITriple> take(URI space, URI graph)
- Set<ITriple> take(URI space, ITemplate template)

The operation take equals a read and delete (has the same semantics as read). The triples read will be deleted locally in the kernel they are read from. It is possible to take via a graph URI as well as by template. If no result is found the space's subspaces are traversed until a result is found and returned. If nothing is found remotely the kernel will try to take triples from the local store returning null when no graph is found.

- Set<ITriple> query(URI space, ITemplate template)

In order to perform a query operation a template needs to be specified. In contrast to the read operation query looks for triples in all graphs (triples that are in contexts of different graph URIs) matching the template. Query will return only single triples (not graphs) that match a template. Query is a operation for getting one (new) graph in one space. It combines triples found remotely with the ones found locally. The COMBINE_QUERY_RESULTS flag has to be

[1] Universally Unique Identifier [6]

set in *jxta.properties* in case triples from different remote kernels should be combined. If the property is set to false the method will return as soon as the first triples form a remote kernel arrive. Triples found in all subspaces will also be contained in the return. A non successful query returns an empty set (size = 0) of triples.

NOTE: Triples can be read, taken and queried from a non joined space. This mechanism is called temporary join! The kernel joins the space, executes the operation and leaves the space automatically afterwards.

- URI write(URI space, Set<ITriple> triples)

A write causes the triples to be written to a space locally. The triples can then be accessed remotely by other kernels.

4.2.3 Advertise and Subscribe

The following kernel operations are:

- URI advertise(URI spaceURI, ITemplate template)
- void unadvertise(URI spaceURI, URI advertisement)
- URI subscribe(URI spaceURI, ITemplate template, INotificationListener listener)
- void unsubscribe(URI spaceURI, URI subscription)

An alternative for solving the information dissemination problem is motivated by the *publish/subscribe* paradigm: publishers provide (advertise) data and subscribers express interest in (subscribe to) certain data. Consequently, subscribers listen for publication events, or they are notified if they occur, in order to react. This is a particularly interesting procedure from an applications point of view, as it further decouples the involved parties.
The participants in the publish/subscribe communication model are coordinated by the help of JXTA advertisements (Advertisement, *net.jxta.document*) that are used for indexing at the RDV peers (see: Figure 4.4). The advertisement, as exemplified in Table 4.1, contains a template and was created because a *peer X* executed the advertise (operation 300) method. The advertisement in Table 4.2 was created because a *peer Y* executed the subscribe (operation 301) method. *peer Y* would be notified – INotificationListener (*at.sti2.tsc.network.jxta.communication.event*) will be fired – that a publisher is available for the certain template. The unadvertise/unsubscribe methods take an URI (i.e. an <ID>, created before by advertise/subscribe, Table 4.1 and 4.2) in order to abandon a pub/sub position for a certain template.

Table 4.1: JXTA Advertisement, advertise Template

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ts:TemplateAdvertisement>
<ts:TemplateAdvertisement xml:space="default" xmlns:jxta="http://jxta.org">
  <ID>
    urn:jxta:uuid-3748B9A994274D20A1D3A00F3AF97265DD6651578FB64D5F92748395F3C3087801
  </ID>
  <spaceURI>
    ts://www.example.org/space/sti/
  </spaceURI>
  <template>
    &lt;http://www.example.org/TSC> &lt;http://www.sti2.org/#has> ?o.
  </template>
  <operation>
    300
  </operation>
</ts:TemplateAdvertisement>
```

Table 4.2: JXTA Advertisement, subscribe Template

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ts:TemplateAdvertisement>
<ts:TemplateAdvertisement xml:space="default" xmlns:jxta="http://jxta.org">
  <ID>
    urn:jxta:uuid-3748B9A994274D20A1D3A00F3AF97265662D4ABBED8B4BB6836A8ECED0714E3201
  </ID>
  <spaceURI>
    ts://www.example.org/space/sti/
  </spaceURI>
  <template>
    &lt;http://www.example.org/TSC> &lt;http://www.sti2.org/#has> ?o.
  </template>
  <operation>
    301
  </operation>
</ts:TemplateAdvertisement>
```

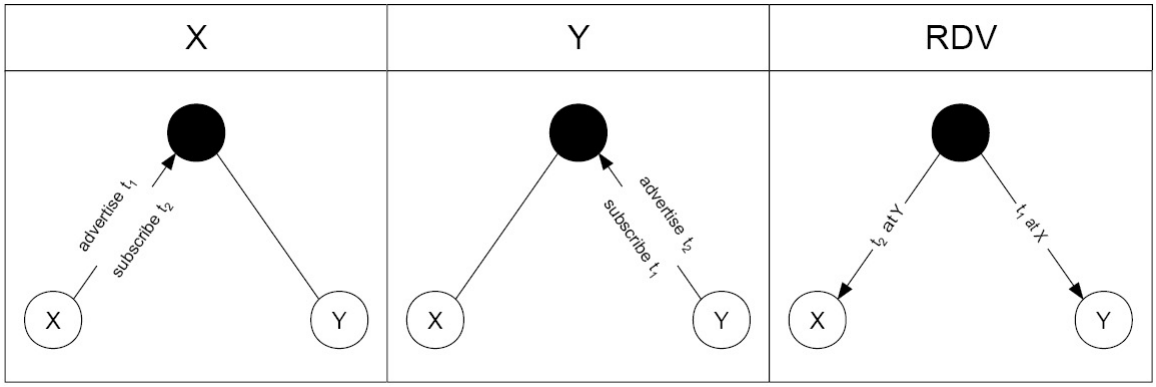


Figure 4.4: Advertise Subscribe Model Overview

4.3 Templates

To read, query and take triples from spaces SPARQL templates can be used. Templates are proposed to specify the WHERE-clauses of SPARQL queries. Full fledged SPARQL queries are (currently) not supported. Such queries would allow to override the behavior of a triple space which is not intended. Templates are path expressions specifying a sequence of adjacent triple patterns in the query string.

Instead of

CONSTRUCT { *?s ?p ?o* }
WHERE {
GRAPH <*ts://www.sti2.at/space/*> { *?s ?p ?o* } .
}

a template only contains the last part of the query string

?s ?p ?o

but executes the same query.

4.3.1 Template Matching Example

The following set of triples is stored in space ts://www.example.org/space/.

Table 4.3: Set of triples stored in a space — N3 notation

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<http://members.sti2.at/membername> rdf:type foaf:Person .
<http://members.sti2.at/membername> foaf:name "Member Name" .
<http://members.sti2.at/membername> foaf:firstName "Member" .
<http://members.sti2.at/membername> foaf:phone <callto://membername> .
<callto://membername> rdf:type <http://skype.com> .
```

- **query**

  <http://members.sti2.at/membername>
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#firstName>
  ?o.

  returns

  <http://members.sti2.at/membername>
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#firstName>
  "Member".

- **read** *<http://members.sti2.at/membername>*
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#firstName>
  ?o.

  returns

  <http://members.sti2.at/membername>
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  <http://xmlns.com/foaf/0.1/Person>.

  <http://members.sti2.at/membername>
  <http://xmlns.com/foaf/0.1/name>
  "Member Name".

  <http://members.sti2.at/membername>
  <http://xmlns.com/foaf/0.1/firstName>
  "Member".

  <http://members.sti2.at/membername>
  <http://xmlns.com/foaf/0.1/phone>
  <callto://membername>.

  <callto://membername>
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  <http://skype.com>.

- **query**

  <http://members.sti2.at/membername>
  <http://xmlns.com/foaf/0.1/firstName>
  <http://skype.com>.

  returns

  no match will be found, an empty set of triples with **size = 0** will be returned

- **read**

  <http://members.sti2.at/membername>
  <http://xmlns.com/foaf/0.1/firstName>
  <http://skype.com>.

  returns

  no match will be found, **null** will be returned

Table 4.4: General Matching Rules

```
exact match
<http://tsc.sti2.at/>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://usefulinc.com/doap/Project>.

matching all triples containing specified subject
<http://tsc.sti2.at/>
?p
?o.

matching all triples containing specified subject and predicate
<http://tsc.sti2.at/>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
?o.

matching all triples containing specified subject and object
<http://tsc.sti2.at/>
?p
<http://usefulinc.com/doap/Project>.

matching all triples containing specified predicate and object
?s
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://usefulinc.com/doap/Project>.

matching all triples containing specified object
?s
?p
<http://usefulinc.com/doap/Project>.

matching all triples containing specified predicate
?s
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
?o.

match every triple
?s
?p
?o.
```

4.4 Storage details

Note: all storage solutions store data persistently. The main differences between them are:

4.4.1 Sesame

- 'Sesame can be deployed on top of a variety of storage systems (relational databases, in-memory, filesystems, keyword indexers, etc.).' [11]
- Sesame offers the smallest memory overhead because only the write operation is directly consuming memory. The read, take and query operations access the information stored on the Harddisk.
- store can be cleaned up (reset to empty store on startup) by setting CLEAN_DIR to true in *sesame.properties* file.
- Sesame creates a main folder called store with a folder for spaces containing triples.

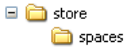


Figure 4.5: Sesame Folders

4.4.2 Owlim

- 'SwiftOWLIM is the fastest RDF(S) and OWL engine.' [8]
- Owlim storage must be started with Java Virtual Machine Argument *'-Xmx256m'* for extension of the Java Heap Space. It can be assumed that Owlim is memory hungry but fast for a huge dataset.
- store can be cleaned up (reset to empty store on startup) by setting CLEAN_DIR to true in *owlim.properties* file
- Owlim creates a main repositories folder containing the same folder structure as Sesame plus a System folder. Owlim's store is bigger than Sesame's store for the same set of data because internal ontology statements are implicitly stored and loaded.

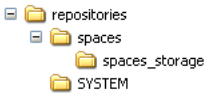


Figure 4.6: Owlim Folders

4.5 JXTA Protocols

JXTA consists of a set of six protocols[2] designed for Peer-to-Peer (P2P) network computing. These protocols work together to allow the discovery, organization, monitoring and communication between peers. The most important protocol, the Peer Resolver Protocol (PRP), will be

[2]https://jxta-spec.dev.java.net/JXTAProtocols.pdf

discussed further. PRP provides a generic query/response interface and is used by *tsc++* to perform its operations (see: Section 4.2). JXTA realizes this query/response interface by exchanging XML messages (see: Figure 4.7). The triples contained in a messages can be compressed (using a deflate [3] and BASE64 encoding [4]) to save bandwidth by setting the COMPRESS_MESSAGES flag in *general.properties*. A resolver query message (see: Figure 4.5) is sent to a named handler on one or more peers that are members of the peer group (the space). In *tsc++* this handler is named *tsc:TSCResolverMessageHandler*. Each query message contains a query (i.e. <Query>, a string), has an originator (i.e. <SrcPeerID>, a peer id) and can be identified (i.e. <QueryID>, an integer number). An identifier, included in the response, is used by the querier to match replies. A resolver response message (see: Figure 4.6) is the response (i.e. <Response>, a string) to a resolver query message (i.e. <QueryID>, query message reference). Queries (i.e. <Query>, Figures 4.7, 4.8 and 4.9) and responses (i.e. <Response>, Figures 4.10, 4.11, 4.12 and 4.13), in *tsc++*, are valid XML instead of their ordinary string representation – being an obvious benefit of information sharing and system interoperability.
The example from Section 5.2 from the query/response point of view means:

- Kernel1 sends a query (see: Figure 4.7) and waits for the response from Kernel2 (see: Figure 4.10).
- Kernel2 sends a query (see: Figure 4.8) and waits for the response from Kernel1 (see: Figure 4.11).
- Kernel3 sends a query (see: Figure 4.9) and waits for the responses from Kernel1 and Kernel2 (see: Figure 4.12 and 4.13).

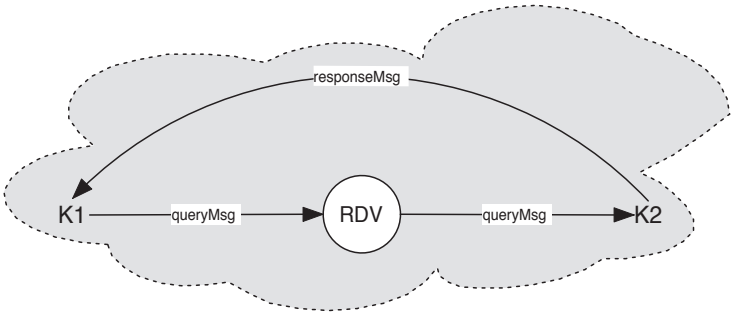


Figure 4.7: Query/Response Interface

[3] A lossless compression Algorithm.
[4] A special kind MIME content transfer encoding.

Table 4.5: JXTA ResolverQuery message

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jxta:ResolverQuery>
<jxta:ResolverQuery xmlns:jxta="http://jxta.org">
  <HandlerName>
    tsc:TSCResolverMessageHandler
  </HandlerName>
  <QueryID>
    1876477922
  </QueryID>
  <HC>
    0
  </HC>
  <SrcPeerID>
    urn:jxta:uuid-5961626164616261...C5233A403
  </SrcPeerID>
  <Query>
    ...
  </Query>
</jxta:ResolverQuery>
```

Table 4.6: JXTA ResolverResponse message

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jxta:ResolverResponse>
<jxta:ResolverResponse xmlns:jxta="http://jxta.org">
  <HandlerName>
    tsc:TSCResolverMessageHandler
  </HandlerName>
  <QueryID>
    1876477922
  </QueryID>
  <Response>
    ...
  </Response>
</jxta:ResolverResponse>
```

Table 4.7: Kernel1 read-operation, ResolverQuery message

```
<?xml version="1.0"?>
<!DOCTYPE tsc:TSCResolverQueryMessage>
<tsc:TSCResolverQueryMessage>
  <spaceURI>
    ts://www.example.org/space/
  </spaceURI>
  <template>
    &lt;http://www.example.org/TripCom>
    &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    &lt;http://usefulinc.com/doap/Project>.
  </template>
  ...
  <operation>
    201
  </operation>
</tsc:TSCResolverQueryMessage>
```

Table 4.8: Kernel2 read-operation, ResolverQuery message

```
<?xml version="1.0"?>
<!DOCTYPE tsc:TSCResolverQueryMessage>
<tsc:TSCResolverQueryMessage>
  <spaceURI>
    ts://www.example.org/space/
  </spaceURI>
  <template>
    &lt;http://www.example.org/TSC>
    &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    &lt;http://usefulinc.com/doap/Project>.
  </template>
  ...
  <operation>
    201
  </operation>
</tsc:TSCResolverQueryMessage>
```

Table 4.9: Kernel3 query-operation, ResolverQuery message

```
<?xml version="1.0"?>
<!DOCTYPE tsc:TSCResolverQueryMessage>
<tsc:TSCResolverQueryMessage>
  <spaceURI>
    ts://www.example.org/space/
  </spaceURI>
  <template>
    ?s
    &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    &lt;http://usefulinc.com/doap/Project>.
  </template>
  ...
  <operation>
    202
  </operation>
</tsc:TSCResolverQueryMessage>
```

Table 4.10: Kernel1 read–operation, ResolverResponse message

```
<?xml version="1.0"?>
<!DOCTYPE tsc:TSCResolverResponseMessage>
<tsc:TSCResolverResponseMessage>
  <spaceURI>
    ts://www.example.org/space/
  </spaceURI>
  <template>
    &lt;http://www.example.org/TripCom>
    &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    &lt;http://usefulinc.com/doap/Project>.
  </template>
  <triples>
    &lt;http://www.example.org/r1>
    &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    &lt;http://www.example.org/deliverable> .
    &lt;callto://membername>
    &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    &lt;http://skype.com> .
    &lt;http://members.deri.at/membername>
    &lt;http://xmlns.com/foaf/0.1/phone>
    &lt;callto://membername> .
    &lt;http://www.example.org/TripCom>
    &lt;http://www.example.org/hasInConsortium>
    &lt;http://www.uibk.ac.at> .
    &lt;http://www.example.org/TripCom>
    &lt;http://www.example.org/hasDeliverable>
    &lt;http://www.example.org/r1> .
    &lt;http://members.deri.at/membername>
    &lt;http://xmlns.com/foaf/0.1/firstName>
    "Member" .
    &lt;http://www.example.org/TripCom>
    &lt;http://purl.org/dc/elements/1.1/title>
    "Triple Space Communication" .
    &lt;http://members.deri.at/membername>
    &lt;http://xmlns.com/foaf/0.1/currentProject>
    &lt;http://www.example.org/TripCom> .
    &lt;http://www.example.org/TripCom>
    &lt;http://www.example.org/hasInConsortium>
    &lt;http://www.cefriel.it> .
    &lt;http://www.example.org/TripCom>
    &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    &lt;http://usefulinc.com/doap/Project> .
    &lt;http://members.deri.at/membername>
    &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    &lt;http://xmlns.com/foaf/0.1/Person> .
    &lt;http://members.deri.at/membername>
    &lt;http://xmlns.com/foaf/0.1/name>
    "Member Name" .
  </triples>
  ...
  <operation>
    201
  </operation>
</tsc:TSCResolverResponseMessage>
```

Table 4.11: Kernel2 read-operation, ResolverResponse message

```
<?xml version="1.0"?>
<!DOCTYPE tsc:TSCResolverResponseMessage>
<tsc:TSCResolverResponseMessage>
  <spaceURI>
    ts://www.example.org/space/
  </spaceURI>
  <template>
    &lt;http://www.example.org/TSC>
    &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    &lt;http://usefulinc.com/doap/Project>.
  </template>
  <triples>
    &lt;http://members.deri.at/membername>
    &lt;http://xmlns.com/foaf/0.1/phone>
    &lt;callto://membername> .
    &lt;http://members.deri.at/membername>
    &lt;http://xmlns.com/foaf/0.1/firstName>
    "Member" .
    &lt;http://www.example.org/TSC>
    &lt;http://www.example.org/hasInConsortium>
    &lt;http://www.tuwien.ac.at> .
    &lt;http://www.example.org/TSC>
    &lt;http://purl.org/dc/elements/1.1/title>
    "Triple Space Computing" .
    &lt;http://members.deri.at/membername>
    &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    &lt;http://xmlns.com/foaf/0.1/Person> .
    &lt;http://www.example.org/TSC>
    &lt;http://www.example.org/hasDeliverable>
    &lt;http://www.example.org/r1> .
    &lt;http://www.example.org/r1>
    &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    &lt;http://www.example.org/deliverable> .
    &lt;callto://membername>
    &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    &lt;http://skype.com> .
    &lt;http://members.deri.at/membername>
    &lt;http://xmlns.com/foaf/0.1/currentProject>
    &lt;http://www.example.org/TSC> .
    &lt;http://members.deri.at/membername>
    &lt;http://xmlns.com/foaf/0.1/knows>
    &lt;http://www.example.org/r2> .
    &lt;http://www.example.org/TSC>
    &lt;http://www.example.org/hasInConsortium>
    &lt;http://www.uibk.ac.at> .
    &lt;http://www.example.org/r2>
    &lt;http://www.w3.org/2000/01/rdf-schema#seeAlso>
    &lt;http://www.debruijn.net/foaf.rdf> .
    &lt;http://members.deri.at/membername>
    &lt;http://xmlns.com/foaf/0.1/name>
    "Member Name" .
    &lt;http://www.example.org/TSC>
    &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    &lt;http://usefulinc.com/doap/Project> .
  </triples>
  ...
  <operation>
    201
  </operation>
</tsc:TSCResolverResponseMessage>
```

Table 4.12: Kernel3 query-operation, ResolverResponse message

```
<?xml version="1.0"?>
<!DOCTYPE tsc:TSCResolverResponseMessage>
<tsc:TSCResolverResponseMessage>
  <spaceURI>
    ts://www.example.org/space/
  </spaceURI>
  <template>
    ?s
    &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    &lt;http://usefulinc.com/doap/Project>.
  </template>
  <triples>
    &lt;http://www.example.org/TripCom>
    &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    &lt;http://usefulinc.com/doap/Project> .
  </triples>
  ...
  <operation>
    202
  </operation>
</tsc:TSCResolverResponseMessage>
```

Table 4.13: Kernel3 query-operation, ResolverResponse message

```
<?xml version="1.0"?>
<!DOCTYPE tsc:TSCResolverResponseMessage>
<tsc:TSCResolverResponseMessage>
  <spaceURI>
    ts://www.example.org/space/
  </spaceURI>
  <template>
    ?s
    &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    &lt;http://usefulinc.com/doap/Project>.
  </template>
  <triples>
    &lt;http://www.example.org/TSC>
    &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    &lt;http://usefulinc.com/doap/Project> .
  </triples>
  ...
  <operation>
    202
  </operation>
</tsc:TSCResolverResponseMessage>
```

JXTA creates 4 folders for storing information:

- msg folder containing
  - a subfolder (named after the space URI hash code) containing in and out XML Messages
  - hops.txt
  - received_requests.txt
  - received_responses.txt
  - sent_requests.txt
  - sent_responses.txt
- adv folder containing advertisement XML Messages (named after the advertisement id hash code)
- logs folder containing
  - jxta.log
  - network.log
  - dataaccess.log
- .jxta folder for JXTA internal storage of information

Chapter 5

Installation and Configuration

tsc++ is a standalone middleware solution following the Triple Space Computing paradigm. The upcoming part contains a detailed description on how to install, configure and use the sample application bundled with the source code.

5.1 Contents of the Distribution Package

The distribution of *tsc++* contains:

bin folder containing the compiled Start*.class files

kernel1 folder containing property files, test data and startKernel2 scripts for starting the second kernel

kernel2 folder containing property files, test data and startKernel3 scripts for starting the third kernel

kernel3 folder containing property files

lib folder containing the jar file of the current *tsc++* release.

NOTE: external libraries must be placed in this folder

src folder containing the source files

ChangeLog.txt listing the changes for the actual release

LICENSE.txt digital copy of the underlying LGPL license

README.txt containing general information about this project

Start.bat script for Windows to start the (Read, Write, Query) getting started application

StartAdvertiseSubscribe.bat script for Windows to start the (Advertise, Subscribe) getting started application

Start.sh script for Linux to start the (Read, Write, Query) getting started application

StartAdvertiseSubscribe.sh script for Linux to start the (Advertise, Subscribe) getting started application

5.2 Getting started Application

The distribution contains two sample applications (consisting of **Start.java** in the src folder and **Start*.class** files in the bin folder).
To start the demo the under Windows **Start.bat** or **StartAdvertiseSubscribe.bat** should be called within a (dos) command shell.
To start the demo the under Linux **Start.sh** or **StartAdvertiseSubscribe.sh** should be called within a terminal.

NOTE: the application *xterm* must be installed in order to run the getting started correctly on Linux.

The samples should provide an overview over the following topics:

- starting a kernel instance
- creating spaces
- workflow visualisation
- operations on spaces
- advertise and subscribe

Every instance of TSkernel must be executed in a different Virtual Machine because JXTA does not allow multiple instances in the same VM. So every kernel needs its own folder and properties files. The demo uses the *tsc++* jar file in the lib folder so changes on the kernel source code will have no effect. Before starting, the external jar files listed in **README.txt** must be provided in the lib folder. Be sure to set the script's *CLASSPATH* variable correctly.

5.2.1 Read, Write, Query getting started Application

Consider a test scenario with three kernels **K1, K2, K3** co-authoring one space (space URI, ts://www.example.org/spaceGettingStarted/). Each kernel is performing different operations. Kernels have to wait (with a maximum timeout) for the output of others because triples need to be written before they can be read. A kernel that wants to read triples waits until those are found or stops doing so if a certain timeout is reached. The example looks the following (see: Figure 5.1):

- **K1**: wants to read triples specified by a certain template **t1** and writes triples **triples1**
  - K1 **read(space, t1)**
  - K2 is started by **startKernel2** script
  - wait for K2 to write its triples
- **K2**: writes triples **triples2** and wants to read triples specified by a certain template **t2**
  - K2 **write(space, triples2)**
  - K1 **read(space, t1)** and receives a set of triples
  - K1 **write(space, triples1)**
  - K2 **read(space, triples1)** and receives a set of triples
  - K3 is started by **startKernel3** script

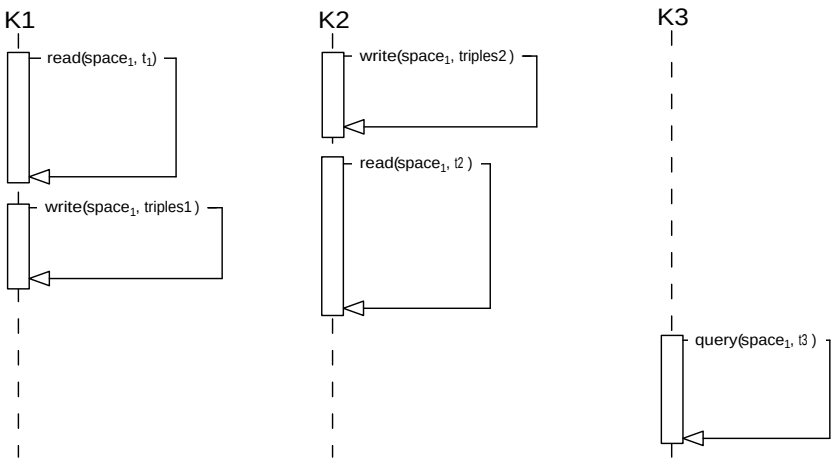


Figure 5.1: Execution Path read write query

- **K3**: wants to query triples specified by a template **t3**

  - K3 **query(space, t3)** and receives 2 triples from K2 and K1 since the template matches triples in both kernels

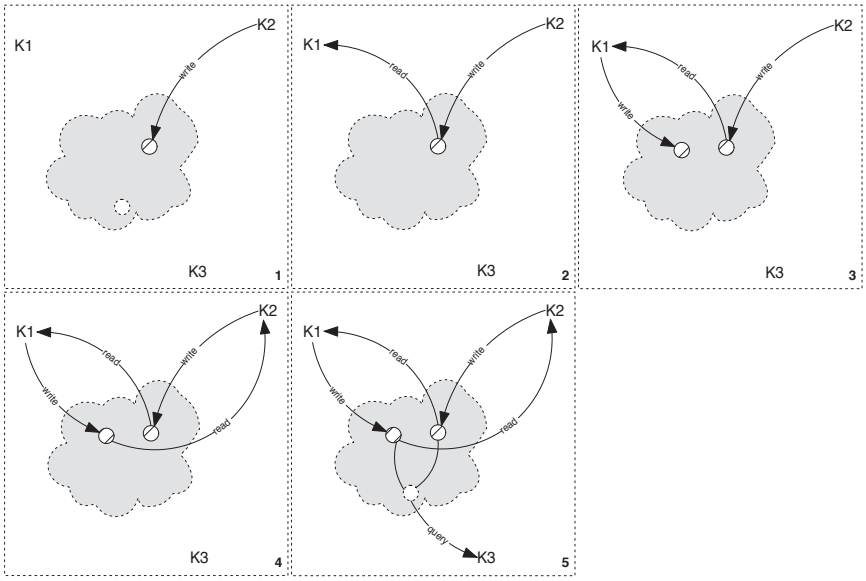


Figure 5.2: 3 Kernels in one space performing read, write and query

5.2.2 Advertise , Subscribe Application

Consider a test scenario with three kernels **K1, K2, K3** co-authoring one space (space URI, ts://www.example.org/spaceGettingStarted/). Each kernel is performing different advertisements and subscriptions. The example looks the following (see: Figure 5.3):

- **K1**: subscribes to a certain template **t2**, creates NotificationListener **l1** to listen for advertisements and advertises template **t1**

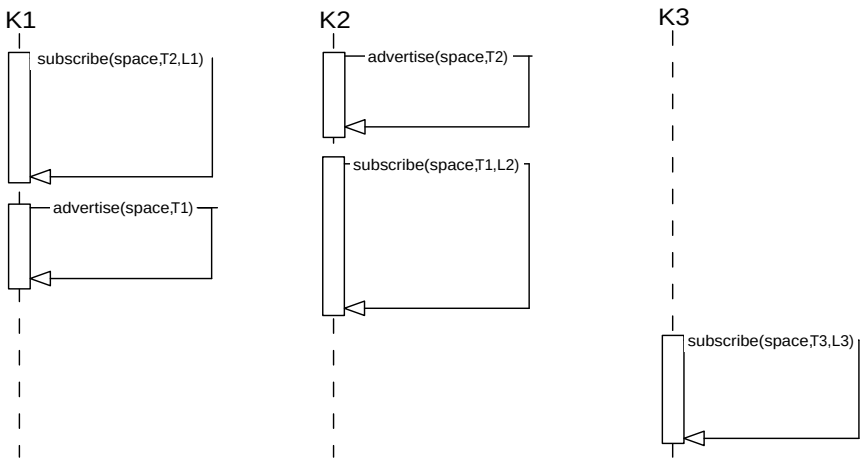


Figure 5.3: Execution Path advertise subscribe

- K2 is started by **startKernel2** script

- K1 **subscribe(space, t2, l1)**

- **K2**: advertises template **t2**, subscribes to template **t1** and creates NotificationListener **l2** to listen for advertisements

  - K2 **advertise(space, t2)**

  - K1 **l1** receives a notification

  - K2 **subscribe(space, t1, l2)**

  - K2 **l2** receives a notification

  - K3 is started by **startKernel3** script

- **K3**: subscribes to template **t3** which matches both templates and creates NotificationListener **l3** to listen for advertisements

  - K3 **subscribe(space, t3, l3)** receives notification for both advertisements, since **t3** matches **t1** and **t2** as well

5.3 Configuration

Every new kernel instance can be configured separately by changing the properties contained in separate files inside the kernel's properties folder. The properties are divided into the following files.
NOTE: to use Owlim the Java VM arguments *'-Xmx256m -DentityExpansionLimit=100000000'* are mandatory

5.3.1 General Properties

Property	Values	Description
STORE	sesame owlim	which store to use for persistency
ACCESSLOG	true false	turn on access log
COMPRESS_MESSAGES	true false	compress messages

5.3.2 JXTA Properties

TIMEOUT_READ_URI	Long	time in which a response is expected for a certain read
MAX_TIMEOUT_READ_URI	Long	wait MAX_TIMEOUT/MAX_ATTEMPTS after unsuccessful read before next attempt is started
MAX_ATTEMPTS_READ_URI	Long	max attempts a read operation should be executed until read is successful
TIMEOUT_READ_TEMPLATE	Long	time in which a response is expected for a certain read
MAX_TIMEOUT_READ_TE...	Long	wait MAX_TIMEOUT/MAX_ATTEMPTS after unsuccessful read before next attempt is started
MAX_ATTEMPTS_READ_TE...	Long	max attempts a read operation should be executed until read is successful
TIMEOUT_TAKE_URI	Long	time in which a response is expected for a certain take
MAX_TIMEOUT_TAKE_URI	Long	wait MAX_TIMEOUT/MAX_ATTEMPTS after unsuccessful take before next attempt is started
MAX_ATTEMPTS_TAKE_URI	Long	max attempts a take operation should be executed until take is successful
TIMEOUT_TAKE_TEMPLATE	Long	time in which a response is expected for a certain take
MAX_TIMEOUT_TAKE_TE...	Long	wait MAX_TIMEOUT/MAX_ATTEMPTS after unsuccessful take before next attempt is started
MAX_ATTEMPTS_TAKE_TE...	Long	max attempts a take operation should be executed until take is successful
TIMEOUT_QUERY_TEMPLATE	Long	time in which a response is expected for certain query
MAX_TIMEOUT_QUERY_TE...	Long	wait MAX_TIMEOUT/MAX_ATTEMPTS after unsuccessful query before next attempt is started
MAX_ATTEMPTS_QUERY_TE...	Long	max attempts a query operation should be executed until query is successful
TIMEOUT_DISCOVER	Long	timeout to discover advertisments and peers
MAX_TIMEOUT_DISCOVER	Long	max timeout for discovery of advertisements and peers
MAX_ATTEMPS_DISCOVER	Long	max attemts to discover advertisements and peers
THRESHOLD_DISCOVER	Long	the upper limit of responses to a discovery message
DEFAULT_EXPIRATION	Long	default expiration time for advertisements, this is the amount of time advertisements will live in caches. After this time, the

		advertisement should be refreshed from the source.
DEFAULT_LIFETIME	Long	default lifetime time for advertisements, this is the maxiumum amount of time the advertisement will remain valid. If the advertisement remains valid after this time, then the creator will need to republish the advertisement.
SEND_TRIPLES_ON_LEAVE	true false	send triples to other kernel in space when leaving space
RELAY_SEEDING_URI	String	connection URI to the Relay server
RDV_SEEDING_URI	String	connection URI to the Rendezvous server
WAIT_FOR_RDV_CONNECTION	true false	waiting for (peer group) RDV connection (Rendezvous peer) is essential except for the kernel (normally the first one stared up) that should become the Rendezvous peer
TYPE_OF_COMMUNICATION	100 101 102	flood based communication random walk based communication biased random walk based communication
COMBINE_QUERY_TEM...	true false	combines the query results if multiple kernels answer to a request
MAX_HOP_COUNT	Long	max hops used for random walk communication
CLEAN_DIR	true false	clean the JXTA directories at startup

5.3.3 Sesame Properties

Used if 'store = sesame' is set in the *general.properties* file.

CLEAN_DIR	true false	clean the Sesame directories at startup

5.3.4 Owlim Properties

Used if 'store = owlim' is set in the *general.properties* file.

NUM_WORKER_THREADS	Long	number of worker threads
CLEAN_DIR	true false	clean the Owlim directories at startup

5.4 Using the RESTKernel

Instead of using the full distribution a light version of *tsc++* is available. The restkernel[current version].jar comes bundled with all interfaces mandatory for running *tsc++*, log4j and the current version of Sesame. The size of the jar is minimal and every RESTKernel object can be replaced by a TSKernel object easily. RESTKernel's communicate over HTTP (following the REST principle) with a remote TSkernel on a STI server. Information on the communication

can be found in this chapter. **NOTE:** RESTKernel comes preconfigured and therefore no properties file are contained in the package.

RESTKernels communicate with the TSKernel on the server via HTTP Get and Post methods.

Get The information is encoded in the URL query string and accessible directly through a webbrowser. e.g. the operation 'read ts://rest.examplespace/space0 using template ?s ?p ?o with a timeout of 20000ms' is perfomed by

```
http://../readTemplate
?spaceURI=ts%3A%2F%2Frest.examplespace%2Fspace0%2F
&template=%3Fs+%3Fp+%3Fo
&timeout=20000
```

A read, take, query operation without a specified timeout uses the standard timeout of 15000ms.

Post The information is sent as HTML form. e.g. the operation 'write
<http://www.example.org/TSC>
<http://www.example.org/hasInConsortium>
<http://www.tuwien.ac.at>
to space ts://rest.examplespace.space0' (triples will be sent in N3 format) is performed by

```
http://../write

in body of message:

spaceURI: ts://rest.examplespace.space0
triples: <http://www.example.org/TSC>
<http://www.example.org/hasInConsortium>
<http://www.tuwien.ac.at>
```

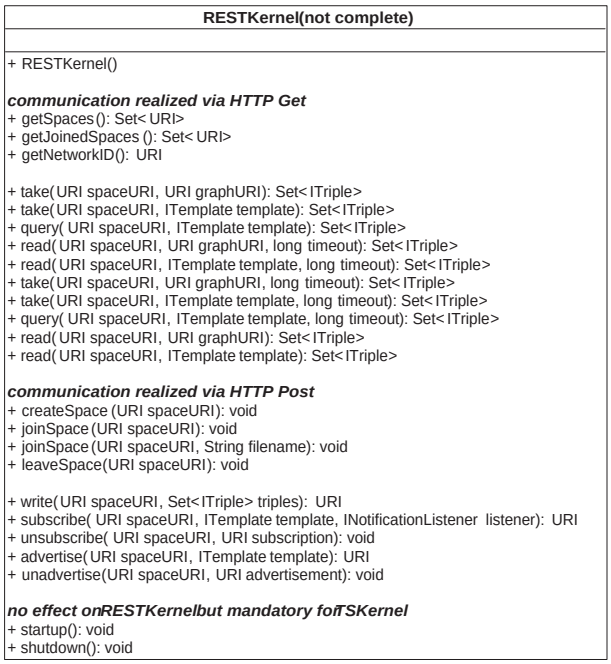


Figure 5.4: RESTKernel overview

According to the found reult the TSkernel answers result as follows.

if result is a Set of Triples the TSKernel will return the set as RDF file N3 format. e.g. the operation 'query using template ?s ?p ?o' results in

```
<http://www.example.org/TSC>
<http://www.example.org/hasDeliverable>
<http://www.example.org/r1> .
<http://www.example.org/TSC>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://usefulinc.com/doap/Project> .
```

if result is a Set of URIs the TSKernel will return the result as plain text containing the URIs. e.g. the operation 'get joines spaces' returns the list

```
ts://rest.examplespace/space0/
ts://rest.examplespace/space1/
```

else The answer will be packed into an XML message. e.g. the operation 'get network id' returns

```
<TSKernelRestMessage>
<result class="java.net.URI">
urn:jxta:uuid-59616261646162614E50472050325033405BD8DA1969495D958EDF6CA1D1B67D03
</result>
</TSKernelRestMessage>
```

Occurring exceptions will also be reported using this XML format.

Chapter 6

Deployment of an AMI for EC2

Since *tsc++* can be very memory (huge datasets in memory) and bandwidth (sending big sets of triples over P2P) hungry the execution of multiple kernel instances on a single machine is limited. To test or run systems based upon *tsc++* the usage of a server is recommended. This Chapter explains how a kernel can be bundled as an AMI (Amazon Machine Image) [1] and uploaded to Amazon EC2 (Elastic Compute Cloud) [2], which offers the possibility to virtually run multiple kernels on multiple machines.
NOTE: Usage of the service is connected with charges.

6.1 Cloud Computing

Cloud computing is a term strongly interwoven with the vision of grid computing. The users of a software do not operate their applications and necessary hardware by themselves. A provider operates both for them. The application and data is not stored on a local machine anymore, instead it is distributed in a 'cloud' of multiple remote systems.

6.1.1 EC2

Amazon Elastic Compute Cloud is part of Amazon's web services and 'provides resizable compute capacity in the cloud'. It is a large pool of virtual machines that can be rented by the hour. To use the service an Amazon account, a Linux based operating system, a SSH client and Java Runtime Environment (at least version 1.5) are mandatory. Following steps must be performed to get an application up and running:

- creation of an AMI (see: 6.1.2) containing the operating system, applications, libraries, data and their associated configuration settings
- upload of the AMI into Amazon S3 (Simple Storage Service) [3] (see: 6.1.3)
- configuration of security and network access via EC2 web service
- start, termination, and monitoring of as many instances of the AMI as needed, using the web service APIs
- running multiple instances in multiple locations

NOTE: The consumed resources, like instance-hours or data transfer are to be paid.

For communication with EC2 web service the command line tools (provided on http://aws.amazon.com) serve as client interface. For detailed information Judith Myerson's guide 'Creating Applications with Amazon EC2 and S3' [5] is recommended.

6.1.2 AMI

An Amazon Machine Image is a bootable Linux image, with software (applications using *tsc++*) pre-installed. The system is based on Xen[1] virtualisation [19]. To create a snapshot of your operating system the Amazon EC2 AMI (command line) Tools (available at http://aws.amazon.com) have to be downloaded and installed. The tools contain the ec2-bundle-vol utility, which encrypts (X.509 certificate) and signs the image to ensure it cannot be tampered with. This ensures that the image can only be decrypted by EC2. Afterwards the the AMI must be uploaded to S3 store, using ec2-upload-bundle tool. The uploaded image must be registered with EC2 before running any instances. The creation of the image is free of charge.

6.1.3 S3

The Simple Storage Service is 'storage for the Internet'. It provides a web service (REST and SOAP) interface and is capable of storing and retrieving a huge amount of data, from anywhere on the web. An unlimited number of objects (each up to 5 gigabyte) of data can be stored. Objects can be private or visible to the public. Provided authentication mechanisms ensure the data is kept secure from unauthorized access. A Perl script to access S3 from the command line is available.

[1] Xen is a free software virtual machine monitor for multiple instruction set architectures. It allows several different operating systems to be executed on the same hardware at the same time

Bibliography

[1] *Amazon Elastic Compute Cloud*. http://aws.amazon.com/ec2/.

[2] *Amazon Machine Images*. http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=171.

[3] *Amazon Simple Storage Service*. http://aws.amazon.com/s3/.

[4] *CORSO*. Coordinated Shared Data Objects,http://www.complang.tuwien.ac.at/eva/.

[5] *Creating Applications with Amazon EC2 and S3*. http://www.onlamp.com/pub/a/onlamp/2008/05/13/creating-applications-with-amazon-ec2-and-s3.html.

[6] *IETF (Internet Engineering Task Force) UUID Proposed Standard RFC 4122*. http://tools.ietf.org/html/rfc4122.

[7] *Mobile Web Initiative*. W3C, http://www.w3.org/Mobile/.

[8] *OWLIM*. http://www.ontotext.com/owlim/.

[9] *P-Grid*. http://www.p-grid.com/.

[10] *Representational State Transfer*. http://en.wikipedia.org/wiki/Representational_State_Transfer.

[11] *SESAME*. http://www.openrdf.org/about.jsp.

[12] *Triple Space Communication*. http://www.tripcom.org/.

[13] *Triple Space Computing*. DERI, http://tsc.deri.at/.

[14] *Triple Space Computing, TSC D1.2*. DERI, http://tsc.deri.at/deliverables/D12v11.html.

[15] *tsc++, Triple Space Computing*. STI Innsbruck, http://tsc.sti2.at/.

[16] *W3C announces best practices for mobile web*. http://www.mobilemag.com/content/100/344/C6258/.

[17] *World Wide Web Consortium*. W3C, http://www.w3.org/.

[18] *World Wide Web Size*. http://www.worldwidewebsize.com/.

[19] *Xen virtualisation*. http://www.xen.org.

[20] *The Zero-Delay Data Warehouse: Mobilizing Heterogenous Databases*. TECCO Coordination Systems.

[21] First and second generation of peer-to-peer systems. In *Peer-to-Peer Systems and Applications*, pages 35–56. Springer-Verlag, 2005.

[22] Past and future. In *Peer-to-Peer Systems and Applications*, pages 17–23. Springer-Verlag, 2005.

[23] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, 2000. Chair-Richard N. Taylor.

[24] Reto Krummenacher, Martin Hepp, Axel Polleres, Christoph Bussler, and Dieter Fensel. Www or what is wrong with web services. In *ECOWS '05: Proceedings of the Third European Conference on Web Services*, page 235, Washington, DC, USA, 2005. IEEE Computer Society.