



PROTOTYPE DOCUMENTATION

TASK 3.3.2 GRID MONITORING SANTA-G

WP3

Document Filename:	CG3.3.2-TCD-D3.3-v1.1-SANTAG.doc
Work package:	WP3
Partner(s):	TCD
Lead Partner:	TCD
Config ID:	CG3.3.2-TCD-D3.3-v1.1-SantaG
Document classification:	PUBLIC

Abstract: According to the CrossGrid Technical Annex [CGTA] Trinity College Dublin (TCD) will 'develop specific relational producer/consumers of non-invasive performance monitoring information, as well as associated SQL-query-based performance analysis tool support'. This document describes the first prototype of these tools and services, referred to in the rest of this document as SANTA-G non-invasive monitoring services. It describes the implementation structure as well as functionality of the prototype. The document also includes a description of the installation procedure and a user guide.

Delivery Slip

	Name	Partner	Date	Signature
From				
Verified by				
Approved by				

Document Log

Version	Date	Summary of changes	Author
1.0	17/01/2003	Draft version	Brian Coghlan, Stuart Kenny
1.1	21/02/2003	Altered with respect to internal reviewers comments	Brian Coghlan, Stuart Kenny

CONTENTS

EXECUTIVE SUMMARY	4
1. INTRODUCTION	5
1.1. PURPOSE	5
1.2. DEFINITIONS, ABBREVIATIONS, ACRONYMS	5
2. REFERENCES	6
2.1. SOURCE CODE	6
2.2. CONTACT INFORMATION	6
2.3. REFERENCED DOCUMENTS	6
3. IMPLEMENTATION STRUCTURE	7
4. PROTOTYPE FUNCTIONALITY	12
5. USER MANUAL	17
5.1. DEPENDENCIES	17
5.2. INSTALLATION	17
5.3. RUNNING	17
5.4. INTERFACE DESCRIPTION	20
6. INTERNAL TESTS	21
7. ISSUES	22

EXECUTIVE SUMMARY

This document describes the first prototype of the SANTA-G services.

Sections 1 and 2 provide an introduction as well as references. Section 3 describes the implementation structure of the first prototype. Section 4 describes the functionality. The User Manual for the prototype is included in Section 5, which describes how to install, and run the code.

1. INTRODUCTION

1.1. PURPOSE

SANTA-G services are a specialized non-invasive complement to other more intrusive monitoring services. The application of these services will be in validation and calibration of both intrusive monitoring systems and systemic models, and also for performance analysis. The objectives are to allow information captured by external monitoring instruments to be introduced into the Grid information system, and to support analysis of performance using this information. The first prototype illustrates these concepts with a NetTracer, a demonstrator that allows network trace information obtained by the program TCPDump to be accessed through the Grid information system. In reality the underlying concepts have wider applicability; they allow information from a great variety of instruments to be accessed through the Grid information system.

1.2. DEFINITIONS, ABBREVIATIONS, ACRONYMS

CrossGrid	The EU CrossGrid Project IST-2001-32243
DataGrid	The EU DataGrid Project IST-2000-25182
GUI	Graphical User Interface
R-GMA	DataGrid relational Grid monitoring architecture
SANTA	System Area Network Trace Analysis
SANTA-G	Grid-enabled System Area Network Trace Analysis
SRS	Software Requirements Specification

2. REFERENCES

2.1. SOURCE CODE

The source code can be found in the FZK CVS, please see:

<http://gridportal.fzk.de/?group=cg-wp3-3> for details on how to obtain it.

The source code for the CanonicalProducer and CanonicalProducer Servlet form part of the DataGrid R-GMA and can be found in the DataGrid CVS repository. For instructions on how to access it, please see: <http://datagrid.in2p3.fr/HOWTO/CCIN2P3CVSaccessHOWTO.html>

2.2. CONTACT INFORMATION

Stuart Kenny stuart.kenny@cs.tcd.ie

2.3. REFERENCED DOCUMENTS

CGTA	CrossGrid Project Technical Annex <i>CROSSGRIDANNEX1_V3.0</i>
Task3.3 SRS	Task3.3 Grid Monitoring Software Requirements Specification <i>CG-3.3-SRS-0013</i>
Task3.3.2 SDD	Task3.3 Grid Monitoring Software Design Document <i>CG3.3.2-D3.2-v1.2-TCD020-SantaGDesign</i>

3. IMPLEMENTATION STRUCTURE

In the design document [Task 3.3 SDD] it was stated that the SANTA-G demonstrator, the NetTracer, was composed of two main modules, the publishing module, and the viewer module. Both of these modules are present in the prototype, although the full functionality of some components of each of these has yet to be implemented. The following diagram (Figure 3.1) shows the structure of the NetTracer as per the design document.

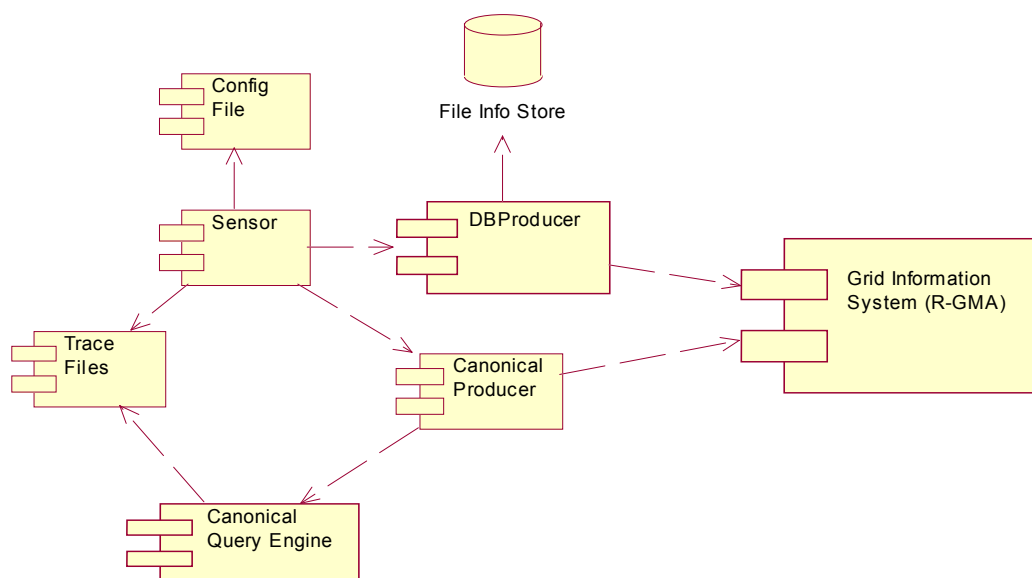


Figure 3.1 NetTracer composition as per design document

The publishing module, for the prototype, has four main components, the CanonicalProducer Servlet, the CanonicalProducer API, the Query Engine, and the Sensor (for the first prototype the Sensor has only limited functionality). The CanonicalProducer Servlet and API have been implemented and are now a part of the DataGrid R-GMA. They can be seen in the DataGrid CVS repository (see references). They allow a user to publish arbitrary data from any source, e.g. log files, into the Grid Information System. The SANTA-G demonstrator of this concept is a NetTracer, which provides access to network trace information obtained by the program TCPDump, through the information system.

An initial, very basic, Sensor implementation has been provided for the first prototype, with only a very limited functionality. The following class diagram shows the classes that make up the Sensor (Figure 3.2).

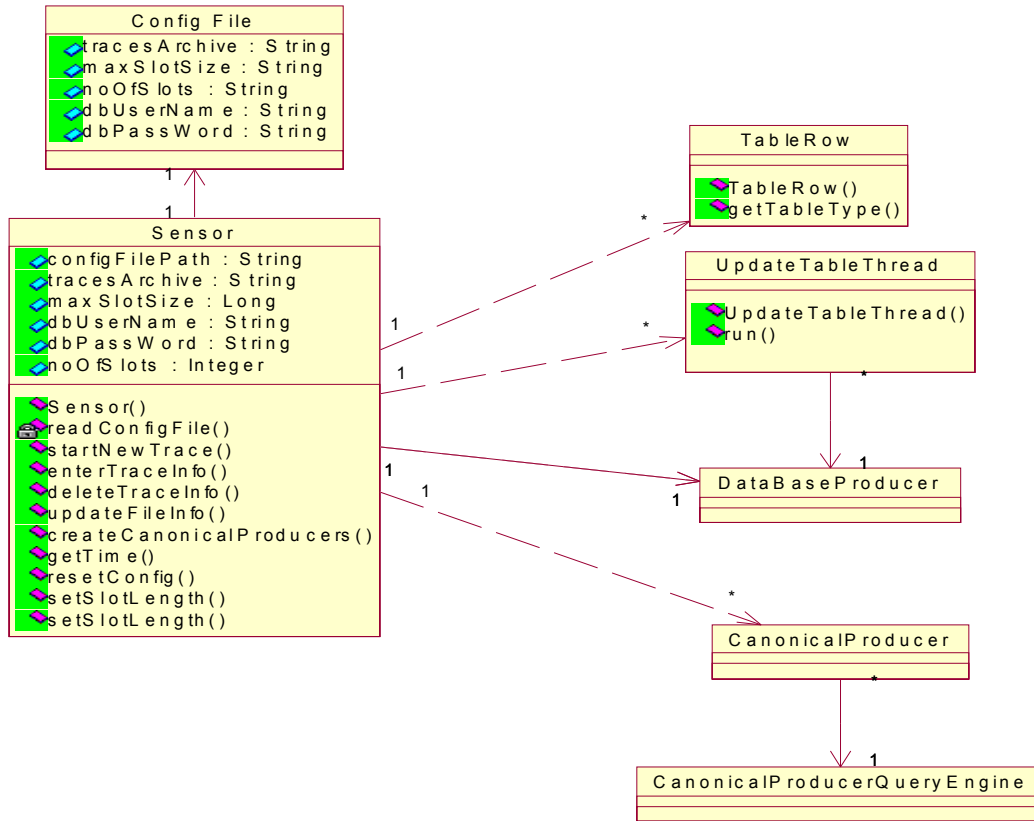


Figure 3.2 Sensor Class Diagram

The NetTracer Query Engine has been implemented for the prototype. The structure of the Query Engine is shown in the component diagram below (Figure 3.3).

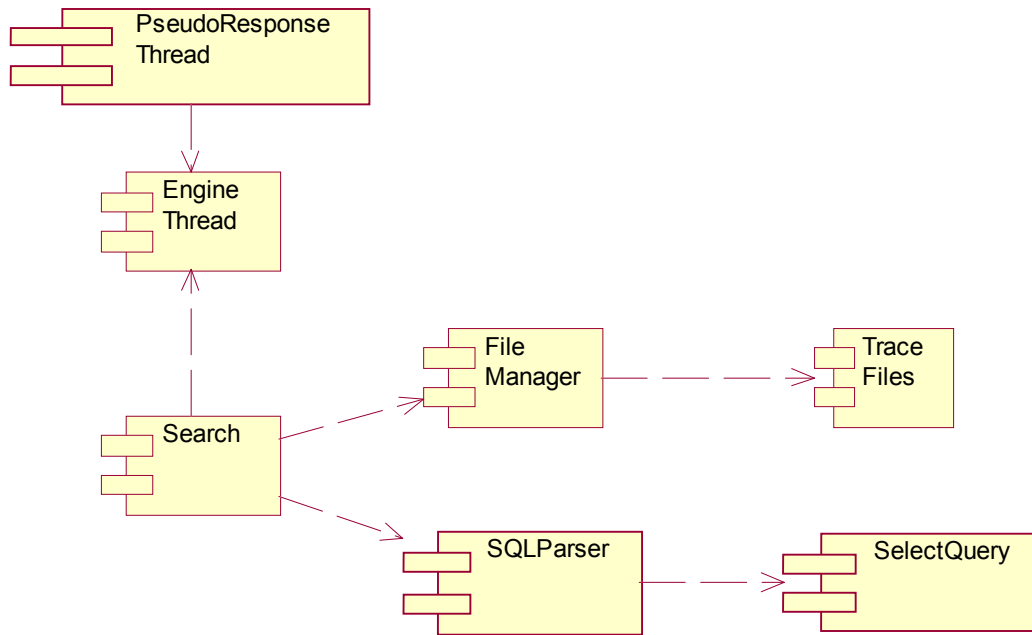


Figure 3.3 Components of the Query Engine

The following diagram (Figure 3.4) shows the classes that make up the query engine:

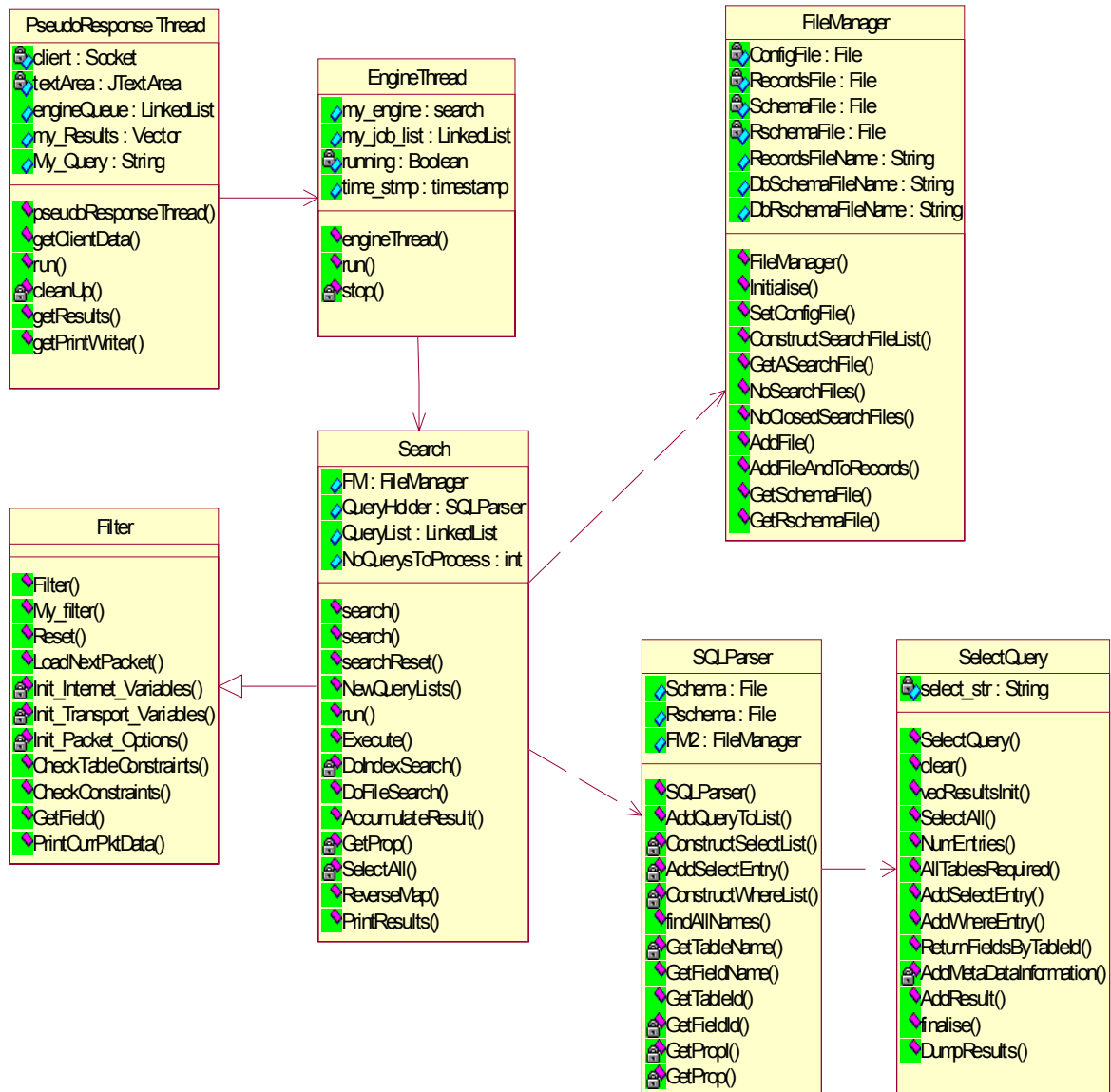


Figure 3.4 Query Engine class diagram

The viewer module allows a user to view the TCPDump data through the Grid information system. Interaction with the viewer module is through the Viewer GUI. For the first prototype of the NetTracer demonstrator, the Viewer contains two panels. The first panel provides a graphical view that allows the user to navigate a log file and displays the packets from the file. The second panel allows for direct SQL querying of a log file. The Viewer module is composed of two main components, the Viewer GUI and the Consumer API.

In the spirit of R-GMA, a SQL query statement is used to define what data is to be acquired. The Viewer receives and displays resultsets.

Some of the functionality of the components has yet to be implemented. The Sensor application is one of these. The purposes of the Sensor is to allow a user to start and stop TCPDump, and also maintain tables of information relating to the resulting log files by using a R-GMA DBProducer (a standard R-GMA component). For the first prototype the Sensor provides only limited tables of information using a DBProducer, and it does not allow a user to control TCPdump. The Sensor therefore only provides information on static pre-existing log files. Because of this the prototype can only use logfiles generated off-line by TCPdump. Also the prototype can only handle a subset of the potential schema for the trace files, and can only handle a more limited subset of SQL than will eventually be supported.

4. PROTOTYPE FUNCTIONALITY

In the design document the following functions were listed for SANTA-G:

1. Allow a user to initiate non-invasive tracing of grid resources. Collect the trace data, and provide access to the data through the Grid information system.
2. Allow a user to select the required subset of trace data, by way of the Grid information system, persistently store this subset of data in a relational trace database, and provide access to this stored data for further analyses.

Most of the aspects of this overall functionality have been implemented, at least in a limited way. The full functionality of the Sensor application (see the design document) has yet to be implemented, so for the first prototype only static pre-generated logfiles may be used. The user may issue queries to the Grid information system to view relational subsets of the data in the logfiles. The following describes the functionality of the prototype.

In order to obtain data from the Grid Information System a user may use the SANTA-G Viewer. In the spirit of R-GMA, a SQL query statement is entered into the GUI in order to define the data that is to be acquired. The Viewer, by using its own Consumer interface, will contact a Consumer Servlet, which in turn contacts a R-GMA registry in order to locate the required producers of the information. The information is returned through the same mechanisms to the Viewer in the form of a ResultSet. The individual fields of the result set are then extracted and displayed by the Viewer as a table. The user may also use the graphical display panel, which provides a number of controls for navigating a trace. Selected packets can then be displayed graphically in the Viewer GUI. The sequence diagram below, Figure 4.1, shows this process.

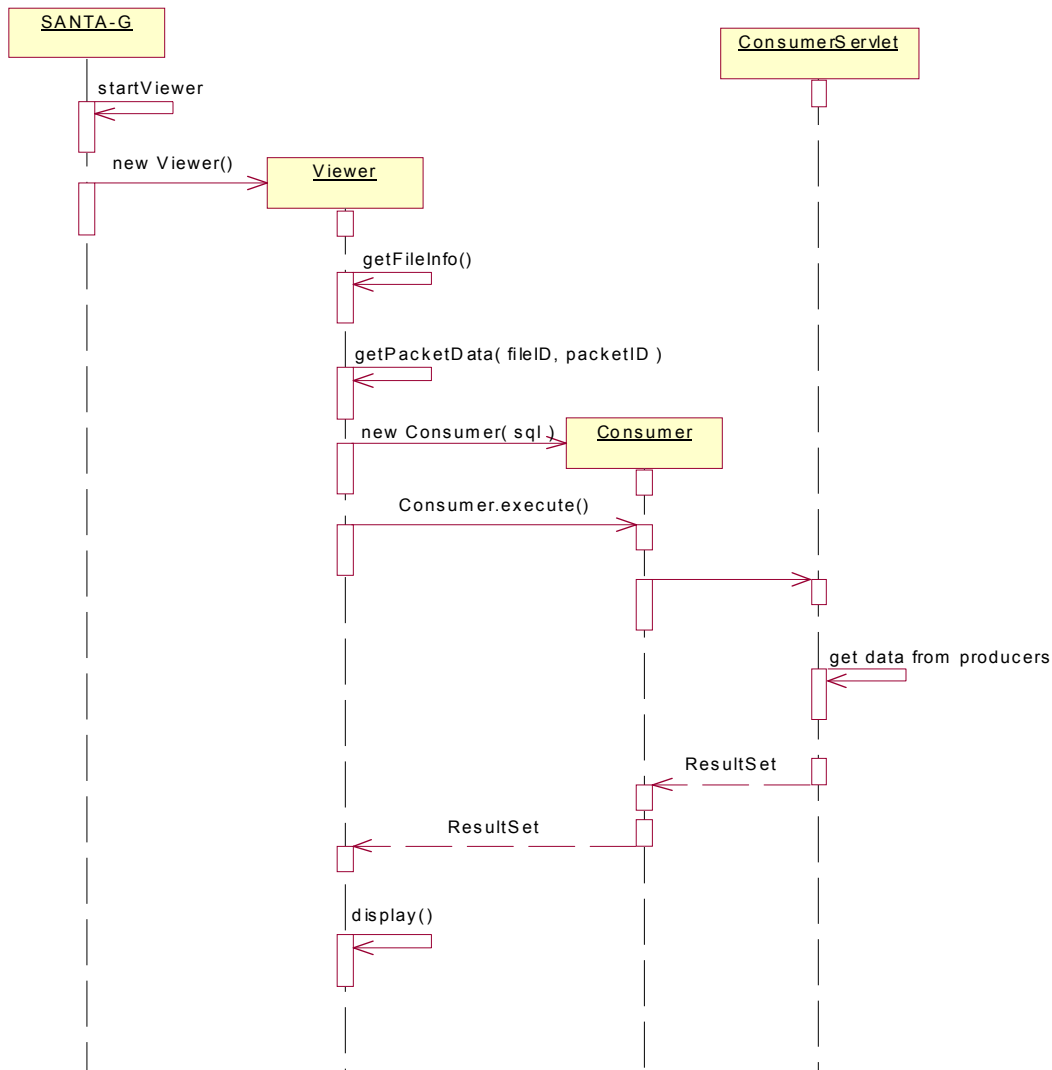


Figure 4.1 Viewing data, Interaction between the Viewer and ConsumerServlet

Some aspects of this sequence are yet to be fully implemented. For example, the Viewer cannot yet obtain available file information dynamically. This must be obtained in advance by issuing a query to the Sensor file information tables.

The Sensor maintains two tables of information relating to the trace available. The first is the Trace Information table. It stores a trace ID, and the path to the directory used to store the trace log files. The second table, the File Information table contains information relating to the files used to store the raw trace data within the trace directory. This table stores again, the path of the directory used to store the trace circular queue, a file ID, which identifies an individual file within the trace, and the filename. These two tables are stored, and subsequently published, by using a DBProducer.

Figure 4.2 shows the interaction between the CanonicalProducer Servlet, the CanonicalProducer, the QueryEngine and a TCPDump logfile in response to a query received from a consumer (e.g the SANTA-G Viewer). An application that requires data does so by using a Consumer interface. As described above the Consumer contacts a ConsumerServlet, sending it a SQL SELECT statement that defines the data required. The ConsumerServlet uses a R-GMA registry to find the Producers that can satisfy the query. Once found the query is forwarded to these Producers. The CanonicalProducer servlet will receive the query and in turn forward it to the CanonicalProducer that holds the information needed. The CanonicalProducer now uses the QueryEngine to collect the data from the raw TCPDump log file by performing seek operations. Once the required data is found it is returned to the QueryEngine.

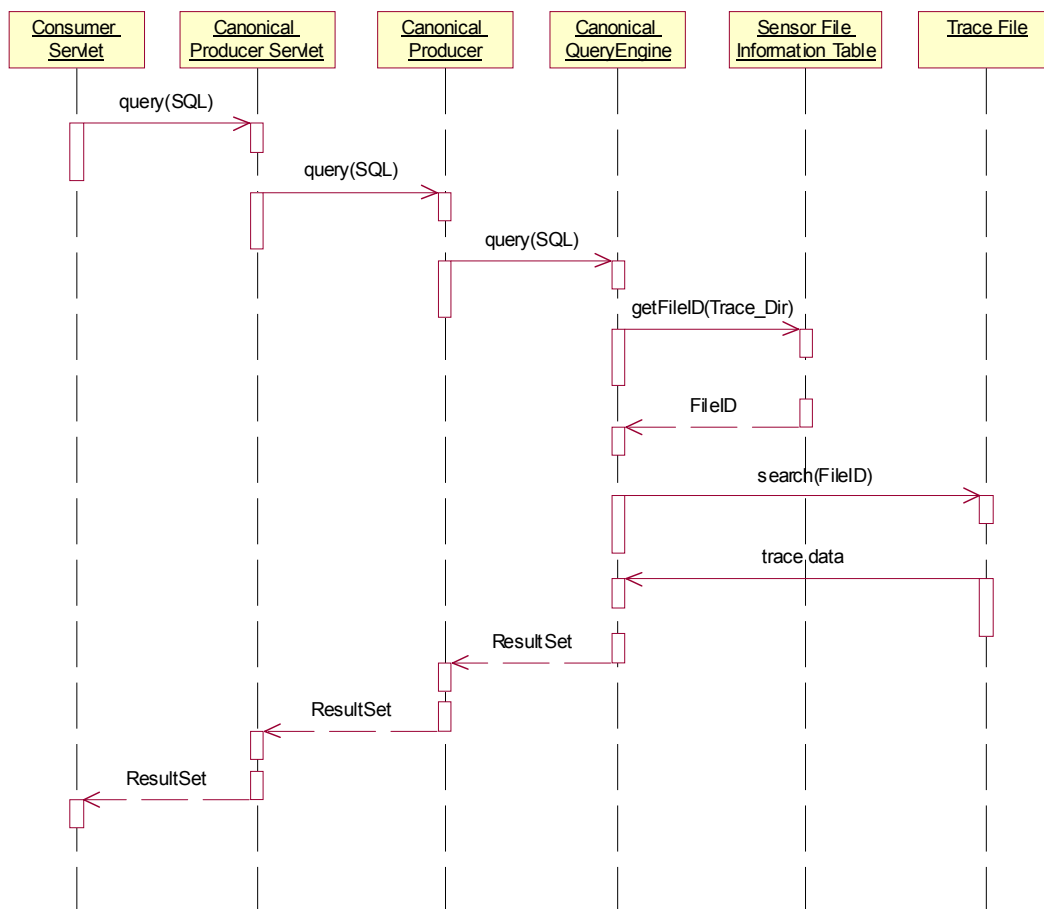


Figure 4.2 Interaction between ConsumerServlet and Trace File

The Query Engine constructs a ResultSet from the data and returns it to the CanonicalProducer. The ResultSet is then sent by the same mechanism back to the Consumer. The Consumer can be any type of application, a custom coded Consumer, an Archiver or a compound Consumer/Producer. Refer to the *DataGrid Information and Monitoring (WP3) Architecture Report* for a description of these

components. However, for the NetTracer demonstrator a standard R-GMA Consumer is used to query the information in the CanonicalProducer.

Within the Query Engine a list of pending jobs is stored. These jobs contain queries that must be carried out. CanonicalProducers connect to a server by way of a socket. The Server listens to the socket and upon receiving a connection it creates and starts a response thread. A PseudoResponseThread reads the query from the socket and places itself on the list of jobs, which is maintained and serviced by an EngineThread, started by the Server upon initialisation. An EngineThread uses an instance of the Search class to carry out the query on a log file. The Search class uses the File Manager to obtain pointers to these files. The SQLParser takes the query and creates an instance of a SelectQuery. This breaks the query down into three separate linked lists, each representing a section of the SQL SELECT statement, i.e. a *select list*, a *from list* and a *where list*. The Search class can then search the logfile – many classical optimisations can be applied to accelerate this. At present, no optimisation is done. If a packet satisfies the query given, then it is added to a result set object created by the Search class and returned to the waiting PseudoResponseThread, and once this is done the ResultSet can then be returned to the client. This process is shown in Figure 4.3.

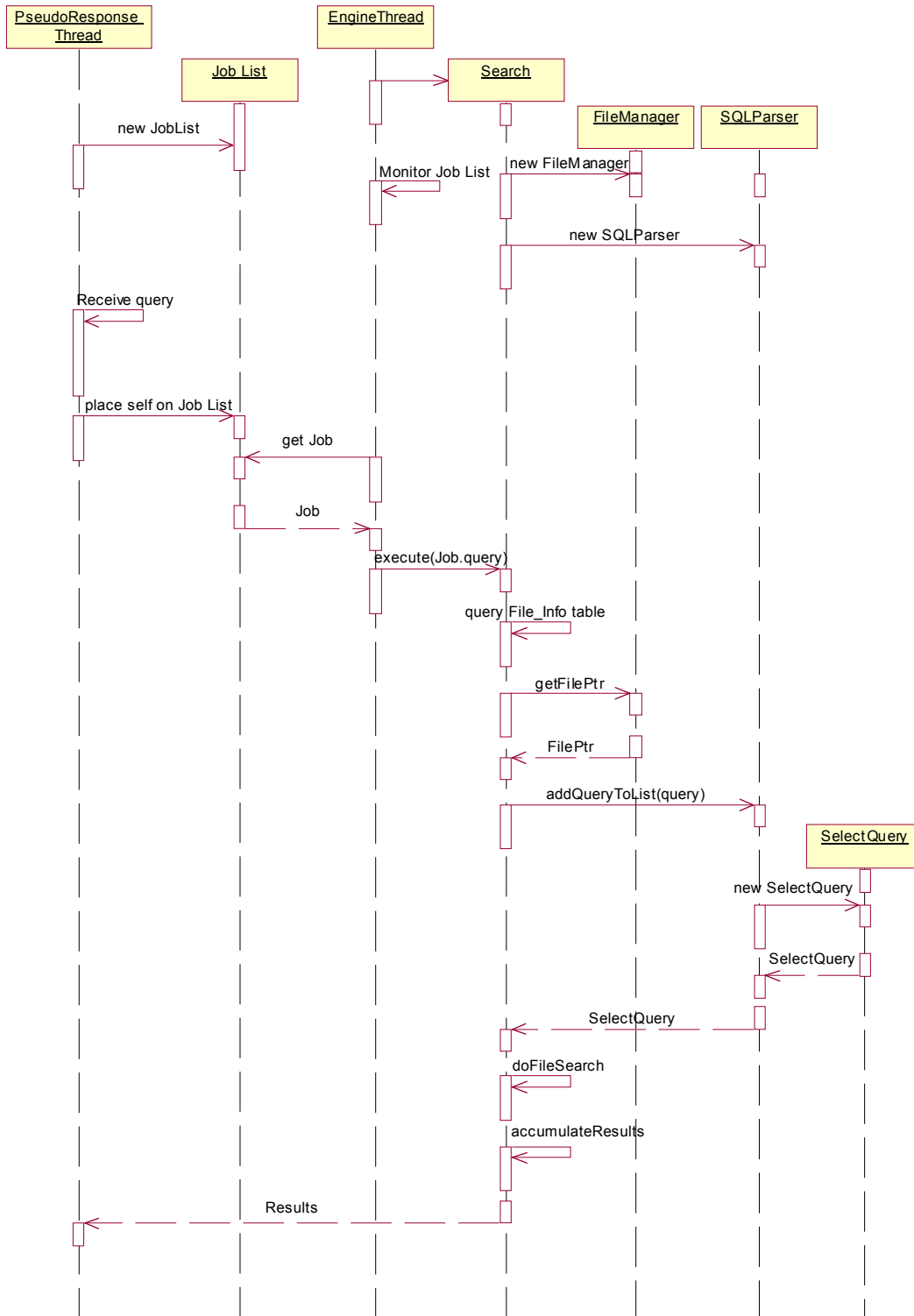


Figure 4.3 Query Engine sequence diagram

5. USER MANUAL

5.1. DEPENDENCIES

- The R-GMA information and monitoring system must be installed and running

To build SANTA-G

- Ant (a java based build tool, available from <http://ant.apache.org/>)

For the creation of logfiles:

- TCPdump 3.6.x
- libpcap 0.6.2

5.2. INSTALLATION

The source code is available in the FZK CVS repository. See <http://gridportal.fzk.de/cvs/?group=cg-wp3-3> for details on how to obtain it. Checkout the code into the **/opt/santag** directory. This is the install location for the first prototype. The scripts must be run as root, or with root privileges, unless you have write access to the /opt directory. Then follow the following steps:

- Run the R-GMA setup script. This can be found in the gma/etc directory, for example:

```
> ./home/stkenny/gma/etc/rgma-setup.sh
```

- Enter the SANTA-G install directory.

```
> cd /opt/santag
```

- Run Ant

```
> ant
```

This will build all the necessary class files.

- Enter the bin directory and run the sensorDBSetup script.

```
> cd bin
```

```
> . sensorDBSetup.sh
```

This script will build the necessary database and database tables in MySQL for the SANTA-G Sensor.

- Now run the schemaSetup script.

```
> . schemaSetup.sh
```

This script will enter the necessary table descriptions in the R-GMA schema for the SANTA-G NetTracer.

5.3. RUNNING

A number of run scripts are provided in order to access the SANTA-G prototype. These can be found in the bin directory, i.e. /opt/santag/bin.

- Change directory to the SANTA-G install directory, > **cd /opt/santag**
- Enter the bin directory.

- There are three scripts that can now be run:
 - `startupSensor.sh` starts the Sensor.
 - `startupQueryEngine.sh`, starts the Query Engine.
 - `StartupViewer.sh`, starts the Viewer GUI

The following illustrates an example session:

1. First, run the R-GMA setup script, see Figure 5.1,
> **`./home/stkenny/gma/etc/rgma-setup.sh`**
2. Change directory to the SANTA-G install location
> **`cd /opt/santag/`**
3. Enter the bin directory
> **`cd bin`**
4. Run the Sensor startup script
> **`./startupSensor.sh`**
5. Run the Query Engine startup script
> **`./startupQueryEngine.sh`**
6. You will be asked for a port number on which the Query Engine will listen for communications. Once this is done the query engine will start.
7. Next run the Viewer script
> **`./startupViewer.sh`**
This will bring up the Viewer GUI (see Figure 5.1). A sample query is displayed in the query panel, this can be executed by simply pressing the execute button.
7. When execute is pressed the Viewer will construct a consumer to carry out the query.
8. The query should be received by the query engine and echoed to the terminal.
9. The query will then be carried out and the results returned to the viewer where they will be displayed in a table.

To obtain information on the available log files the tables maintained by the Sensor can be queried by using the query panel of the Viewer GUI. Enter the SQL query, `SELECT * FROM File_Info`. This will display the available file ID's.

By way of example, let us say that you wish to know what Ethernet packets are stored in a logfile. You need to enter a SQL query into the Viewer that `SELECTs Ethernet.packet_type FROM Ethernet WHERE Ethernet.fileid = <N>`, where `<N>.log` is the filename of a logfile in at default location known to the Query Engine – at present only this default location is able to be used. The Viewer displays the resultSet.

Let us say that you wish to examine one of the displayed Ethernet packets. Enter a SQL query into the Viewer that `SELECTs Ethernet.destination_address, Ethernet.source_address, Ethernet.packet_type FROM Ethernet WHERE Ethernet.fileid = <N> AND Ethernet.packetid=<J>`, where `<J>` is the sequential packet number in the logfile. Again the Viewer displays the resultSet, see Figure 5.1:

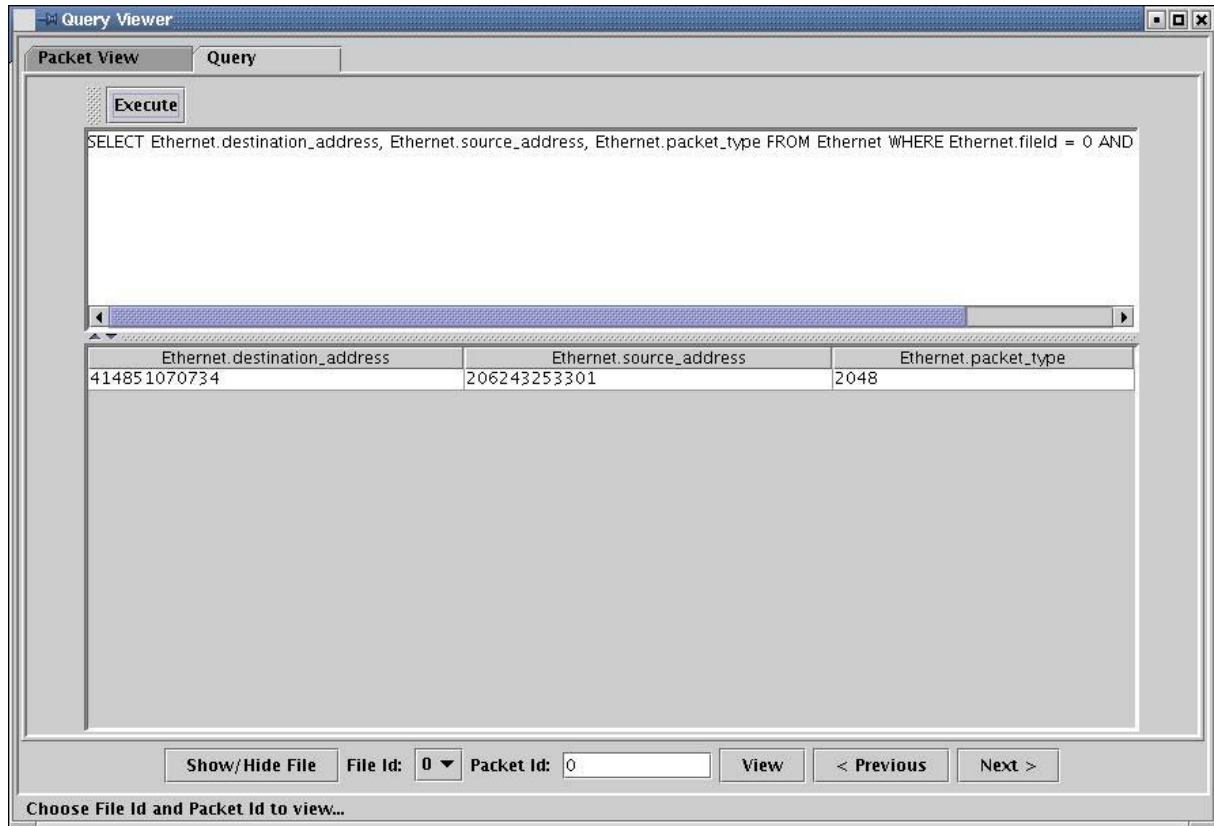


Figure 5.1 SANTA-G Viewer

Log files can also be viewed graphically by using the Viewer GUI Packet View panel. By using the drop down file ID box, and available file ID can be chosen. A packet ID may then be entered into the packet ID text field. By pressing view the selected packet will be displayed, see Figure 5.2. The log file can also be navigated by pressing the Previous and Next buttons, which will display the previous or next packet respectively.

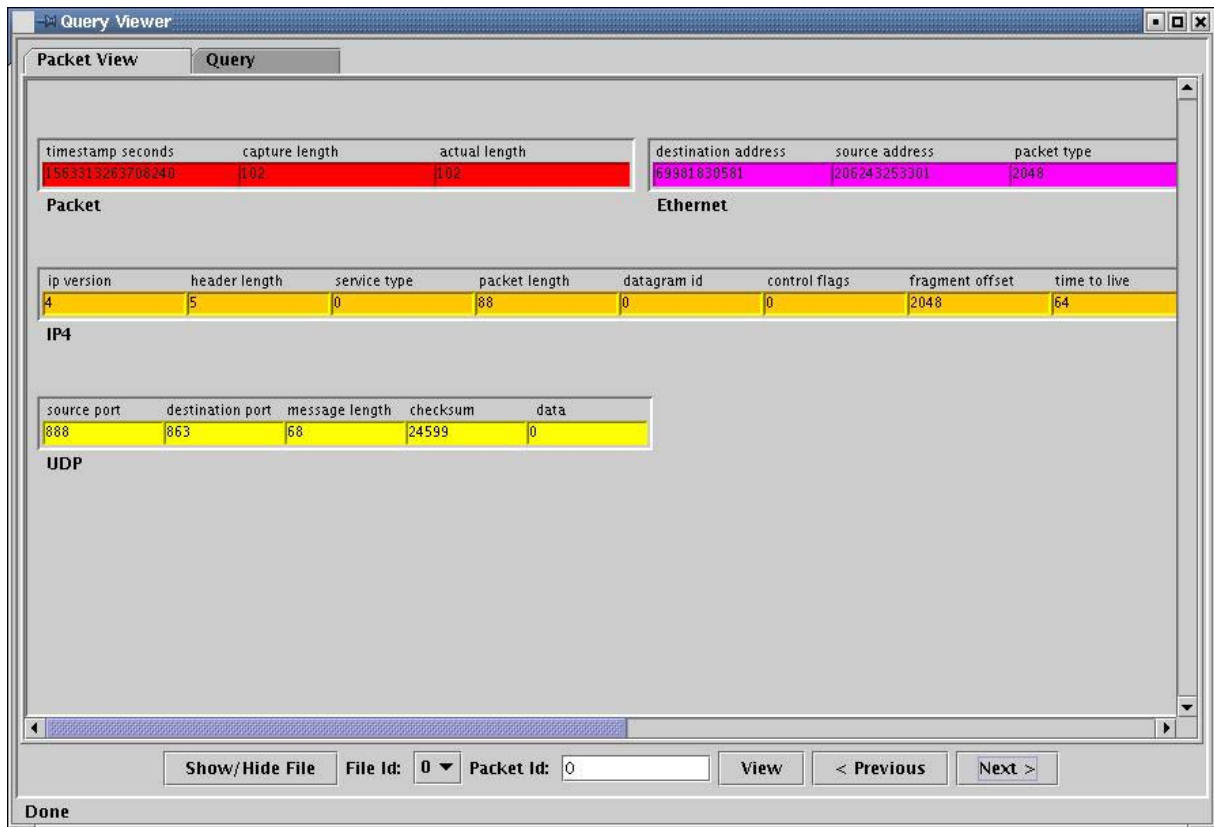


Figure 5.2 Packet viewed graphically

5.4. INTERFACE DESCRIPTION

The API for the CanonicalProducer and CanonicalProducer Servlet form part of the DataGrid R-GMA and can be found in the DataGrid WP3 website, please see:

<http://hepunix.rl.ac.uk/edg/wp3/documentation/doc/api/java/>

The SANTA-G NetTracer demonstrator is defined by its classes and code, which can be found in the FZK CVS repository <http://gridportal.fzk.de/>

6. INTERNAL TESTS

Thus far no systematic testing has been conducted, only functional testing during software development and debugging.

Please also see the Task 3.5 Tests and Integration document:

CG3.5-D3.3-v1.0-CSIC-TestIntegrationPrototype.

7. ISSUES

The primary issue is that the prototype is incomplete, although all the important components are working in accordance with the design document. Only a very limited initial Sensor implementation is provided. This should allow control of TCPdump – instead TCPdump must be run off-line. The Viewer GUI is also not complete, it has only limited SQL query support, and the graphical display also only supports a subset of available packets. Only a limited schema is supported by the Query Engine, and only a small subset of SQL is supported.